DISSERTATION
INTELLIGENT SYSTEMS

INTERPRETABLE ANALYSIS OF MOTION DATA

BABAK HOSSEINI

Bielefeld University,
Faculty of Technology,
Machine Learning Research Group

SUPERVISED BY
PROF. DR. BARBARA HAMMER

REVIEWED BY
PROF. DR. BARBARA HAMMER,
PROF. DR. XIAOYI JIANG

JUNE 29, 2021

ACKNOWLEDGEMENTS

This work was possible with the help of many people, both within and beyond the workgroup. First, I wish to thank my supervisor, Barbara Hammer, who has greatly supported my Ph.D. research throughout my time in Bielefeld. Additionally, I appreciate my reviewer Xiaoyi Jiang for his careful and in-depth reading of the manuscript.

I also extend my gratitude to two of my exceptional co-workers, Benjamin Paßen and Alexander Schulz, who have supported me in different technical and non-technical phases of my work. Beyond that, I owe appreciation to my external collaborators, Felix Hülsmann at TU Dortmund University and Romain Montagne at the University of Montreal. They have kindly contributed their time, knowledge, and skill to this research.

I also wish to thank my parents for their constant support, guidance, and trust over all these years, my friends for their confidence in me, and finally, my wife, who has helped me up whenever needed and who has made many sacrifices without which this work would not be possible.

Finally, I would be remiss in thanking the “Center of Cognitive Interaction Technology” (CITEC, grant number EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

ABSTRACT

Recent developments in motion capture technologies such as Microsoft Kinect and Vicon systems have facilitated motion data acquisition in diverse areas such as entertainment, sports, medical applications, or security systems. This type of data typically consists of recorded body parts' movement through time, which describes a semantically meaningful action to the domain experts. Coinciding notable growth in the size of available motion datasets, it is necessary to design machine learning methods to analyze motions regarding their underlying characteristics systematically.

Although many approaches have been suggested for motion analysis ranging from component analysis methods to deep learning algorithms, the majority of the state-of-the-art designs lack in providing semantically interpretable models. Such particular models for motion data contain building blocks that are connected to commonalities and particularities semantically understandable by domain experts. In this dissertation, I propose efficient algorithms to address the interpretable analysis of motion data from several significant aspects. These algorithms contribute to the state-of-the-art by introducing interpretable models in four specific categories of metric learning, sparse embedding, feature selection, and deep learning for the purpose of motion data analysis.

I propose a novel metric learning algorithm for motion data that benefits from a flexible time-series alignment. This algorithm can transfer motion data to another space in which semantically similar motions are located in tighter neighborhoods while semantically different motions are pushed further away from each other. A post-processing regularization of the learned metric reduces the usual existing correlations between the dimensions of the motion. As a result, the proposed model is interpretable by providing a small subset of dimensions (joints) that are closely relevant to the given discriminative task.

Furthermore, I present novel embedding frameworks that transfer the raw motion representation to a vector space. The resulting embeddings are non-negative vectorial representations that are sparse and semantically interpretable. They specifically carry understandable information about the encoded motions, such as the particularities of motion classes or commonalities of different motions. Additionally, I extend my proposed metric learning and embedding algorithms to different feature selection frameworks. In each framework, a sparse set of motion dimensions is selected that are semantically connected to the given overarching objective.

The last designed framework in my Ph.D. project focuses on using convolutional neural networks (CNN) to perform sequence-based labeling on motion data. More specifically, my developed deep learning algorithm introduces a novel CNN-based architecture benefiting from the time-series alignment concept in its filters. This framework learns local patterns in the temporal dimensions of the data. These temporal patterns are interpreted as significant parts of motion sequences, which lead to better discrimination of them.

I implement the above frameworks on different real-world benchmarks of motion data and analyze their performance from the above-discussed perspectives by comparing them to relevant state-of-the-art baselines.

CONTENTS

1 INTRODUCTION

Generally speaking, motion describes an object's movement in the real world, which changes its location or orientation (Elert 1998). In particular, this movement can be applied to living things such as humans and animals or artificial objects such as robots, vehicles, or other objects in the Universe. To be more specific, this work focuses on the motion of objects to which a skeleton body and a Kinematics model can be applied (Beggs 1983). Typical examples of such motions are movements of humans (Rosenhahn, Klette, and Metaxas 2008), animals (Muybridge 2012), insects (Woiwood, Reynolds, and Thomas 2001), or robots (M. Müller and Röder 2006). Accordingly, various fields of technology greatly benefit from or are constructed upon the study of motion-related information. As some examples, we can cite rehabilitation and physical therapy (Hueter-Becker and Doelken 2014), human gait analysis (Harris and Smith 2000), robotic motion planning (Latombe 2012), intelligent sports analysis (Sha et al. 2018), biomechanical studies (T.-W. Lu and C.-F. Chang 2012), and computational biology (Risse et al. 2017).

Therefore, great attention is paid toward the field of motion data analysis, which focuses on models and methods to explore important aspects of motion data related to its specific application (H. Zhou and H. Hu 2008; Durantin, Heath, and Wiles 2017; Arami et al. 2019; Jalal, Quaid, and K. Kim 2019; Bortolini et al. 2020; Ferdinands 2010)

With notable advances in current recording technologies for multimedia information, a considerable amount of motion data is available for any further processing (Pouyanfar et al. 2018). In general, motion data can be collected in various environments, such as uncontrolled daily human movements in public areas (Weinzaepfel, Martin, and Schmid 2016; Kuehne et al. 2011), expected movement scenarios like sports activities (Karpathy et al. 2014; Soomro, Zamir, and Shah 2012), controlled laboratory-based motion experiments (Mandery et al. 2015; Liang et al. 2020), and many other imaginable scenarios. Regardless of the motion's source, in a motion capture system (mocap), an object's movement is recorded and is represented by multi-dimensional digital signals (Liang Wang, L. Cheng, and G. Zhao 2010). Currently, several technologies are available for capturing motion information, which can be generally categorized as optical systems (Guerra-Filho 2005), inertial systems (Roetenberg et al. 2013), mechanical motion (Rahul 2018), and Magnetic systems (Yabukami et al. 2000). As the widely used mocap systems for research-based purposes, we can mention the marker-based optical Vicon (Oxford, UK) technology (Merriaux et al. 2017) and marker-less infrared Kinect^{TM} system (Smisek, Jancosek, and Pajdla 2013). Hence, depending on the utilized mocap technology, the captured motion data's raw representation could be different. However, as a common feature among motion representations, the captured motion is projected on a skeleton structure, which corresponds to the body links and joints of the moving subject Figure 1.1. Specifically, powerful analysis software enables the reconstruction of this underlying skeleton when dealing with human motion (Bregler 2014; Joon 2010; Baak et al. 2013).

In the area of machine learning, many works specifically focus on the analysis of motion data. Generally, based on the particular purpose and the target application, it is possible to categorize these algorithms and methods into individual groups such as motion classification (Bodor et al. 2009; Cao et al. 2004; Jalal, Quaid, and K. Kim 2019), learning motion primitives (Kulić et al. 2012; Hauser et al. 2008; Saveriano, Franzel, and

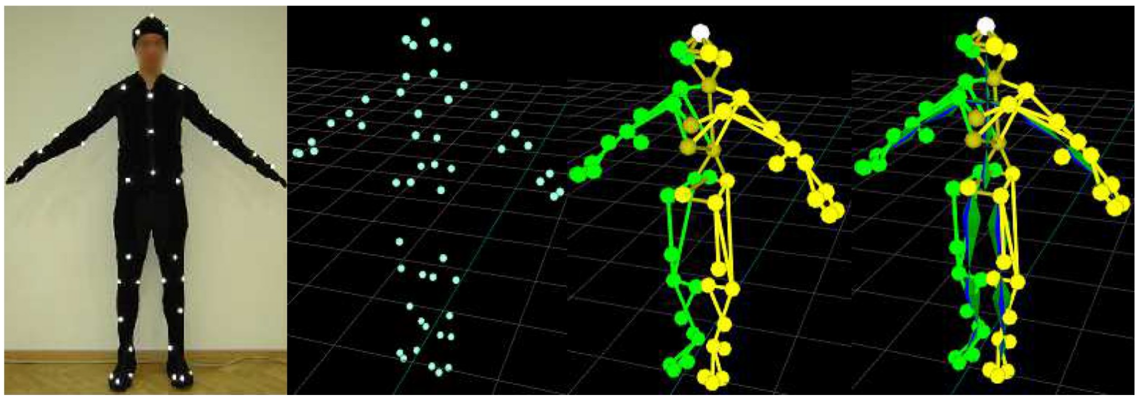


Figure 1.1: Capturing joint information of a subject using a marker-based mocap system and forming its skeleton-based posture description. The image is taken from (Pawlyta and Skurowski 2016).

D. Lee 2019), motion generation (Arikan and Forsyth 2002; Y. Yan et al. 2017; Z. Xie et al. 2020), retrieval of motion sequences (Kapadia et al. 2013; F. Liu et al. 2003; Q. Xiao and C. Chu 2017), and motion clustering (Torr and Murray 1994; F. Zhou, De la Torre, and Hodgins 2012; C. Xie et al. 2019). Such machine learning models' notable ability in coping with noise, adjusting to specifics of persons and measurement, and gradual adaptation to new observations makes them constitute a particularly promising approach for these motion analysis tasks. In each category, state-of-the-art approaches are usually compared based on their performance in fulfilling their given tasks and their computational/space complexity.

Regardless of the reported considerable performance of machine learning models in analyzing motion data, they are often not easily interpretable. Popular examples of such models belong to the family of deep neural networks, which constitute the state-of-the-art in the majority of the mentioned fields (Si et al. 2018; Z. Xie et al. 2020; C. Xie et al. 2019). These algorithms are generally complex in explaining their underlying decision-making process, which renders them unsuitable, specifically for a practitioner or a domain expert. Apart from the usual complexity and performance measures, another essential characteristic of a machine learning algorithm is its trained model's interpretability. Model Interpretability focuses on investigating how a trained model makes its prediction or inference (Molnar 2020; Doshi-Velez and B. Kim 2017). In practice, interpretability is an essential characteristic for domain practitioners and helps them understand the designed model's decision-making mechanism (Murdoch et al. 2019). Relatively, in natural language processing (NLP), semantic interpretation is considered a mapping between synthetically analyze information and the meaningful concepts, for humans, it carries (Hirst 1992). This concept plays a vital role in various NLP applications, such as named entity recognition and retrieval of semantically related words (Şenel et al. 2018).

By transferring semantic interpretation to motion data analysis in a broad sense, we investigate a connection or a mapping between the model mechanism and semantically meaningful information related to motion data (Hosseini and Hammer 2019b; Liang Wang, L. Cheng, and G. Zhao 2010; V. Krüger et al. 2007). Such a characteristic increases the model's usability for practitioners and areas such as human-computer interaction (Finlay 1997; Gillies et al. 2016; Mohseni, Zarei, and Ragan 2018; Gates et al. 2019). For example, an interpretable embedding of a motion sample should relate the resulting vector's entries to partial or complete movements semantically similar to the original motion sample. To be more specific, by interpretable motion analysis, we investigate a captured movement for its meaningful characteristics for humans. For instance, when we call an observed movement *walking*, we apply certain characteristics to that motion that semantically represent a walking movement in our mind. This means a specific body formation, particular types of joint movements, and the body parts relevant to us when we call it *walking*. Nevertheless, the amount and the depth of such characteristics depend on the context and the goal of our observation. Such motion models enable a meaningful retrieval of motion databases since it enables a blending of high-level semantics and low-level signals. Despite the recent advances in motion analysis methods, the above concept has not yet been explored for them properly. Therefore, assigning such characteristic to the existing models will significantly improve their usability in practical motion analysis problems.

When comparing the samples in a dataset, we can consider two given entities semantically similar when they carry the same semantic meaning (Vigliocco, Vinson, and Siri 2005; Lord et al. 2003; Kandola, Cristianini, and Shawe-taylor 2002). Such a concept can play a significant role in designing machine learning algorithms that lead to interpretable models by employing the nearest neighbor search in the space. Talking about the semantic similarity of motion data, we can categorize the motion samples into specific groups, such that each group contains motion sequences that are based on human interpretation more similar to each other than to sequences from other groups. At first sight, this process might seem approachable by applying any advanced sequential-data classifier to motion data (Fawaz et al. 2019). However, to have an interpretable model based on semantic similarity, we are more interested in two other important aspects:

1. Using a proper technique that determines if two given motions are semantically similar with to a relative extent.
2. Obtaining a motion representation that facilitates the interpretability of the decision making process by emphasizing the above semantic relationship between motion sequences.

The first concept can be addressed by measuring the **distance** between two given data points (x, y) as a distance function $d(x, y)$. For two semantically similar points (x, y), it is expected that $d(x, y)$ is considerably small compared to $d(x, z)$, where x and z are not semantically related. A typically used distance measure for vectorial data is the Euclidean distance $\|\vec{x}^2 - \vec{y}^2\|_2$ (Bellet, Habrard, and Sebban 2013). However, for sequential data forms, other techniques such as dynamic time warping (DTW) (Berndt and Clifford 1994) and sequence alignment kernel (Saigo, Vert, and Akutsu 2006) are popularly used, which generally focus on the temporal alignment of two given motions $(\vec{x}, \vec{y})$.

Relevantly, distance-based algorithms such as k-nearest neighbor classifier (kNN), k-means clustering (Bishop 2006), and learning vector quantization (Kohonen 1995) categorize data samples based on their similarity to other locally nearby samples (or prototypes) in the given data distribution. The relative location of data points in the data distribution is measured and analyzed by their pairwise distance $d(\vec{x}, \vec{y})$. Application of those techniques to motion data provides the interpretable classification of motion samples by relating each motion sequence to its nearby semantically similar data points (Switonski, Josinski, and Wojciechowski 2019; Petitjean, Forestier, Geoffrey I Webb, et al. 2016; Keskin, Cemgil, and Akarun 2011). Nevertheless, they do not influence the data representation in terms of altering the motion distributions such that they better respect such similarity in their neighborhoods.

On the other hand, metric learning algorithms focus on changing the original data representation to emphasize the similarity of semantically related data points (Bellet, Habrard, and Sebban 2013; Kulis 2012). More specifically, current metric learning methods focus on leaning a linear transform $\mathbf{L}\vec{x}$ that modifies the Euclidean distance as $d_{\mathbf{L}}(\vec{x},\vec{y}) = \|\mathbf{L}\vec{x} - \mathbf{L}\vec{y}\|_2$. The metric coefficient matrix $\mathbf{L}$ is learned with the objective to reduce $d_{\mathbf{L}}(\vec{x},\vec{y})$ for data points that are considered semantically similar while increasing $d_{\mathbf{L}}(\vec{x},\vec{y})$ for the semantically distinct points. Ideally, such an objective should yield a new representation in which similar data points are gathered in condensed and separated local neighborhoods in the data space. Such representation considerably enhances the interpretability of distance-based classifiers such as kNN by focusing on those condensed local neighborhoods.

Metric learning has shown promising applications in various areas of machine learning and data analysis, such as face recognition (Guillaumin, Verbeek, and Schmid 2009), deep learning (Jian Wang et al. 2017), information retrieval (McFee and G. Lanckriet 2010), and human activity recognition (Tran and Sorokin 2008). However, current metric learning algorithms are mostly in favor of vectorial representation of data points, in which $\vec{x}$ and $\vec{y}$ are vectors in $\mathbb{R}^d$. The available metric learning algorithms for structured data are mostly applicable to string and tree data forms (Bellet, Habrard, and Sebban 2013). This limitation is a considerable practical barrier for applying metric learning on structured data such as motions. Accordingly, in order to benefit from metric learning for interpretable analysis of motion data, I pose the first research question of my work as the following:

RQ1: Can we apply metric learning on motion data to enhance the interpretability of distance-based decision making processes based on the resulted data representation?

To answer this question, I propose a distance-based metric learning framework in Chapter 3, which brings similar motion samples closer while pushes away dissimilar data points. In this framework, I use pairwise distances of motion samples as the input, computed by the DTW algorithm as an elastic alignment technique. My metric learning framework benefits from DTW's unique characteristic, a robust alignment of semantically similar motions (generally time-series). In the space created by the newly learned metric, each motion sample is assumed to be surrounded mostly by other motions of its type. In addition to the above interpretation, I use a regularization method to clear the obtained metric parameters from existing redundant information (Frénay et al. 2014; Strickert et al. 2013). This post-processing step determines the motion dimensions (body joints) that are significantly relevant to the given task and have more effect on the decision making process.

Another concern with temporal data such as motion sequences is their space complexity (M. Kim et al. 2019; Deri, Mainardi, and Fusco 2012; Tahmassebpour 2017). Generally speaking, motion data is created out of recorded frames of body gestures in the temporal axis. Therefore, the captured motion can quickly become spacious due to an increase in the frame-rate of recording, size of extracted data per frame, or the length of a movement (M. Müller 2007). The space complexity can be a barrier to utilizing many advanced algorithms designed mainly for the vectorial source of information, especially if we apply them directly to the vectorized raw motion representation (Glardon, Boulic, and

Thalmann 2004; Guodong Liu and McMillan 2006). One effective solution for such cases is to find an **embedding to the vector space**, which considerably reduces data representation's space complexity. This idea has already shown its success in different application areas, such as word embedding (Yang Li and T. Yang 2018), graph embedding (H. Cai, V. W. Zheng, and K. C.-C. Chang 2018), and computer vision (Schroff, Kalenichenko, and Philbin 2015). Accordingly, other works such as (S. Li, K. Li, and Fu 2015; Zhen et al. 2013; C. Kong and Lucey 2019; Zhao Wang, Y. Feng, S. Liu, et al. 2016) applied that idea to motion data, which resulted in encapsulated representations. Such sparse encoded vectors could be used efficiently for any further supervised or unsupervised motion information analysis.

Despite the above achievements in obtaining sparse embeddings for motion and other structured data, a practical demand is to have representations that are understandable w.r.t. the semantic information they carry related to the original data source (T. Chen et al. 2018; N. Li et al. 2018). Transferring this concept to motion data, a practitioner requires an interpretable embedding for a given motion sample. The constituent elements of the resulting representation should be connected to specific movement information that is semantically related to the original motion. In (Yale Song and Soleymani 2019), this concept is addressed for image embedding by learning several local embeddings for each given image, which provide partial semantic descriptions for that image. Despite the promising performance of the available motion encoding algorithms in reducing the space complexity, as a typical observation, extracting useful information out of their representations is difficult or maybe impossible in general cases. According to this specific concern, I formulate my second research question as:

RQ2: Can we obtain a rich embedding of motion data that is sparse and interpretable regarding its entities?

In Chapter 4, I introduce non-negativity constraints into a specific type of sparse coding framework. I demonstrate that the resulting novel framework can encode each motion data by relating it to other semantically similar motions. The obtained encoding is interpretable in terms of the motion class to which each encoded motion sample is mainly related. I show that my sparse coding framework and its novel supervised and unsupervised extensions result in more interpretable and better discriminating sparse vector encodings throughout empirical evaluations.

As explained, motion data describes the movement of different body joints, which can be generally addressed as motion dimensions. From that perspective, another relevant and practical question is that if we can analyze a motion sequence based on the information that exists in its individual dimensions. In machine learning, the above question is explored as the **feature selection** problem. The goal of feature selection is to select a sufficiently small set of features (dimensions) from the given data while maximizing (or minimizing) another given objective (Kumar and Minz 2014; Alelyani, J. Tang, and Huan Liu 2013). According to clinical studies, there is a specific correlation between different body joints' movements, even in partial-body actions. For example, in a proper throwing action, the torso and the lower body parts have a synchronized movement with respect to the arm motion (Raine, Meadows, and Lynch-Ellerington 2013; Janet and Roberta 2003). Extending this observation to other human motions, considering only a subset of body joints is sufficient to represent or categorize a motion type. Upon this assumption, different methods have utilized feature selection in motion analysis problems such as motion retrieval (Zhao Wang, Y. Feng, Qi, et al. 2016), classification of motions (Z. Yan, Zhizhong Wang, and H. Xie 2008), and motion reconstruction (Kusakunniran et al. 2010).

Therefore, an interpretable model for motion representation selects relevant motion dimensions that have an underlying semantic relationship with the model's primary purpose. In other words, we expect to observe a understandable connection between those selected body joints and the given motion analysis task (Hosseini and Hammer 2015). This specific objective has not been addressed yet in any feature selection methods related to motion data. Even though I show in Chapter 3 that we can obtain a set of relevant body joints to the given classification task using post-processing techniques, we are still interested in models that actively consider the existing redundancy or particularity of information in the motion dimensions. Also, from the perspective of vectorial motion encoding (**RQ2**), it is of specific interest to learn the motion dimensions that directly correspond to the encoding objective. Even as a more optimal framework, the goal is to find the features that considerably facilitate obtaining such desired enriched sparse embedding. According to the above problem specifications, the following relevant question will be raised.

RQ3: How can we extend existing motion analysis models to interpretable feature selection frameworks, which are formulated by also respecting the main objectives of these models?

In Chapter 5, I demonstrate that we can extend Chapters 3 and 4's frameworks to more general formulations by considering multi-dimensional motion data as a multiple-kernel source of information (Gönen and Alpaydın 2011). In the models that I propose in this chapter, I actively involve individual motion dimensions in the learning process of the model. To be more specific, I address this problem by finding a sparse kernel combination corresponding to the scaling of feature space. Therefore, the scaling parameters reflect the relevance of each component's kernel information and its corresponding motion dimension. Accordingly, the multiple-kernel extension of my distance-based metric learning algorithm focuses on *learning* a sparse set of dimensions, which result in dense neighborhoods of semantically similar motions.

Besides, I propose two other multiple-kernel frameworks in Chapter 5 as the extensions of my non-negative sparse coding framework (**RQ2**). The first framework finds a set of features based on which motions can be efficiently represented by a prototype-based model. In this model, both the prototypes and the vector encodings are considerably interpretable in terms of their basis motion classes. The second framework learns meaningful motion attributes, each of which focuses on the movement of a subset of body joints (motion dimensions). I demonstrate that we can partially encode (and also categorize) an unobserved motion type with the help of these learned attributes, which mainly focus on the possible semantic similarity of different motion types in the movements of particular joint groups.

One fundamental assumption for my interpretable frameworks related to previous questions is that we need to **segment** motion data in advance. More specifically, the distance-based information as the input for my metric learning or sparse coding models can be computed when motion sequences are temporally synchronized. Even though my presented solutions can take motions of different lengths, to compute a valid input for the models, all motion sequences in the dataset need to contain the same number of movement cycles. This problem is addressed in the literature as temporal segmentation of motion data, in which a long stream of motion is split into individual meaningful subsequences as motion segments (F. Zhou, De la Torre, and Hodgins 2008; Spriggs, De La Torre, and Hebert 2009). Several unsupervised methods have been suggested to segment motion sequences into their constituent temporal parts (B. Krüger et al. 2017; F. Zhou, De la Torre, and Hodgins 2013; C. Lu and Ferrier 2004; Qifei Wang et al. 2015). These unsupervised algorithms do not require pre-annotated data. However, they usually result in over-segmentation of a motion stream into its sub-components due to their unsupervised structure (B. Krüger et al. 2017; F. Zhou, De la Torre, and Hodgins 2013).

The supervised segmentation of temporal data is mainly investigated in other domains such as speech recognition and text analysis, typically referred to as sequence labeling (Gehring et al. 2017; Akbik, Blythe, and Vollgraf 2018; X. Ma and Hovy 2016). Such algorithms usually benefit from deep neural networks' feature extraction power to both segments and classify the input sequence's time-frames. Nevertheless, the application of these methods on skeleton-based representations is not straightforward and usually demands hand-coded modifications to the model's structure.

Additionally, these methods do not provide an expected level of interpretation related to certain existing properties of the given skeleton-based input motion. For instance, it is desirable to observe a semantic connection between different joints' movements of a full-body motion, such as walking, and the decision making process of the network (Hosseini, Montagne, and Hammer 2019). Accordingly, the above discussion begs the following research question:

RQ4 How can we design a deep neural architecture for the segmentation of motion sequences, which is interpretable based upon semantic components of motion data?

In another closely related area of sequential data analysis, it is shown that by learning specific exemplar (sub-)sequences, we can retrieve or classify their other semantically similar sequences in the dataset (Rakthanmanon and E. J. Keogh 2013; Petitjean, Forestier, Geoffrey I Webb, et al. 2016; L. Ye and E. Keogh 2009). In (C. Ji et al. 2019; L. Ye and E. Keogh 2009), relatively short time-series are learned as prototypes and can identify other longer sequences that belong to one specific group of time-series. (Yeh 2018) discusses that these temporal prototypes contain meaningful information with respect to the overarching analysis objective, which makes the decision-making model highly interpretable. Inspired by these works, I emphasize that another valuable semantic information that one seeks in motion data lies in the temporal aspect of the data. Despite the success of the works mentioned above in time-series problems, they cannot be directly applied to skeleton-based sequences, especially for segmentation purposes.

On the other hand, several deep neural architectures have shown promising results in the classification of motion data, specifically skeleton-based human mocap sequences (Si et al. 2018; J. Liu, Shahroudy, et al. 2018; H. Wang and Liang Wang 2017; S. Song et al. 2017). Despite their notable discriminative performance, which is resulted from their complex architectures, these models suffer from weak **interpretability**. This issue prevents them from being exploited by domain-specialists and practitioners. Putting the above two views together, I formulate my last research question as:

RQ5: Can we design a deep neural network architecture to find relevant information in the temporal aspect of motion data, leading to an interpretable deep neural model?

To address the research questions **RQ4** and **RQ5**, I design a novel convolutional neural network (CNN) in Chapter 6 that is applicable to motion inputs of different lengths and their sequential concatenations. This network can take in a long stream of different motion sequences and determine the temporal location and type of its constituent motions. More specifically, I design the structure of my CNN model such that it finds significant subsequences in the motion data that are semantically meaningful. I address them as temporal prototypes. These prototypes enhance the interpretability of the network by indicating the relevant movement patterns in specific body joints, resulting in the separation of different motion types according to a given classification task. Through empirical evaluations on large-scale human action recognition benchmarks, I show that my proposed convolutional model obtains comparable classification and segmentation accuracy to the state-of-the-art deep neural architectures while is also interpretable based on the temporal patterns it finds in the given motion data.

In summary, my work contributes to state-of-the-art:

- A distance-based metric learning framework applicable to motion data, which concentrates the semantically similar motion samples in the distance space. Furthermore, the regularization of this metric interprets the joints that are semantically relevant to the given classification task.
- A non-negative sparse coding framework and its variations that provide interpretable encodings of motion samples by relating each motion to its semantically similar samples.
- Novel multiple-kernel learning frameworks to learn interpretable dimension-based models for motion data with respect to different goals, such as classification, partial reconstruction, and prototype learning.
- A deep architecture that learns discriminant temporal prototypes for interpretable segmentation and classification of full-body motion data by proposing a novel deep learning architecture.

I have presented individual parts of this work in different renowned international venues. More specifically, the works which are covered by this thesis are presented in the following listed publications.

Conference Publications:

- Hosseini, Babak and Barbara Hammer (2015). "Efficient metric learning for the analysis of motion data". In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.
- Hosseini, Babak, Felix Hülsmann, et al. (2016). "Non-negative kernel sparse coding for the analysis of motion data". In: *International Conference on Artificial Neural Networks (ICANN)*. Springer, pp. 506–514.
- Hosseini, Babak and Barbara Hammer (2018a). "Confident kernel sparse coding and dictionary learning". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1031–1036.

- — (2018b). “Feasibility Based Large Margin Nearest Neighbor Metric Learning”. In: *26th European Symposium on Artificial Neural Networks (ESANN)*.
- — (2018c). “Non-negative Local Sparse Coding for Subspace Clustering”. In: *Advances in Intelligent Data Analysis XVII. (IDA)*. Ed. by Ukkonen A. Duivesteijn W. Siebes A. Vol. 11191. Lecture Notes in Computer Science. Springer, pp. 137–150. DOI: 10.1007/978-3-030-01768-2_12.
- — (2019b). “Interpretable Multiple-Kernel Prototype Learning for Discriminative Representation and Feature Selection”. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM.
- — (2019c). “Large-Margin Multiple Kernel Learning for Discriminative Features Selection and Representation Learning”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE.
- — (2019d). “Multiple-Kernel Dictionary Learning for Reconstruction and Clustering of Unseen Multivariate Time-series”. In: *27th European Symposium on Artificial Neural Networks (ESANN)*.
- Hosseini, Babak, Romain Montagne, and Barbara Hammer (2019). “Deep-Aligned Convolutional Neural Network for Skeleton-based Action Recognition and Segmentation”. In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE.

Journal Publications:

- Hosseini, Babak, Romain Montagne, and Barbara Hammer (2020). “Deep-Aligned Convolutional Neural Network for Skeleton-Based Action Recognition and Segmentation (extended article)”. In: *Data Science and Engineering*. ISSN: 2364-1541. URL: https://doi.org/10.1007/s41019-020-00123-3.

The structure of my thesis is as follows. In Chapter 2, I discuss the notations and foundation of the work. Chapter 3 describes the interpretable metric learning framework for the classification of motion data along with its mathematical improvement. In the next chapter, I present my proposed sparse coding models, which learn semantically interpretable encodings. Chapter 5 studies the dimension-based analysis of motion data and presents my interpretable multiple-kernel models. My proposed deep temporally interpretable CNN architecture is described and evaluated in Chapter 6, and finally, I conclude the work in Chapter 7 together with discussing its outlook.

FOUNDATIONS 2

In this chapter, I explain the data representation and notations that we use throughout this work. Furthermore, I explore the standard overarching technologies for analyzing motion data, the formulation of the concerns and challenges of this field of study that I address in this work, and the benchmarks that I use to evaluate my algorithms empirically.

2.1 MOTION DATA REPRESENTATION

Regardless of different existing technologies to record motion data, there are two generally used methods to capture motions: extracting motions from recorded videos and capturing movement by mounted body sensors or markers. In the first method, the subject's video is recorded by a conventional or special camera, and a skeleton structure is fitted to the subject's body in each video frame. Hence, the recorded motion data consists of the movements related to the skeleton structure's different joints during these video frames. This group of mocap methods is often considered as low-cost marker-less technologies (Spiro, Huston, and Bregler 2012; Y. Liu et al. 2013). Specifically, the Kinect$^{\text{TM}}$ motion capture system (Smisek, Jancosek, and Pajdla 2013) is one of the pioneer technologies which uses such a mocap approach.

The other way of capturing motion information is carried out by mounting several markers (sensors) on different locations of the subject's body (usually near the joints)(Figure 1.1). Then either one or several specially designed cameras track these markers' movement, or these markers transmit their movement information to a computer. Respectively, Vicon (Oxford, UK) technology (Merriaux et al. 2017) and inertial measurement unit (IMU) system (Roetenberg et al. 2013) are examples of these marker-based technologies.

Regardless of the utilized technology, the motion information can be generally represented by a multivariate times series

$$\mathbf{X} = (\vec{x}(1) \dots \vec{x}(T)) \in (\mathbb{R}^d)^*. \tag{2.1}$$

In Equation 2.1, T denotes the time series' length, which can have arbitrary but finite values within a given mocap dataset, and $\vec{x}(t) \in \mathbb{R}^d$ represents the vector of joint values at time frame t. As illustrated in Figure 2.1, $\mathbf{X}$ consists of d time-series related to different body joints' movement. Depending on the utilized mocap technology, there is a mapping between each skeleton joint and some dimensions of $\mathbf{X}$. For instance, in a 3D point cloud representation (K.-C. Chan, Koh, and C. G. Lee 2014), each joint's coordinates are represented by three specific dimensions of $\mathbf{X}$. There are three dimensions per joint in a three degree-of-freedom Euler angles coordinate system (Beggs 1983). However, this number becomes four in a quaternion representation system (Q. Liu, Prakash, et al. 2003). Although the number of motion dimensions (d) is generally bigger than the number of joints, I may interchangeably use them in this document to facilitate the understanding of the main concepts.

Even though motion data is commonly assumed as a full-body motion, we can represent specific local joints' movement by a matrix $\tilde{\mathbf{X}} \in (\mathbb{R}^{\tilde{d}})^*$ where $\tilde{d} < d$. The multivariate time-series $\tilde{\mathbf{X}}$ is formed through a row-selected submatrix of $\mathbf{X}$, where $\tilde{d}$ corresponds to the local joints involved in the motion (Jun Wang et al. 2013; Ko et al. 2005). The pairwise mapping of joint information to time-series dimensions could be different from one dataset to another. However, it is essential to have this representation consistent for all motion samples within the same dataset. To extend the notation setting for this work, for a given matrix $\mathbf{A}$, column-vector $\vec{a}_i$ denotes its i-th column, row-vector $\vec{a}^j$ or $a(j,:)$ denotes its j-th row, and scalar a_{ji} refers to the j-th entry in $\vec{a}_i$.

2.2 FORMULATING MOTION ANALYSIS PROBLEMS

This section explains the base formulations for the particular data-driven problems related to motion analysis that I address in the following chapters of this thesis. These problems consist of motion classification, feature selection, motion embedding, and motion representation.

Motion Classification

One typical way to analyze a motion dataset is via a classification task, in which a model is designed to project the supervised information into the motion samples correctly (Alpaydin 2020). In such a setting, we have a training set $\mathcal{X} = \{\mathbf{X}_i\}_{i=1}^{N}$ containing N motion samples $\mathbf{X}_i$, and its corresponding binary class label matrix $\mathbf{H} = [\vec{h}_1, \ldots, \vec{h}_N] \in \{0,1\}^{C \times N}$. Accordingly, each $\vec{h}_i$ is a zero vector except in its q-th entry where $h_{qi} = 1$ if $\vec{x}_i$ belongs to motion class q in a C-class setting. Hence, the classification model finds a mapping $\mathcal{C} : \mathcal{X} \to \mathbf{H}$, which is evaluated by finding the class labels for a test set $\mathcal{Z}$ from motion data given that $\mathcal{Z} \cap \mathcal{X} = \emptyset$.

From a another point of view, the classifier $\mathcal{C}$ projects $\mathcal{X}$ to a space spanned by columns of $\mathbf{H}$, in which semantically similar motions have identical representations. In a more relaxed formulation, a mapping $\mathcal{M} : \mathcal{X} \to \tilde{\mathcal{X}}$ is desired that projects motion data to

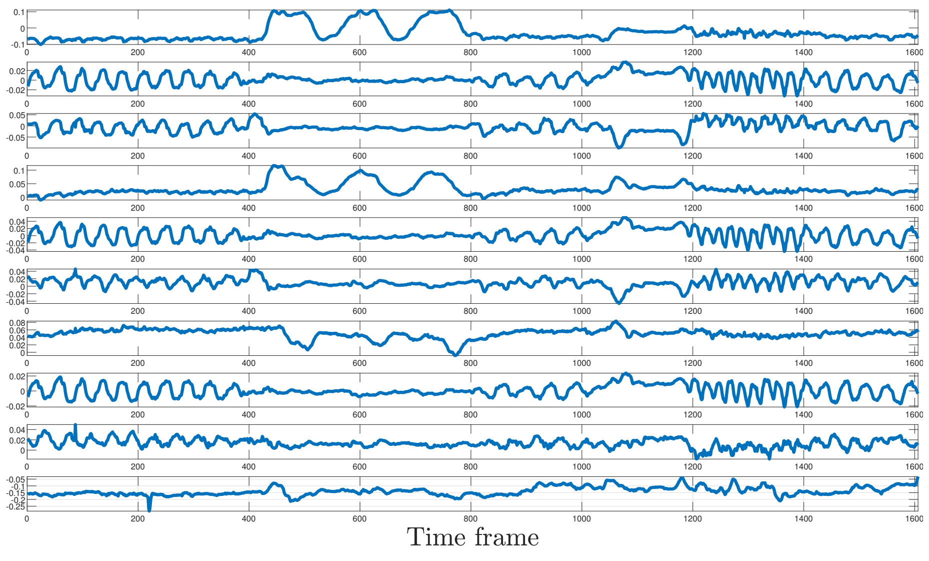


Figure 2.1: Representation of the captured motion data by a multivariate time series. Here a subset of motion dimensions is plotted.

another space, where semantically similar motions are not identical but yet locally close. This view leads us to interpretable distance-based frameworks such as the family of metric learning approaches (Bellet, Habrard, and Sebban 2013). These methods generally try to learn a metric that makes the pairwise distance between semantically similar data points smaller while resulting in larger distances for distant samples. This objective is effective for local neighborhoods of the data distribution and makes the decision making process directly interpretable by the label of nearby neighbors. Therefore, the learned metric can be seen as a linear or non-linear projection $\mathcal{M}$ of data points into a target space, where semantically similar data points are locally closer to each other than the original space (Bellet, Habrard, and Sebban 2013). A popular way of evaluating such mapping's effectiveness is to use a neighborhood-based method such as the k-nearest neighbor (kNN) classifier (Goldberger et al. 2005). The kNN method determines the class of a motion $\mathbf{Z} \in \mathcal{Z}$, relying on the assumption that $\mathbf{Z}$ is locally surrounded by motion samples to which it is semantically similar. Therefore, finding a proper mapping can make the decision making process interpretable by definition.

Feature Selection

Another challenging area related to motion data analysis is performing a feature selection for such type of data. Denoting $\mathbf{S}$ as the feature set of $\mathbf{X}$, we can define the feature selection problem as a multi-objective optimization:

$$\tilde{\mathbf{S}} = \underset{\tilde{\mathbf{S}} \subset \mathbf{S}}{\arg\min} \quad (|\tilde{\mathbf{S}}|, f(\mathbf{X}|_{\tilde{\mathbf{S}}})) \tag{2.2}$$

In Equation 2.2, $\mathbf{X}|_{\tilde{\mathbf{S}}}$ denotes the submatrix of $\mathbf{X}$ formed by selecting the subset $\tilde{\mathbf{S}}$ from the rows of $\mathbf{X}$. The quality of the selected features $\tilde{\mathbf{S}}$ is evaluated by a function or measure $f(.)$. Typically, $f(.)$ is an objective to be minimized according to a defined supervised (Dash and Huan Liu 1997) or unsupervised (Alelyani, J. Tang, and Huan Liu 2013) task. Hence, it is always expected to observe Pareto optimal solutions for $\tilde{\mathbf{S}}$ as a trade-off between minimizing the size of selected features and fulfilling the given task.

In feature selection, when the solution to a given task relies on an existing semantic relationship between motion samples, it is expected that a sparse $\tilde{\mathbf{S}}$ leads to a highly interpretable model by reflecting the semantic connection between the selected features and the overarching task (Hosseini and Hammer 2019c). As an example for motion data, when the task is distinguishing between different leg movements, it is anticipated from a sparse feature selection to choose the dimensions of $\mathbf{X}$ that are mostly related to foot and leg joints. On the other hand, if two given motions $\mathbf{X}_i$ and $\mathbf{X}_j$ are similar only in the movement of a specific set of body joints, an interpretable feature selection model would find a sparse set $\tilde{\mathbf{S}}$, which contains mostly dimensions from the shared body joints between $\mathbf{X}_i$ and $\mathbf{X}_j$ (Hosseini and Hammer 2019d).

Motion Embedding

Another typical problem associated with motion data, as well as other structured data types, is that the raw representation of $\mathbf{X}$ is not mathematically suitable for many machine learning algorithms. In fact, such algorithms often require inputs from a vector space of fixed and finite dimensionality. In some works, the multivariate raw form of the motion (matrix $\mathbf{X}$) is directly used as the input to the main algorithm (Glardon, Boulic, and Thalmann 2004; S. Li, K. Li, and Fu 2015). However, it is shown for structured data that a proper embedding to a vector space, as $\mathcal{E} : \mathbf{X} \to \vec{\gamma}$ where $\vec{\gamma} \in \mathbb{R}^k$, can considerably reduce the space complexity as well as the computational complexity of the next level algorithm (Y. Ma and Fu 2011; Yang Li and T. Yang 2018; H. Cai, V. W. Zheng, and K. C.-C. Chang 2018). Furthermore, a sparse embedding of $\mathbf{X}$ can result in a vector $\vec{\gamma}$ with a limited number of non-zero entries (Zhao Wang, Y. Feng, S. Liu, et al. 2016; M. Zhang and Sawchuk 2013). Such embedding reduces the existing underlying redundancy in the raw data. At the same time, $\vec{\gamma}$ is rich enough by carrying mostly the relevant properties of $\mathbf{X}$. Similar to the feature selection problem in Equation 2.2, we can evaluate an embedding method by considering the trade-off between the sparseness and quality of the resulting $\vec{\gamma}$ in satisfying the given supervised or unsupervised task.

Analyzing a motion embedding $\mathcal{E}$ from the interpretable point of view, we are interested in finding embedded vectors $\vec{\gamma}$ with entries that are semantically understandable. For instance, a $\vec{\gamma}$ representing a *walking* motion sample should contain specific non-zero entries that we can consider as the particularities of the *walking* class of motion. Also, depending on the model design, some entries in an interpretable $\vec{\gamma}$ may refer to particular movement types only related to a specific subset of body joints (Qiu, Z. Jiang, and Chellappa 2011; Hosseini and Hammer 2019d). Evaluation of such properties is possible by designing measures that particularly depend on the existence of these meaningful characteristics in the encoding $\mathcal{E}$.

Motion Representatives

Another problem associated with the analysis of motion data is learning a prototype-based representation for $\mathcal{X}$. Generally speaking, a prototype based representation aims for a small number of exemplary time series or generalizations thereof, which can serve as representatives for all data within a given set. The suitability of such representatives can be measured, e.g., in terms of the information contained in the set that can be covered by the prototypes already, or approximations thereof such as the quantization error. With that perspective, there are generally two different yet connected interpretations from the concept of motion prototypes. In a group of works, a (motion) prototype is either a real exemplar $\mathbf{X}_p \in \mathcal{X}$ (affinity propagation) or a virtual signal $\mathbf{Q} \in \mathbb{R}^{d \times *}$ (kernel GLVQ, K-means) created as a combination of a set $\mathbf{P}$ of samples $\{\mathbf{X}_p\}_{p \in \mathbf{P}} \in \mathcal{X}$ (Guan et al. 2011; Schleif et al. 2011; Bishop 2006; Nienkötter and X. Jiang 2016). We can treat this type of prototype as a representative for a subset of samples (motions) in $\mathcal{X}$ to which it is semantically similar, or they can be described efficiently based on this prototype (Hosseini and Hammer 2019a). In a different research branch, a motion prototype is assumed as another multivariate time series $\mathbf{Q} \in \mathbb{R}^{\tilde{d} \times T}$ where $\tilde{d} \leq d$, which still provides the above purposes. However, the structure of $\mathbf{Q}$ may share only partial similarity to $\mathbf{X}$ based on the temporal or the joints axis (C. Ji et al. 2019; Yeh, Kavantzas, and E. Keogh 2017; Hosseini, Montagne, and Hammer 2019). More specifically, the prototype's length T can be relatively small compared to the input motions' average length. A semantically interpretable (or meaningful) prototype $\mathbf{Q}$ may carry information about a particular motion class or signify a semantic movement related to a specific body joints group. Such information helps us semantically characterize other motion samples based on their (partial-)similarity to $\mathbf{Q}$.

2.3 MEASURING MOTIONS SIMILARITY BY DTW

A principal part of many machine learning algorithms is calculating the similarity of two data points $\vec{x}$ and $\vec{y}$. In a Euclidean space, the similarity between the two vectors $(\vec{x}, \vec{y})$ can be measured by calculating their pairwise distance $\|\vec{x} - \vec{y}\|_2$, where $\|.\|_2$ denotes the l_2-norm (Bellet, Habrard, and Sebban 2013). Specifically for motion data, distance-based methods are often used for the initial analysis or motion retrieval tasks (Vieira et al. 2012; Sedmidubsky and Valcik 2013; Demuth et al. 2006). However, using the above Euclidean distance for real-world captured motion is not practical nor always possible (Ratanamahatana and E. Keogh 2004). As a common observation, two motion samples $(\mathbf{X}_i, \mathbf{X}_j)$ may semantically belong to the same type of movement; however, due to temporal shifts and the difference in the movement's frequency, the value of $\|\mathbf{X}_i - \mathbf{X}_j\|_2$ can be relatively large. For example, a direct comparison of a *walking* cycle between a child and a senior person can make a considerable difference. Talking about the possible frequency difference between $\mathbf{X}_i$ and $\mathbf{X}_j$ in the temporal axis, we confront another typical real-world motion data issue: the difference in the temporal length of movement sequences even for semantically similar samples. Such a difference makes the direct application of Euclidean distance more difficult and less practical.

Due to the Euclidean distance's lack of flexibility and robustness in comparing real-world sequential data, time-series alignment techniques have been proposed for this type of information. These techniques include several distance measure algorithms such as complexity-invariant distance (CID), invariant version of Euclidean distance, longest common sequence distance (LCSS), time warp with edit distance (TWE), dynamic time warping (DTW), and edit distance with the real penalty (ERP) (Batista et al. 2014; E. Keogh, Wei, et al. 2009; Bergroth, Hakonen, and Raita 2000; Marteau 2008; Berndt and Clifford 1994; L. Chen and R. Ng 2004). These algorithms focus on a more intuitive comparison between two given time-series $(\mathbf{X}_i, \mathbf{X}_j)$, and they mostly differ either in the application domain or the specific inconsistency they address between $\mathbf{X}_i$ and $\mathbf{X}_j$. Nevertheless, the most widely used technique among those methods is the DTW algorithm (Berndt and Clifford 1994), which has shown a relatively high degree of robustness and invariance against typical temporal distortions in real sequential data (Lines and Bagnall 2014). In comparison, DTW provides a flexible alignment between two time-series based on a non-linear matching of their time steps. Due to its elastic one-to-many points alignment, DTW can successfully cope with temporal deformations and frequency differences associated with real sequential data (F. Zhou and De la Torre Frade 2012; Adistambha, Ritz, and Burnett 2008; Petitjean, Forestier, Geoffrey I. Webb, et al. 2014).

The DTW algorithm aligns two time series of possibly different lengths according to warping paths such that the aligned points match as much as possible, respecting the temporal ordering of the sequence entries. Dynamic programming enables efficient computation of an optimum match in quadratic time with respect to sequence lengths. Analogous to what described in (E. Keogh and Ratanamahatana 2005), we have

Definition 2.1 (Dynamic Time Warping)**.** DTW defines a warping path W for the two time-series X and Y as a sequence of L indices $(w_1, \cdots, w_L) \in (\{1, \cdots, M\} \times \{1, \cdots, N\})^*$, where

$$w_1 = (1,1),\ w_L = (M, N)$$
$$w_{l+1} - w_l \in \{(1,0), (0,1), (1,1) \text{ for all } l < L\}.$$

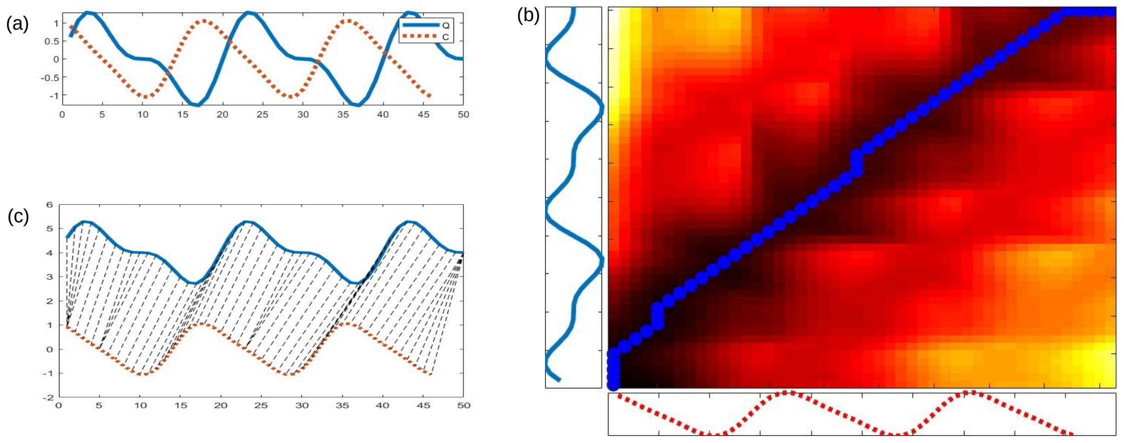


Figure 2.2: (a)) Two time-series Q and C are similar in shape but different in the frequency and the phase. (b) DTW finds the optimal warping path to align Q and C sequences. (c) Q and C are non-linearly aligned by the DTW technique.

Given a warping path W, its warping cost is computed as

$$d_W(X,Y) = \sum_{l=1}^{L} d(X(w_l), Y(w_l))$$

where $d(X(w_l), Y(w_l))$ is the squared Euclidean distance between $X(w_l)$ and $Y(w_l)$. Given the above, the DTW cost is defined with respect to an optimum warping path W:

$$D_{DTW}(X,Y) = \min_W d_W(X,Y).$$

Therefore, DTW finds a warping path to align the entries of the given two time-series. Based on its definition, DTW can have a non-linear warping path, which considers the time frequencies in the time-series. An efficient way to compute the warping path is using the Bellman equation:

$$\begin{aligned} d_W(X[1:i], Y[1:j]) = \;& d(X(i), Y(j)) + \\ & \min\{d_W(X[1:i-1], Y[1:j-1]), \\ & \quad d_W(X[1:i-1], Y[1:j]), \\ & \quad d_W(X[1:i], Y[1:j-1])\}, \end{aligned} \tag{2.3}$$

which is a part of the dynamic programming that computes DTW in a quadratic time. However, several ways are suggested to speed up the computations in practice as discussed in (Al-Naymat, Chawla, and Taheri 2012). Figure 2.2 shows two example time-series of different lengths and their alignment path using the DTW technique.

As an interesting characteristic, DTW can provide a dissimilarity value of $\mathcal{D}_{\text{DTW}}(X^i, X^j)$ for any given two time series X^i and X^j of possibly different lengths. This feature makes DTW a practical technique to compute the similarity/dissimilarity of motion data, whereas a common observation, even real-world motions of the same type have different lengths. It is important to note that DTW is not a metric since the triangle inequality does not hold; rather, it is a pairwise symmetric distance function, which can serve as a data dissimilarity measure. Nevertheless, we refer to DTW as a metric in some places for the sake of simplicity, although strong metric properties do not apply.

There are several variants and extensions of this elastic alignment techniques such as derivative dynamic time warping (E. J. Keogh and Pazzani 2001), multi-dimensional DTW (Shokoohi-Yekta et al. 2015), DTW under amplitude offset and scaling (T.-W. Chen, Abdelmaseeh, and Stashuk 2015), applying a limitation on the warping path length (Zheng Zhang et al. 2017), and considering a global transformation of the time-series in the DTW algorithm (J. Zhang and X. Jiang 2020). As another important fact to consider, although I have chosen the vanilla DTW technique (E. Keogh and Ratanamahatana 2005) as a flexible and intuitive distance measure for analyzing motion data, it is possible to replace it with any other of its variants or also other desirable distance measures for all the proposed algorithms in this work. The purpose of this work is to take advantage of any proper alignment technique to perform interpretable analysis on motion data from the perspectives described in Chapter 1.

2.4 BENCHMARK MOTION DATASETS

In order to empirically evaluate the proposed motion analysis algorithms in this work, I use the following real-world motion datasets.

CMU Mocap : Carnegie Mellon University's human motion dataset (CMU. 2007) is an extensive collection of different human activities (Figure 2.3-a). The data is collected from 144 subjects, each performing a sequence of different human actions per recording session. The dataset is captured by Vicon infra-red cameras (Merriaux et al. 2017) using 41 markers (Figure 2.3-b). However, the Euler angle representation of the data provided in (CMU. 2007) results in a 62-dimensional time series for each motion. I use the following four subsets of the CMU Mocap dataset:

Walking: This dataset is collected from 7 different walking styles (*normal*, *fast*, *slow*, *turn right*, *turn left*, *veer right*, and *veer left*) carried out by 4 different subjects. This dataset consists of 49 samples (7 samples per class). This dataset is used for empirical evaluations of Chapter 3.

Dance: It contains 35 data samples related to two different dance styles, *Modern* and *Indian*, collected from subjects 5 and 94 of the CMU Mocap dataset. This dataset is used in experiments of Chapters 3 and 4. I use the Dance dataset for the experiment section of Chapters 3 and 4.

CMU Mocap 9: I combined the movement data of subject 86 from the dataset, which is a combination of 9 different types of human movements such as *walking*, *running*, *clapping*. Then, the data is segmented in order to break down the long movements into smaller segments as single periods of each type of motion. Consequently, we obtain 9 classes of data with 10 samples per class. This dataset is employed for empirical evaluations of Chapters 4 and 5.

CMU Mocap Segment: In this subset of the CMU Mocap dataset, our task is to segment and recognize each recorded session's performed actions. We collect the sessions which contain actions from 15 highly observed categories in the whole dataset. These categories include movements such as *walk*, *punch*, *wave*, *run*, *jump*, and *raise*. For the segmentation task, we treat the rest of the action classes as blank spaces (gaps). I use this dataset in experiments of Chapter 6.

Cricket Umpire's Signals: Cricket Umpire's Signals is the dataset of different arm movements representing the cricket umpire's signals (Shepherd 2005). For example, the event *No-ball* is signaled by holding one arm out at shoulder height to indicate that the ball is delivered while also signifying the player's fault (Fig. 2.4). I employ the Cricket dataset of (Ko et al. 2005) for my experiments, which is captured via accelerometers on the umpire's wrists while performing the signals (Chambers et al. 2004). The dataset has 12 classes and 180 samples, while each sample is a 6-dimensional time-series (related to X, Y, and Z coordinates of both hands). This dataset is employed in empirical evaluations in Chapters 3, 4, and 5.

Articulatory Words: The Articulatory Words dataset contains the recorded movement of different facial parts while the person utters 25 different English words in individual trials. The dataset is captured by (Jun Wang et al. 2013) via attaching 12 Electromagnetic Articulograph (EMA) sensors to the person's different parts of the forehead, lips, and tongues, resulting in 36 features. I use the 9-dimensional subset of this dataset (Shokoohi-Yekta et al. 2015), which contains 575 samples. Each sample is a 3D spatial data (X, Y, and Z) related to the tip of the tongue (T1), upper lip (UL), and lower lip (LL), which shapes 9 features in total. I use this dataset to empirically analyzed the methods proposed in Chapters 3, 4, and 5.

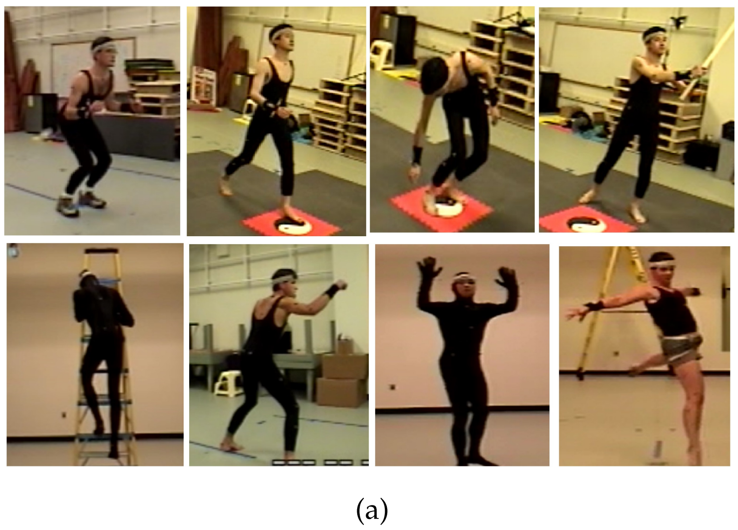
Squat dataset: The Squat dataset is collected as a part of the large-scale intelligent coaching project (Waltemate et al. 2015). The data is a set of squat movements performed by three coaches while captured by the optical mocap system. Each squat is segmented into three movement primitives *preparation*, *going down*, and *coming up*, producing 87 samples of data and 9 class labels (by also distinguishing the coaches). The Squat dataset is used in some of the experiments in Chapter 5.

UTKinect Actions : This dataset consists of 3D locations of body joints recorded using Kinect device related to 10 different actions (L. Xia, C.-C. Chen, and J. Aggarwal 2012). The motion classes include *walk*, *push*, *pick up*, *stand up*, *throw*, *wave*, *pull*, and *clap hands*. The dataset is collected from 20 subjects and contains 199 action instances in total, while all the motions are recorded in pre-segmented settings. I use this dataset in the experiment sections of Chapters 4 and 5.

HDM05 Mocap : This dataset is recorded using an optimal marker-based mocap system with a 120 Hz sample rate (M. Müller, Röder, et al. 2007). It consists of 130 actions related to the performance of 5 actors (non-professional). The data format is 3D coordinates of 31 body joints. As suggested by (Kyunghyun Cho and X. Chen 2014), the classification task is defined by putting some of the semantically similar action classes into one category resulting in a total of 65 motion classes. I include this dataset in the experiment sections of Chapters 4 and 5.

Montalbano V2 dataset This dataset is related to the "ChaLearn Looking at People" challenge, which is recorded with Kinect technology as described by (Escalera et al. 2014). It includes 13,858 samples of 20 different Italian sign gestures, which are recorded in continuous sequences. This challenge aims to perform action recognition and segmentation for sign gestures based on the given test and training streams. This dataset is employed in the experiment section of Chapter 6.

SYSU-3D Human-Object Interaction dataset (SYSU) : The SYSU dataset contains 12 different action classes recorded in 480 video sequences (J.-F. Hu, W.-S. Zheng, Lai, et al. 2015). SYSU dataset is captured from 40 human subjects. . In each time-frame, it represents the 3D coordinates of 20 body joints. This dataset is included in the empirical evaluations of Chapter 6.



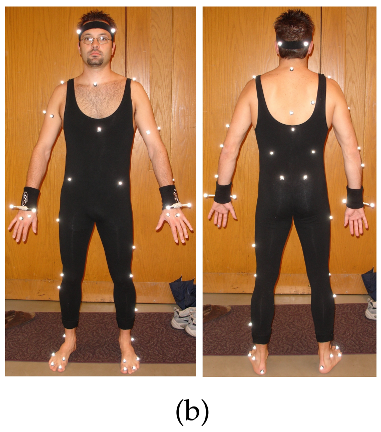


Figure 2.3: (a) Examples of different human activities recorded in the CMU Mocap dataset. (b) The marker locations on the body of the subject to track movements of different joints during motion activities. Images are taken from CMU mocap website[1].

NTU-RGB+D Dataset (NTU) : This action recognition dataset consists of 60 classes of actions captured from 40 human subjects (Shahroudy et al. 2016). It has 56,000 sequences with 4 million frames in total, and the recorded data of 25 main body joints are used for the action recognition task. There are two typically used evaluation protocols for this benchmark: The Cross-Subject (CS) recognition task, which uses data of 20 subjects for training and the rest for testing, and the Cross-View (CV) task in which the recorded samples from camera 2 and 3 constitute the training set and the rest is preserved for the test set. I use this dataset for experiments in Chapter 6.

I also employed the following additional datasets for some of my evaluations, which are multi-dimensional time-series, but they are not skeleton-based motions:

DynTex++: The DynTex++ benchmark is a large-scale Dynamic Texture dataset, consisting of recorded videos of natural objects' movements in different environments (Ghanem and Ahuja 2010). The dataset includes 36 different classes, such as *boiling water*, *fire*, *flowers*, *fountains*, *plants*, *sea*, and *smoke*. Each sample data has a size of $50 \times 50 \times 50$, and each category contains 100 sequences. To convert it to a multi-dimensional times series, each video frame is mapped to a vector of size 3 using the descriptors used in (Ghanem and Ahuja 2010). The experiments of Chapter 4 also include implementations on DynTex++ .

Schunk Dexterous : The Schunk Dexterous dataset is the recorded tactile sensors of a robot's hand during grasping 10 different objects (Drimus et al. 2014). The objects have both rigid and deformable types, such as *rubber ball*, *duck*, *wood block*, and *tape*. The dataset contains 10 samples per object, while each sample is an 8×8 pressure grid resulting in a 64-dimensional time-series. This dataset is employed in the experiment section of Chapters 4 and 5.

1 http://mocap.cs.cmu.edu/info.php
2 https://parkhillcricket.com/2014/04/11/cricket-umpiring-how-to-umpire-knowing-the-basics/

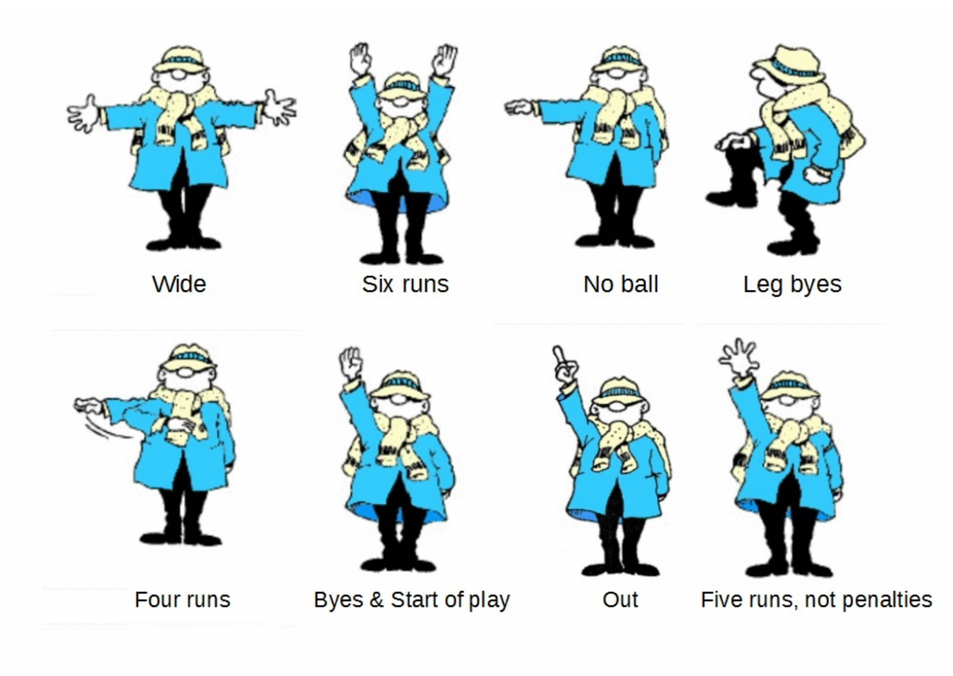


Figure 2.4: 8 sample classes from the Cricket dataset. Photo is created using the information from Park Hill cricket club website[2].

2.5 MOTION DATA ANALYSIS LITERATURE

Due to the discussed relevance of motion data in different application domains, quite a few approaches have been proposed in the literature focusing on time series analysis for motion data from different perspectives. These approaches range from supervised classification tasks, segmentation techniques, and motion prediction to motion retrieval and further research areas.

For supervised classification of motion data, the objective is to determine the label vectors $\vec{h}_i$ for the test motion data $\vec{z}_i$ given that the class label matrix $\mathbf{H}$ for the training set $\mathbf{X}$ is available. The main technologies for motion data classification include signal processing-based methods, DTW-based classifiers, and the family of deep neural networks. As a common practice in the signal processing domain, multiple features are extracted from the motion or activity time-series. These features are typically extracted using time-domain characteristics of the time-series signal (Dernbach et al. 2012; Martín et al. 2013; Morales, Akopian, and Agaian 2014; J. Guo et al. 2016) or its frequency domain transforms (Anguita et al. 2012; Z. He 2010; H. Xu et al. 2016). Hence, the right choice of extracted features highly affects the performance of the classifier.

The DTW-based classifiers are generally constructed upon using the DTW technique to compare different motion sequences and measure their similarity or difference. Based on such relational input, different classifiers and algorithms can be employed to determine the motion labels (Z. Zeng, Amin, and Shan 2020; Lun and W. Zhao 2015; Ahmed, Paul, and Gavrilova 2015; Hosseini and Hammer 2015). The success of output these methods relies on the choice of the next-level classifier and the pre-processing steps for the effective application of the DTW technique.

The motion classification algorithms that are designed based on deep neural networks are mainly constructed from convolutional neural networks (Sijie Yan, Xiong, and D. Lin 2018; Núñez et al. 2018; Baoding Zhou, Jun Yang, and Q. Li 2019), long short-term memory architectures (C. Li et al. 2017; J. Liu, Shahroudy, et al. 2018; Saha, Sandha, and M. Srivastava 2020), or recurrent neural models (Y. Du, Wei Wang, and Liang Wang 2015; H. Wang and Liang Wang 2017; Uddin et al. 2020). The benefit of these methods is their automated feature extraction units and their relatively high accuracy compared to other mentioned approaches.

Another group of approaches related to the analysis of motion data focuses on motion segmentation. The objective of these methods is to split a long motion into a sequence of smaller motions, which are semantically meaningful to the domain experts. For instance, a long series of human activities can be segmented into a sequence of walking, jumping, kicking, waving, and other activities. To that aim, each time-step of the sequence is labeled as one specific motion class. Significant motion segmentation works include unsupervised methods such as (F. Zhou, De la Torre, and Hodgins 2008; Qifei Wang et al. 2015; B. Krüger et al. 2017; Lichen Wang, Z. Ding, and Fu 2018) and supervised approaches like (Gehring et al. 2017; X. Ma and Hovy 2016; Alzaidy, Caragea, and Giles 2019; Z. Yang, Salakhutdinov, and Cohen 2016). Generally, the supervised methods have better segmentation accuracy because they benefit from label information in the training phase. On the other hand, unsupervised methods do not need any annotation phase to prepare a training set of motions.

Motion prediction aims to recognize actions before their full execution in the temporal axis. More specifically, motion prediction approaches focus on predicting the next step(s) in a given motion sequence. With that purpose, these approaches often overlap with sequence or trajectory prediction methods, which are applicable to multivariate time-series (Letham, Rudin, and Madigan 2013; Koppula and Saxena 2015; Y. Wang et al. 2017). A group of these methods relies on dynamical modeling of the motion sequence (Basharat and Shah 2009; Kratzer, Toussaint, and Mainprice 2018), while another important group of motion prediction approaches is constructed upon deep learning architectures (Hua et al. 2019; Afrasiabi, Mansoorizadeh, et al. 2019; Butepage et al. 2017).

Another stream of work focuses on motion retrieval as finding semantically related motion sequences in a large dataset. This process is sometimes perceived as another form of motion classification. A number of these approaches benefits from the DTW technique to pair and match similar sequences (M. Müller and Röder 2006; Kovar, Gleicher, and Pighin 2008; E. Keogh, Palpanas, et al. 2004), while some other methods use different techniques such as graph matching, distance metric learning, dictionary-based models, and gesture description models (F. Zhou, De la Torre, and Hodgins 2012; Cheng Chen et al. 2010; Q. Xiao and R. Song 2017; Hachaj and Ogiela 2014). Additionally, some successful motion retrieval works combined previous ideas with deep learning models (Q. Xiao and C. Chu 2017; Coskun et al. 2018).

Many of these approaches for motion data analysis are black-box ones, and they do not provide interpretable models or insight into how to represent motion data such that it aligns with semantic meaning. In contrast, the focus of this thesis is on approaches that align with the semantic meaning of motions, which enable some form of interpretation of why a specific decision is made about a given action. For this purpose, I will look at different objectives. More specifically, methodologies will follow the avenue of metric learning, sparse coding, multiple kernel learning, and deep learning, as summarized in Table 2.1.

Table 2.1: Summary of the proposed approaches in this thesis and their functionalities, features, and objectives.

Section	Method	I/O functionality	Features	Mathematical Objectives
3.2	Large margin metric learning based of DTW alignment abbrev.: DTW-LMNN.	**Input**: DTW-distance matrix for labeled Mocap data of arbitrary length. **Output:** • Learned metric that improves *k*NN classifier's performance. • Relevance profile for the body joints.	Bringing semantically similar motion data closer in local neighborhoods by focusing on DTW-distance of relevant body joints.	Large margin nearest neighbor error for a component-wise weighted DTW distance.
3.3	Feasibility based metric learning based on DTW alignment abbrev.: FTW-LMNN.	**Input**: Same as for DTW-LMNN. **Output**: Improved DTW-LMNN's metric by ruling out weakly-feasible target neighbors from optimization.	A measure for validity of triplets by means of their geometry.	DTW-LMNN objective with feasibility weights of triplets.
4.2	Non-negative Kernel-based sparse encoding abbrev.: NNKSC.	**Input:** Similarity Kernel of motion dataset. **Output:** Encoding of motion sequences into sparse non-negative vectors.	• Unsupervised. • Dictionary-based encoding. • Interpretable encoding and dictionary model due to the non-negative term.	• Dictionary-based reconstruction error of input motion in the feature space. • Non-negative constraints on dictionary and sparse codes. • Sparsity constraint.
4.2	Label-consistent Non-negative Kernel-based sparse encoding abbrev.: LC-NNKSC.	**Input:** Similarity Kernel of motion dataset (labeled). **Output:** • Non-negative sparse encoding of input motions. • A linear transform from sparse codes to label space.	• Supervised extension of NNKSC. • Encoding supervised information.	• NNKSC's objective and constraints. • Linear discriminative objective term.

Section	Method	I/O functionality	Features	Mathematical Objectives
4.3	Confidence-based Kernel Sparse Coding abbrev.: CKSC	**Input:** Similarity Kernel of motion dataset (labeled). **Output:** • Non-negative sparse encoding of input motions. • Robust encoding of supervised information.	• Supervised extension of NNKSC. • Consistent test and training encoding models.	• NNKSC's objective and constraints. • Encode each sequence based on other sequences of the same class. • Robust discriminative objective term.
4.4	Kernel-based Non-negative Local Subspace Sparse Clustering abbrev.: NLKSSC	**Input:** Similarity Kernel of motion dataset. **Output:** Self-representative encoding of motion data into sparse vectors.	Unsupervised encoding of motion data that reveals its underlying subspaces.	• Self-representative reconstruction error in feature space. • Unsupervised local separation of data neighborhoods. • Non-negativity constraint. • Low-rank objective.
5.2	Large Margin Multiple-kernel Learning abbrev.: LMMK	**Input:** Component-wise multiple-kernel representation of motion dataset (labeled). **Output:** • A diagonal metric that improves local separation of classes in RKHS. • Relevant joints (base kernels) to the *k*NN classifier's performance.	• Discriminative feature selection for motion data using multi-kernel input. • Supervised. • Sparse scaling of feature space.	• Non-negative scaling of feature space (kernel combination). • Large margin nearest neighbor error in the feature space.
5.3	Interpretable Multiple-Kernel Prototype Learning abbrev.: IMKPL	**Input:** Component-wise multiple-kernel representation of motion dataset (labeled). **Output:** • Non-negative class-specific prototypes. • Relevant joints to the prototype-based representation.	• Prototypes represent and discriminate their nearby neighborhoods. • Supervised. • Scaling of feature space.	• Non-negativity constraints. • Multiple-kernel extension of prototype learning. • Local separation of data in the combined RKHS. • Prototype interpretability objective.

Section	Method	I/O functionality	Features	Mathematical Objectives
5.4	Multiple-Kernel Dictionary Structure abbrev.: MKD	**Input:** Component-wise multiple-kernel representation of motion dataset. **Output:** • Partial connection between unseen motion and the training sequences. • Recognition and categorization of unseen motions.	• Unsupervised. • Semantic motion attributes linked to individual sets of body joints.	• Non-negativity constraints. • Incremental hierarchical clustering of unseen motion. • Partial encoding of unseen motion.
6.3	Deep-Aligned Convolutional Neural Network abbrev.: DACNN	**Input:** Raw unsegmented motion sequences of arbitrary length. **Output:** Segmented and classified motion subsequences.	• Sequence labeling of motion data and revealing significant patterns in the motion sequence. • Supervised.	• Alignment based computation of the first feature map. • Classification loss + sparsity loss.

3 METRIC LEARNING FOR MOTION ANALYSIS

Publications: This chapter is partially based on the following publications.

- Hosseini, Babak and Barbara Hammer (2015). "Efficient metric learning for the analysis of motion data". In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.
- — (2018b). "Feasibility Based Large Margin Nearest Neighbor Metric Learning". In: *26th European Symposium on Artificial Neural Networks (ESANN)*.

Metric learning constitutes a matured field of research for the standard vectorial setting (data represented by feature vectors) (Bellet, Habrard, and Sebban 2013; Kulis 2012; Schneider, Biehl, and Hammer 2009; Kilian Q. Weinberger and Lawrence K. Saul 2009). In these approaches, usually, quadratic forms are inferred from the given auxiliary information. This vectorial metric adaptation does not only provide increased model accuracy, but it also dramatically facilitates model interpretability, and it can lead to additional functionalities such as a direct data visualization (Biehl, Bunte, and Schneider 2013; Backhaus and Seiffert 2014; Bunte et al. 2012). As another interesting property of metric learning methods, they generally transform the data distribution such that each data point is semantically similar to its closest neighboring points in the space. Such property makes the resulting distribution more interpretable in terms of the semantic connection between nearby data points.

Studying the content of the learned metric, some researchers have focused on the validity of the interpretation of its parameters as a profile of relevance weights (Arlt et al. 2011). However, as pointed out in (Strickert et al. 2013), However, as pointed out in X, considerable redundancy exists in the derived relevance profile for high-dimensional or highly correlated data like human motions. Accordingly, it is possible to avoid these problems by applying an efficient form of metric regularization as detailed in the approaches (Frénay et al. 2014; Strickert et al. 2013). Thus, the obtained relevance profiles reveal significantly relevant data dimensions to the given classification problem.

Currently, the mentioned developments regarding metric learning methods mainly focus on the context of vectorial data. Therefore, they do not apply to distance-based measures, e.g., in the case of using DTW. A few approaches have recently been proposed which address metric-parameter learning for complex non-vectorial data, in particular sequences and sequence alignment (Bernard et al. 2008; Bellet, Habrard, and Sebban 2013; Mokbel et al. 2015). While these approaches lead to increased model accuracy and interpretability, they have the drawback that their training complexity is very costly: typically, these techniques adapt metric parameters within sequence alignment, such that pairwise distances of all data samples have to be recomputed after every metric adaptation step. This limitation leads us to this follow-up of the research question **RQ1**:

RQ1-a: How can we apply the metric learning framework to the distance-based representation of motion data?

Also, another relevant objective is to investigate whether the regularization of such learned metric results in an interpretable relevance profile for body joints.

Another challenge that I address is in particular related to the large margin nearest neighbors (LMNN) metric learning method, but it can be extended to other similar metric learning algorithms. Given a local neighborhood of data points, LMNN focuses on bringing semantically similar data (targets) closer while pushing away semantically distinct samples (Kilian Q. Weinberger and Lawrence K. Saul 2009). Therefore, a significant step of the LMNN algorithm is the proper selection of neighboring targets in its optimization framework. For the original LMNN method (Kilian Q. Weinberger and Lawrence K. Saul 2009), this step is performed by the same-class nearest neighbor's strategy. However, it is possible to show that the wrong choice of targets can severely shrink the size of the possible solution set for the metric parameters (Hosseini and Hammer 2015). Accordingly, the next arising follow-up question for **RQ1** is:

RQ1-b: How can we change the optimization framework of LMNN for a better selection of its target data points?

As a contribution, I employ the component-wise DTW-based dissimilarity representation of motion data. This strategy is similar to the popular treatment of dissimilarity data as *features*, which is detailed in the monograph (Pekalska and Duin 2005). I extend the application of the powerful LMNN metric learner to a metric adaptation for DTW, which adjusts the relevance of single joints and their correlations in the Mocap data according to a given specific classification task. Accordingly, I have the following contributions with respect to the state-of-the-art of metric learning and its distance-based extension.

- My distance-based extension to the powerful LMNN metric learning method (DTW-LMNN) enables us to apply this algorithm to any dissimilarity-based description of data, such as the DTW-based representation of motion data.
- I show the possibility of transferring auxiliary concepts such as metric regularization for motion data, based on the learned distance-based metric by the DTW-LMNN algorithm.
- I introduce a feasibility measure to quantify the size of the feasible solutions set for the selected target points in the LMNN optimization framework. Following this measure, I propose the effective FDW-LMNN framework, in which the target's relevance is treated according to the above measure.

In the next section, I review the formulation of the LMNN algorithm and the metric regularization. Then, the proposed distance-based metric learning algorithm and its feasibility-based improvement are explained in its following sections. After that, the regularization of the metric in the distance space is explained, which is followed by experiments on the mocap benchmarks and the appropriate conclusion.

3.1 STATE OF THE ART

In this section, I review the LMNN algorithm and discuss the way to regularize a metric transform matrix to diminish random effects caused by data correlations.

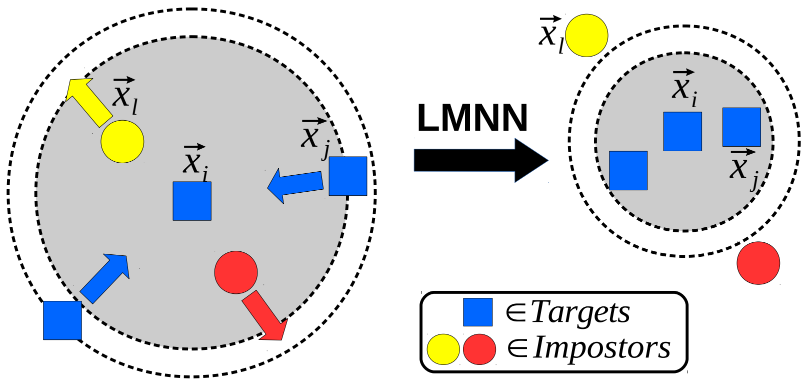


Figure 3.1: **Left:** The original distribution of the neighboring points around $\vec{x}_i$, where the rectangles (*targets*) have the same label as that of $\vec{x}_i$, and circles (*impostors*) have different labels. **Right:** The distribution resulted from LMNN's mapping, in which *targets* are closer to $\vec{x}_i$ while *impostors* are pushed farther away from it.

Large Margin Nearest Neighbors Metric Learning

LMNN is a metric learning algorithm that learns a quadratic form from given labeled data $(\vec{x}_i, \vec{h}_i) \in \mathbb{R}^d \times \mathbb{R}^C$, where c denotes the number of classes, to improve the classification accuracy of the well-known k-nearest neighbors (kNN) method. As a distance-based approach, the accuracy of kNN fundamentally relies on its underlying distance measure, which determines the k nearest neighbors of a given data point. LMNN tries to adjust this neighborhood structure robustly by learning a parameterized form

$$\mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) = (\mathbf{L}(\vec{x}_i - \vec{x}_j))^2 = (\vec{x}_i - \vec{x}_j)^\top \mathbf{L}^\top \mathbf{L}(\vec{x}_i - \vec{x}_j) \tag{3.1}$$

with adjustable linear transformation matrix $\mathbf{L} \in \mathbb{R}^{d \times d}$ which induces a quadratic form characterized by $\mathbf{M} := \mathbf{L}^\top \mathbf{L}$.

The objective function of LMNN is based on a fixed k-neighborhood structure. Based on the intuition of having dense same class neighborhoods (*targets*), while maximizing distances of a data point to its neighbors with different labeling (*impostors*), the costs of LMNN become

$$\begin{aligned} \epsilon(\mathbf{L}) := \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} \mathcal{D}_{\mathrm{L}}(\vec{x}_i, \vec{x}_j) \\ & + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l \in \mathcal{I}_i^k} \left[1 + \mathcal{D}_{\mathrm{L}}(\vec{x}_i, \vec{x}_j) - \mathcal{D}_{\mathrm{L}}(\vec{x}_i, \vec{x}_l)\right]_+ \end{aligned} \tag{3.2}$$

where $[\cdot]_+$ refers to the Hinge loss, and the meta parameter $\mu \in [0\ 1]$ makes a trade-off between the pulling (first) and pushing (second) parts of the objective. The sets $\mathcal{N}_i^k$ and $\mathcal{I}_i^k$ contain the indices of the k-nearest *targets* and *impostors* of $\vec{x}^i$, respectively.

This objective can be interpreted as the goal to adjust the metric $\mathbf{L}$ such that all points with different class labels are located outside of the data neighborhood with a fixed margin (Figure 3.1). It has been shown in (Kilian Q. Weinberger and Lawrence K. Saul 2009) that this optimization problem is equivalent to the following semi-definite optimization:

$$\begin{aligned} \min_{\mathbf{M}} \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} \mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l \in \mathcal{I}_i^k} \xi_{ijl} \\ \text{s.t.} \quad & \mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_l) - \mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \ \ \mathbf{M} \succeq 0, \ \ \forall i, j \in \mathcal{N}_i^k, l \in \mathcal{I}_i^k. \end{aligned} \tag{3.3}$$

Each positive slack variable ξ_{ijl} is related to a triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$, in which $\vec{x}_j$ and $\vec{x}_l$ are respectively a *target* for $\vec{x}_i$ and its *impostor* located between $\vec{x}_i$ and $\vec{x}_j$ (similar to Figure 3.1-left). Hence, the scalars ξ_{ijl} model the costs induced by the existing impostors.

The problem in (3.3) can be optimized efficiently w.r.t. the matrix **M**. Note that it is possible to choose a low-rank matrix **M** which corresponds to a low-dimensional projection $\mathbf{L} \in \mathbb{R}^{\tilde{d}\times d}$ for data vectors, where $\tilde{d} < d$ is the dimension of the data in the lower-dimensional space. As described in (Kilian Q. Weinberger and Lawrence K. Saul 2009), one can minimize Equation 3.3 with respect to an **L** matrix while constraining **L** to have a rectangular form by choosing $\tilde{d} \ll d$. Although this modification results in a non-convex optimization problem in terms of **L**, as discussed by (Torresani and K.-c. Lee 2007), this issue would not lead to poor-quality local minima.

Equation 3.3 constitutes a convex problem with respect to **M** if the targets $\mathcal{N}_i^k$ and impostors $\mathcal{I}_i^k$ are fixed (Kilian Q. Weinberger and Lawrence K. Saul 2009). Nevertheless, different selections for these initial targets can lead to different solution **M**. As suggested in (Kilian Q. Weinberger and Lawrence K. Saul 2009; Göpfert, Paassen, and Hammer 2016), a better strategy is to repeat LMNN's optimization multiple times (multiple-pass LMNN) while updating $\mathcal{N}_i^k$ and $\mathcal{I}_i^k$ in each run based on the resulting quadratic form **M**. Yet, also this strategy relies on the quality of the initial selection of these two sets (Hosseini and Hammer 2018b). To mitigate this problem, I propose a modification to the optimization scheme of Equation 3.3 in Section 3.3. The new formulation focuses on selecting more promising targets in $\mathcal{N}_i^k$ and eliminating the less achievable targets of $\mathcal{N}_i^k$ w.r.t. a linear transform $\mathbf{L}\vec{x}$.

Metric Regularization

The adaptation of a quadratic form as present in LMNN does not only enhance the classification accuracy, but it can also give rise to increased interpretability of the results. A quadratic form corresponds to the linear data transformation $\vec{x}_i \mapsto \mathbf{L}\vec{x}_i$. Hence the diagonal terms of the matrix **M**

$$M_{kk} = \sum_i L_{ik}^2 \tag{3.4}$$

summarize the influence of feature k on the mapping. Due to this observation, metric learners are often accompanied by the *relevance profile*, which is obtained from the diagonal entries of **M**; this gives insight into relevant features for the given task, such as potential biomarkers for medical diagnostics (Arlt et al. 2011).

It has recently been pointed out that this interpretation has problems provided high-dimensional or highly correlated data are analyzed: in such cases, the relevance profile and the underlying linear transformation **L** are not unique, rather data correlations can give rise to random, spurious relevance peaks. I expect this effect for Mocap data due to a high correlation of neighboring joints. For vectorial data, this effect is caused by the following observation, as pointed out in (Strickert et al. 2013): assume $\mathbf{X} = [\vec{x}_1, \ldots, \vec{x}_N]$ refers to the data matrix. Then two linear transformations $\mathbf{L}_1$ and $\mathbf{L}_2$ are equivalent with respect to **X** iff $\mathbf{L}_1\mathbf{X} = \mathbf{L}_2\mathbf{X}$. This relationship is equivalent to the fact that the difference $(\mathbf{L}_1 - \mathbf{L}_2)\mathbf{X}$ vanishes. Hence, by considering the squared form

$$(\mathbf{L}_1 - \mathbf{L}_2)\mathbf{X}\mathbf{X}^\top(\mathbf{L}_1 - \mathbf{L}_2)^\top = 0, \tag{3.5}$$

we can relate this property to the fact that the rows' differences are given by vectors that lie in the null space of the data correlation matrix $\mathbf{C} := \mathbf{X}\mathbf{X}^\top$. This fact gives us a unique characterization of the equivalence class of matrix **L** with respect to the data transformations for **X**: equivalent matrices, e.g., matrices which map data **X** in the same way as matrix **L**, differ from **L** by multiples of eigenvectors related to 0 eigenvalues of **C**. Provided the metric learning method does not take this fact into account, its outcome matrix is random as concerns contributions of this null space.

For LMNN, costs are invariant to null space contributions, e.g., the matrix **L** is random in this respect. Albeit this property does not affect the training data **X**, it influences the result in two aspects: for test data, the null space is usually different, e.g., the generalization ability of the model is affected by random effects of the training data correlation and the initialization point **L** for the optimization problem. Second, more severely, random contributions of the null space of **C** change the relevance profile M_{kk} and can give rise to spurious effects such as high values that are not supported by any *real* relevance of the feature k.

Therefore, it is advisable to regularise the matrix **L** by relying on the representative of the equivalence class of **L** with the smallest Frobenius norm. Equivalently we can consider a projection of **L** to the space of eigenvectors of **C** with non-vanishing eigenvalues, or more precisely, the unique transformation

$$\begin{aligned}&\tilde{\mathbf{L}} := \mathbf{L}\Phi \\ &\text{where } \Phi := \textstyle\sum_{s=1}^{S} \vec{u}_s(\vec{u}_s)^\top \text{ with the eigenvectors} \\ &\vec{u}_1,\ldots,\vec{u}_S \text{ of } \mathbf{C} \text{ with nonvanishing eigenvalues.}\end{aligned} \tag{3.6}$$

For vectorial data, the same effect can be obtained by deleting the null space from the data vectors in the first place employing the principal component analysis (PCA), as a prevalent preprocessing approach. However, we show in Section 3.4 that this reformulation as matrix regularization is beneficial to more general data such as the alignment vectors, where the direct application of PCA is not applicable or is against the resulting metric transform **L**.

Metric Adaptation for Dynamic Time Warping

The DTW technique was initially used for the alignment of 1-dimensional sequences (Berndt and Clifford 1994). However, as addressed in (Shokoohi-Yekta et al. 2015), practitioners more often have to deal with multi-dimensional sequential data, of which motion is a specific example. Accordingly, one way to treat the sequence entries' vectorial nature is to compute DTW on vectorial sequences directly. Then, DTW's outcome is determined by choosing the parameters of the metric used to compare vectorial sequence entries along the warping path. In other words, crucial metric parameters are those involved in computing $\mathcal{D}(\vec{x}_i(t_1), \vec{x}_j(t_2))$, where the warping path determines the time points (t_1, t_2), and $\mathcal{D} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a vectorial metric used to compare the vectorial sequences. As the baseline method, we can simply use the Euclidean distance between two vectors $(\vec{x}_i(t_1), \vec{x}_j(t_2))$ to compute $\mathcal{D}(\vec{x}_i(t_1), \vec{x}_j(t_2))$ as the DTW_D method in (Shokoohi-Yekta et al. 2015).

Related to the above concern, a few approaches have been proposed while focusing on the question of how to learn an optimum transformation **L** provided an alignment (e.g., DTW) is used for vectors (Bernard et al. 2008; Bellet, Habrard, and Sebban 2013; Mokbel et al. 2015). However, these techniques face the problem that metric adaptation can change the form of an optimum warping path, e.g., a computationally costly recalculation of the warping path is necessary to obtain stable results. Therefore, I propose a new approach in the next section, which is based on the dimension-wise computation of DTW for multivariate sequences. I show that such component-wise DTW formulation has the benefit that not only LMNN can efficiently be transferred to a novel dissimilarity-based setting but also other concepts such as metric regularization is applicable to such formulation.

3.2 DISTANCE-BASED METRIC LEARNING

As discussed before, metric learning enables a problem-adapted representation of data, which is interpretable in terms of local data neighborhoods. When applied to motion data, this property increases the resulting representation's usability by making the distance between semantically similar motion sequences smaller. Nevertheless, the majority of metric learning methods have been proposed for vectorial data only. In order to tackle this limitation, I investigate metric learning in the context of dynamic time warping (DTW), the by far most popular dissimilarity measure used for the comparison and analysis of motion capture data. I extend the popular principle offered by the LMNN algorithm via the DTW distance by treating the resulting component-wise dissimilarity values as individual features. Application of such a model to motion data adjusts the relevance of single joints and their correlations in the Mocap data according to the given specific classification task.

Our input motion sequences are multivariate time-series of different lengths. Therefore, we can compute DTW separately for every dimension of a given time series $\vec{x}_i^k = (x_i^k(1) \dots x_i^k(T)) \in \mathbb{R}^*$, where $k \in \{1, \dots, d\}$ refers to the component k of the vectorial sequence entries for motion $\mathbf{X}_i$. For two time series, we thus get a vector of distances

$$\vec{D}^{ij} := (\mathcal{D}_{\mathrm{DTW}}(\vec{x}_i^1, \vec{x}_j^1), \dots, \mathcal{D}_{\mathrm{DTW}}(\vec{x}_i^d, \vec{x}_j^d)) \in \mathbb{R}^d \tag{3.7}$$

of dimensionality d. A real-valued dissimilarity can be computed thereof by a standard quadratic form:

$$\begin{gathered}\mathcal{D}_{\mathrm{DTW-LMNN}}\ (\mathbf{X}_i, \mathbf{X}_j) := (\mathbf{L} \cdot \vec{D}^{ij})^2 = \\ \Big(\mathbf{L} \cdot (\mathcal{D}_{\mathrm{DTW}}(\vec{x}_i^1, \vec{x}_j^1), \dots, \mathcal{D}_{\mathrm{DTW}}(\vec{x}_i^d, \vec{x}_j^d))\Big)^2\end{gathered} \tag{3.8}$$

which is parameterized by a linear mapping $\mathbf{L} : \mathbb{R}^d \to \mathbb{R}^d$ (or a low-dimensional counterpart $\mathbf{L} : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ where $\tilde{d} < d$). In both cases, metric parameters are in the form of a linear transformation $\mathbf{L}$ or corresponding quadratic matrix $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, which have to be adapted according to the given problem for an optimal result.

Therefore, I propose a reformulation of the LMNN approach based on component-wise DTW vectors (Equation 3.8). This formulation has the benefit that not only LMNN can efficiently be transferred to a novel dissimilarity-based setting but also recent concepts for metric regularization apply to such problems. To that aim, for a sequence metric such as $\mathcal{D}_{\mathrm{DTW-LMNN}}$, the LMNN costs (Equation 3.2) become:

$$\begin{aligned}\epsilon(\mathbf{L}) := \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} \mathcal{D}_{\mathrm{DTW-LMNN}}\ (\mathbf{X}_i, \mathbf{X}_j) \\ & + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l \in \mathcal{I}_i^k} \left[1 + \mathcal{D}_{\mathrm{DTW-LMNN}}\ (\mathbf{X}_i, \mathbf{X}_j) - \mathcal{D}_{\mathrm{DTW-LMNN}}\ (\mathbf{X}_i, \mathbf{X}_l)\right]_+\end{aligned} \tag{3.9}$$

Using the component-wise distance computation of Equation 3.8 and following the same principles as in Section 3.1, we obtain an optimization problem which is similar to

Equation 3.3:

$$\begin{aligned} \min_{\mathbf{M}} \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} (\vec{D}^{ij})^\top \mathbf{M} \vec{D}^{ij} + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l \in \mathcal{I}_i^k} \xi_{ijl} \\ \text{s.t.} \quad & (\vec{D}^{il})^\top \mathbf{M} \vec{D}^{il} - (\vec{D}^{ij})^\top \mathbf{M} \vec{D}^{ij} \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \;\; \mathbf{M} \succeq 0, \;\; \forall i,j \in \mathcal{N}_i^k, l \in \mathcal{I}_i^k. \end{aligned} \tag{3.10}$$

This problem can be solved by means of semi-definite programming. As suggested by (Kilian Q Weinberger and Lawrence K Saul 2008), we can use several speedups for both the training and the test phases of the algorithm. Specifically, to reduce the size of triplet choices $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$to check in every iteration of the training, we can employ an active set strategy suggested by (Kilian Q Weinberger and Lawrence K Saul 2008). Such a strategy significantly reduces the size of the optimization problem of Equation 3.8 in practice by limiting the possible triplets for each $\vec{x}_i$. Additionally, a ball-tree search (Ting Liu et al. 2005) can be employed to reduce the test time of *k*NN , and it also improves the training time of the LMNN algorithm by limiting the search size for the imposters (Kilian Q Weinberger and Lawrence K Saul 2008).

Note that the computational complexity of the DTW-LMNN algorithm is the same as for vectorial LMNN; further, the convexity of the problem is preserved. Nevertheless, a pre-training computation time is incurred due to the calculation of DTW distances, which is $\mathcal{O}(N^3)$.

Again, as explained in Section 3.1, a restriction to low-rank matrices **M** and **L** is possible, provided the relevant information is located in a low-rank subspace of the full data space only. In the next section, I discuss the feasibility of target neighbors and the proposed modification to Equation 3.3, which considers this additional measure to adjust the selected targets.

3.3 FEASIBILITY BASED LARGE MARGIN NEAREST NEIGHBORS

This section focuses on the relationship between selected neighboring targets and the feasible set of LMNN's optimization problem. I show that the wrong choices of targets can severely shrink the regime of feasible solutions of the optimization problem. To mitigate this problem, I introduce a feasibility measure that quantifies the impact of neighboring points with respect to the size of the feasible set, and I use this measure as a weighting scheme in a modified version of LMNN. I also extended this modification to the proposed DTW-LMNN algorithm for motion data and other multivariate time-series.

The optimization problem of the LMNN algorithm (Equation 3.3) constitutes a convex problem with respect to **M** if the targets $\mathcal{N}_i^k$ and impostors $\mathcal{I}_i^k$ are fixed (Kilian Q. Weinberger and Lawrence K. Saul 2009). Nevertheless, different selections for these initial targets can lead to different solution **M**. As suggested in (Kilian Q. Weinberger and Lawrence K. Saul 2009; Göpfert, Paassen, and Hammer 2016), a better strategy is to repeat LMNN's optimization multiple times (multiple-pass LMNN) while updating $\mathcal{N}_i^k$ and $\mathcal{I}_i^k$ in each run based on the resulting quadratic form **M**. However, this strategy also relies on the quality of the initial selection of these two sets. In this section, I focus first on the geometric formation of the targets and impostors and its effect on the feasibility of a solution **M**. Then, I propose an extension to the LMNN method, which benefits from this geometrical analysis to increase LMNN's robustness against the relevant local minimums.

Infeasible Target Neighbors

Considering Equation 3.3, we are interested in finding existing feasible solutions of this optimization problem that do not require slack variables $\xi_{ijl} > 0$. Accordingly, this feasible regime is given as

$$S := \{\mathbf{M} \in \mathbb{R}^{d\times d} | \mathbf{M} \succeq 0, \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) < \mathcal{D}_{\mathbf{M}}(\vec{x}_i, \vec{x}_l) \;\; \forall i, j \in \mathcal{N}_i^k, l \in \mathcal{I}_i^k\}. \tag{3.11}$$

For a triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$ the metric constraint in Equation 3.11 can be rewritten as:

$$\operatorname{tr}[\mathbf{Q}_{ijl}\mathbf{M}] := \operatorname{tr}[((\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^\top - (\vec{x}_i - \vec{x}_l)(\vec{x}_i - \vec{x}_l)^\top)\mathbf{M}] < 0. \tag{3.12}$$

Since $\mathbf{M}$ is positive semidefinite (PSD), a PSD matrix $\mathbf{Q}_{ijl}$ leads to the infeasibility of Equation 3.12, whereby this fact depends on the triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$, only, and not the specific neighborhood. In this section, I discuss an extremal case, where such a triplet induces an infeasible constraint for which I propose a proper measure with an exact geometric interpretation. In the next section, I generalize this measure to a suitable weighting scheme for more general settings.

Theorem 3.1. *A triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$ results in Equation 3.12 being infeasible if $(\vec{x}_i - \vec{x}_j)$ and $(\vec{x}_i - \vec{x}_l)$ are linearly dependent vectors.*

Proof. Refer to Appendix A.1. □

As a 2-dimensional (2D) illustration for the infeasible case of Theorem 3.1, consider a small neighborhood of data points in a 2D space as in Figure 3.2(a) in which $\vec{x}_i$ is the main data point and $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$ are its targets while $\vec{x}_4$ is a close imposter point. As depicted in the figure, $\vec{x}_4$ lies on the connecting line between $\vec{x}_i$ and $\vec{x}_1$, which makes $(\vec{x}_i - \vec{x}_1)$ and $(\vec{x}_i - \vec{x}_4)$ linearly dependent. Hence, if we include the triplet $(i, 1, 4)$ in the constraints of the optimization problem in Equation 3.10, its solution transform $\mathbf{M}$ brings the target points closer to $\vec{x}_i$ (Figure 3.2(b)). However, the triplet $(i, 1, 4)$ still does not satisfy the inequality in Equation 3.12. As a result, the nearest neighbor to $\vec{x}_i$ is still the imposter $\vec{x}_4$.

The infeasible case in Theorem 3.1 does not allow a feasible solution without slack variables. In the following, I will argue that the measure $r := -\lambda_{\min}(\mathbf{Q})/\lambda_{\max}(\mathbf{Q})$ constitutes a reasonable weight vector to measure the feasibility of the constraint corresponding to $\mathbf{Q}$ or the size of its feasible domain, respectively. Obviously, $r = 0$ is the case just described, an infeasible setting due to the geometry of $\vec{a} = (\vec{x}_i - \vec{x}_j)$ and $\vec{b} = (\vec{x}_i - \vec{x}_l)$.

Feasibility Measure

I start with a general observation:

Lemma 3.1. *Denote the eigenvalues of a matrix $\mathbf{Q} \in \mathrm{R}^{d\times d}$ by $\lambda_1(\mathbf{Q}) \geq \lambda_2(\mathbf{Q}) \geq \ldots$, its smallest/largest eigenvalue is denoted $\lambda_{min}(\mathbf{Q})$ and $\lambda_{max}(\mathbf{Q})$, respectively. Then, for hermitian $\mathbf{Q} \in \mathrm{R}^{d\times d}$ and symmetric PSD $\mathbf{M} \in \mathrm{R}^{d\times d}$, it holds $\lambda_k(\mathbf{Q})\lambda_{min}(\mathbf{M}) \leq \lambda_k(\mathbf{QM})$ for all k.*

Proof. Refer to Appendix A.2. □

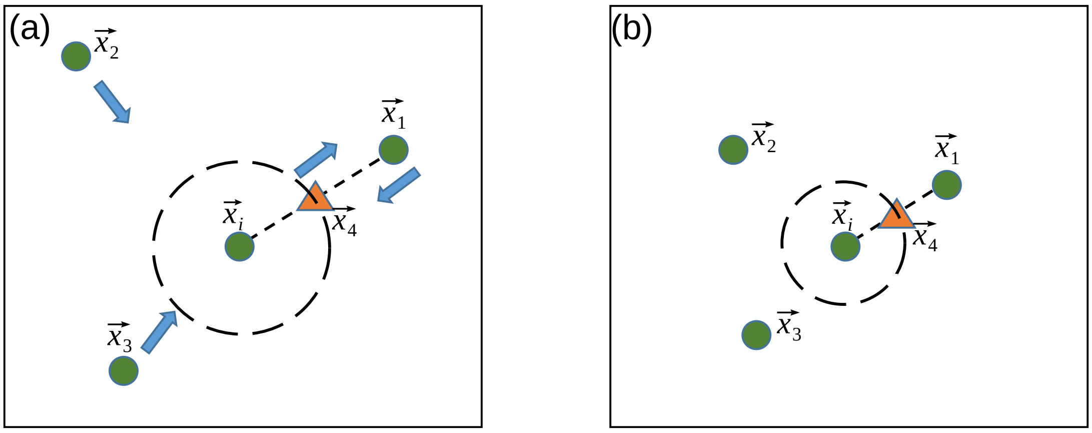


Figure 3.2: A 2-D example of an infeasible triplet set for Equation 3.12. (a) The triplet $(i, 1, 4)$ is included in the set of constraints of Equation 3.10 while the vectors $(\vec{x}_i - \vec{x}_1)$ and $(\vec{x}_i - \vec{x}_4)$ are linearly dependent. (b) A linear transform solution $\mathbb{M}$ (from Equation 3.10) still does not provide a feasible inequality for the triplet $(i, 1, 4)$ as it is desired in Equation 3.12.

Based on Lemma 3.1, we have

$$\lambda_{max}(\mathbf{Q})\lambda_{min}(\mathbf{M}) \leq \lambda_{max}(\mathbf{QM})$$

for $\mathbf{Q} := \mathbf{Q}_{ijl}$ as specified in Equation 3.12. In the setting $\lambda_{min}(\mathbf{Q}) < 0 < \lambda_{max}(\mathbf{Q})$, we can use the Corollary 10 from (F. Zhang, Qingling Zhang, et al. 2006) to infer $\lambda_{min}(\mathbf{Q})\lambda_{max}(\mathbf{M}) \leq \lambda_{min}(\mathbf{QM})$. Combining these two inequalities results in the inequality

$$\lambda_{min}(\mathbf{Q})\lambda_{max}(\mathbf{M}) + \lambda_{max}(\mathbf{Q})\lambda_{min}(\mathbf{M}) \leq \mathrm{tr}(\mathbf{QM}) \tag{3.13}$$

Equation 3.12 induces the objective $\mathrm{tr}(\mathbf{QM}) < 0$, hence the left hand side of Equation 3.13 should be negative, i.e. $-\frac{\lambda_{min}(\mathbf{Q})}{\lambda_{max}(\mathbf{Q})} > \frac{\lambda_{min}(\mathbf{M})}{\lambda_{max}(\mathbf{M})}$. Hence a triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$, I defined the feasibility measure as

$$r = -\frac{\lambda_{min}(\mathbf{Q})}{\lambda_{max}(\mathbf{Q})} \tag{3.14}$$

according to the definition of $\mathbf{Q}$ in Equation 3.12. Therefore, with a small r, such a triplet imposes a tight constraint on the eigenvalue formation of $\mathbf{M}$, resulting in an induced small feasible set S_{ijl}. Later, Note that the metric's feasible domain S in Equation 3.11 is formed as an intersection of the feasible sets S_{ijl}. I include this observation and the obtained measure $r = r_{ijl}$ into the optimization framework of Equation 3.3 in the form of a weighting scheme.

Feasibility-based Large Margin Nearest Neighbors Metric Learning

For a vector $\vec{x}_i$ and a given target $\vec{x}_j \in \mathcal{N}_i^k$, I define $R_{ij} := \min_{\vec{x}_l \in \mathcal{I}_i^k}(r_{ijl})$. Consequently, I formulate *feasibility-based LMNN* as the following optimization problem, which incorporates according feasibility weights in its objective:

$$\begin{aligned} \min_{\mathbf{M}} \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} R_{ij}\mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) + \mu \sum_{i,j \in \mathcal{N}_i^k} R_{ij} \sum_{l \in \mathcal{I}_i^k} \xi_{ijl} \\ \text{s.t.} \quad & \mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_l) - \mathcal{D}_{\mathbf{L}}(\vec{x}_i, \vec{x}_j) \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \quad \mathbf{M} \succeq 0, \quad \forall i, j \in \mathcal{N}_i^k, l \in \mathcal{I}_i^k. \end{aligned} \tag{3.15}$$

By definition, R_{ij} applies Equation 3.14 according to the impostor $\vec{x}_l$ with the worst geometrical formation w.r.t. $(\vec{x}_i, \vec{x}_j)$, where $\vec{x}_j$ is the target for $\vec{x}_i$. Such weighing choice comes from the fact that the worst $\vec{x}_l$ for a given target $\vec{x}_j$ limits the feasible set of $\mathbf{M}$. Therefore, unlike the original LMNN, infeasible or challenging triplets carry less weighting in this formulation.

In order to extend the problem of Equation 3.15 to its DTW-based version, we can apply the same feasibility principle to the component-wise DTW vector $\vec{D}^{ij}$ from Equation 3.7. To that aim, $\mathbf{Q}_{ijl}$ from Equation 3.12 is calculated as

$$\mathbf{Q}_{ijl} = \mathbf{D}^{ij}\mathbf{D}^{ij\top} - \mathbf{D}^{il}\mathbf{D}^{il\top},$$

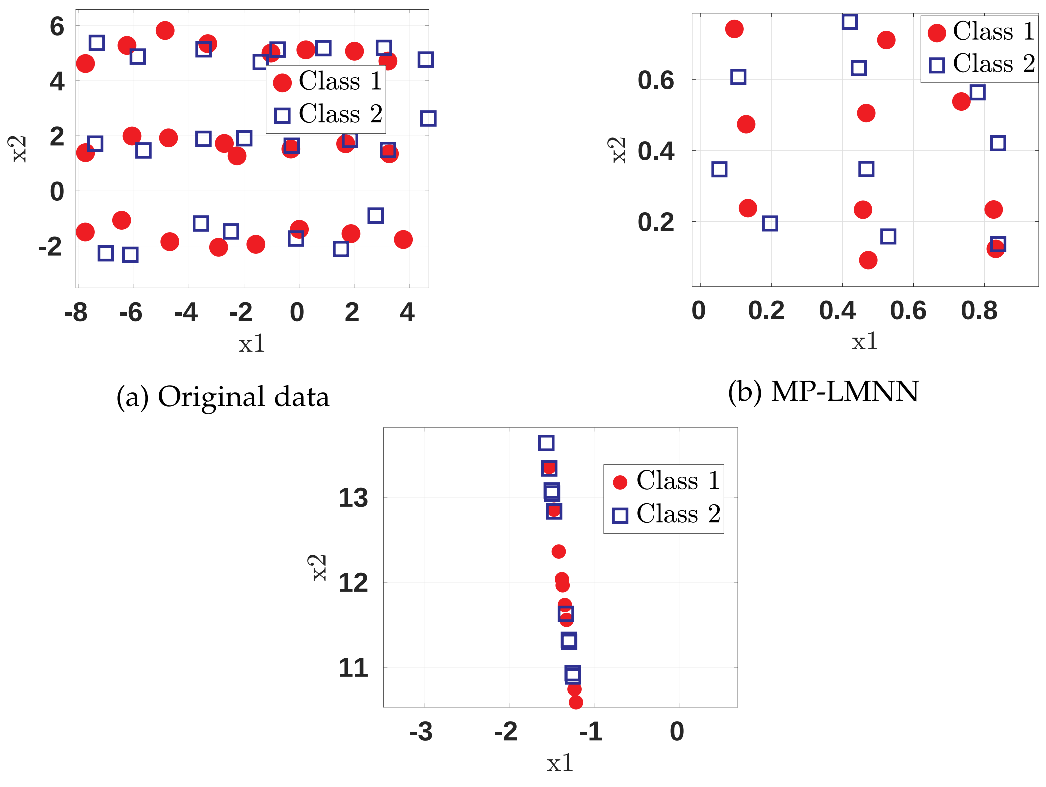
and consequently, the vectorial optimization problem of Equation 3.15 is extended to

$$\begin{aligned} \min_{\mathbf{M}} \quad & (1-\mu) \sum_{i,j \in \mathcal{N}_i^k} R_{ij} (\vec{D}^{ij})^\top \mathbf{M} \vec{D}^{ij} + \mu \sum_{i,j \in \mathcal{N}_i^k} R_{ij} \sum_{l \in \mathcal{I}_i^k} \xi_{ijl} \\ \text{s.t.} \quad & (\vec{D}^{il})^\top \mathbf{M} \vec{D}^{il} - (\vec{D}^{ij})^\top \mathbf{M} \vec{D}^{ij} \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \ \ \mathbf{M} \succeq 0, \ \ \forall i, j \in \mathcal{N}_i^k, l \in \mathcal{I}_i^k. \end{aligned} \tag{3.16}$$

I dub the resulting algorithm FDW-LMNN, which is implemented by first determining the neighborhoods, computing corresponding weights R_{ij}, and then solving the convex optimization problem w.r.t. matrix $\mathbf{M}$. The optimization problem of FDW-LMNN follows the same principles as for DTW-LMNN with the same computational cost. Nevertheless, computation of R_{ij} has the time complexity of $\mathcal{O}(N_t d^3)$, or $\mathcal{O}(N_t d^w)$ with $2 < w < 2.376$ based on (Demmel, Dumitriu, and Holtz 2007), where N_t is the total number of triplets $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$ in Equation 3.16. As a practical speedup, we can use the active-set and ball-tree strategies of (Kilian Q Weinberger and Lawrence K Saul 2008), which significantly reduces N_t and consequently the computational complexity of FDW-LMNN .

For an efficient implementation, as I suggested for DTW-LMNN , a multiple passes strategy and specific speedups can be used to increase the base performance of the FDW-LMNN algorithm (Kilian Q Weinberger and Lawrence K Saul 2008; Kilian Q. Weinberger and Lawrence K. Saul 2009). It is also more practical to choose the neighborhood size k of FDW-LMNN a few samples larger than that of DTW-LMNN . This way, we can benefit from the weighting scheme of Equation 3.16, while still using the same number of effective targets as in DTW-LMNN . Again, analogous to the DTW-LMNN algorithm in Section 3.2, assuming the given task's significant information for the given task is located in a low-rank subspace of the data, it is possible to solve Equation 3.16 for a low-dimensional counterpart $\mathbf{L} : \mathbb{R}^d \rightarrow \mathbb{R}^{\tilde{d}}$ where $\tilde{d} < d$.

In order to better illustrate the rationale behind the explained feasibility concept and visualize its effect on LMNN's solution, I provide an experiment on a synthetic dataset. The utilized synthetic dataset is a variation of the 2D Zebra stripe data from (Kilian Q. Weinberger and Lawrence K. Saul 2009) in which two classes of data are alternately distributed on vertical stripes (Figure 3.3(a)). In contrast to the original Zebra dataset in (Kilian Q. Weinberger and Lawrence K. Saul 2009), the nearest target(s) $\vec{x}_j$ to each data point $\vec{x}_i$ is located on the neighboring stripe of $\vec{x}_i$. In other words, the Euclidean distance of $\vec{x}_i$ to the closest neighbor on its own vertical stripe is much larger than $\mathcal{D}(\vec{x}_i, \vec{x}_j)$ for $\vec{x}_j$ on a neighboring stripe. Hence, no matter how big the neighborhood radius k is chosen, the above such target $\vec{x}_j$ will be chosen among $\mathcal{N}_i^k$. As depicted in Figure 3.3(a), the selected nearest target for each $\vec{x}_i$ and its corresponding impostor is almost located on



(a) Original data

(b) MP-LMNN

(c) FB-LMNN

Figure 3.3: (a) Zebra dataset created based on alternative stripes of two data classes. The closest target to each data point is located on a neighboring stripe of the same class, while an impostor exists near their connecting line. (b) The metric learned by MP-LMNN transfers the data to a distribution with a similar class-formation as in (a). (c) FB-LMNN algorithm learns a metric by selecting the more promising targets (on the same stripes) located farther than the closest neighbors.

a straight line, which results in very tight or infeasible constraints in the optimization framework of Equation 3.3.

As in Figure 3.3(b), even the multiple-pass LMNN (MP-LMNN) hardly changes this selection of impostors and targets. Therefore, even repeating LMNN in a loop of multiple passes is not effective because the infeasible neighboring targets still remain as the closest neighbors in each pass. Hence, such members in $\mathcal{N}_i^k\ \forall i$ yielding in low feasible constraint sets. Consequently, MP-LMNN converges to a non-optimal solution $\mathbf{M}$ with a classification accuracy of 23.51% (almost the same as kNN 's). On the other hand, feasibility-based LMNN (FB-LMNN) assigns small R_{ij} weights to pairs within the same stripe while bigger weights to pairs located on different stripes. Therefore, it obtains a different matrix $\mathbf{M}$ resulting in a more efficient scaling of the space, as in Figure 3.3(c), which consequently leads to a classification accuracy of 72.21%.

In the next section, I explore the application of the metric regularization for the distance-based metric obtained from DTW-LMNN or its feasibility-based variant.

3.4 METRIC REGULARIZATION

Besides the classification accuracy, we are also interested in the feature relevance profile which can be obtained from the diagonal entries of **M**. For DTW-LMNN, this interpretation directly transfers to a relevance profile for the sequential data related to each feature component, such as single joints in the case of Mocap data. For the metric obtained from Equation 3.10, the diagonal entry M_{kk} summarizes the influence of pairwise distances computed based on the sensory feature k. From another perspective, relatively large M_{kk} entries can indicate a considerable semantic connection between a particular joint(s) of the body and the given classification task.

Nevertheless, as pointed out in Section 3.1, the obtained profile and the linear transformation **L** may contain considerable redundant information. This issue typically arises for high dimensional data or highly correlated features. Therefore, it is always essential to perform a matrix regularization similar to Equation 3.6 to obtain an equivalent class of **L** with the smallest Frobenius norm. This reformulation as matrix regularization has the benefit that its principle can directly be transferred to more general data such as the alignment vectors $\vec{D}^{ij}$, as we see in the following.

For alignment vectors Equation 3.7 and the distance Equation 3.8, we find

$$\begin{aligned}&\mathbf{L}_1\vec{D}^{ij} = \mathbf{L}_2\vec{D}^{ij} \text{ for all } i,j\\ \iff\ &(\mathbf{L}_1 - \mathbf{L}_2)\vec{D}^{ij} = 0 \text{ for all } i,j.\end{aligned} \tag{3.17}$$

Hence, similar to Equation 3.5, transformations are equivalent with respect to the given data iff their difference lies in the null space of the correlation matrix $\mathbf{D}\mathbf{D}^\top$ for the distance matrix $\mathbf{D} := [\vec{D}^{11}, \dots, \vec{D}^{1N}, \dots, \vec{D}^{N1}, \dots, \vec{D}^{NN}]$, consisting of all d-dimensional vectors of pairwise distances. Note that this observation enables an effective regularization of the matrix **L** (and $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$) in the same way as for the vectorial case, relying on the regularization Equation 3.6:

$$\begin{aligned}&\tilde{\mathbf{L}} := \mathbf{L}\Phi\\ &\text{where } \Phi = \textstyle\sum_{s=1}^{S} \vec{u}_s(\vec{u}_s)^\top \text{ with the eigenvectors}\\ &\vec{u}_1, \dots, \vec{u}_S \text{ of } \mathbf{D}\mathbf{D}^\top \text{ with non-vanishing eigenvalues.}\end{aligned} \tag{3.18}$$

As for the vectorial case, this yields the equivalent matrix $\tilde{\mathbf{L}}$ of **L** with the smallest Frobenius norm, for which an interpretation of the diagonal entries becomes feasible. Thereby, this principle is applicable for full-rank matrices as well as low-rank counterparts. We see in the experiments section that matrix regularization has a substantial effect on the variance of the resulting relevance profile. Further, it can also enable a slightly better generalization ability since it suppresses noise in the given data.

Following this section, I discuss the experimental results to evaluate the proposed algorithms DTW-LMNN, FDW-LMNN, and the metric regularization technique.

3.5 EXPERIMENTS

In this section, I implement my proposed methods DTW-LMNN and FDW-LMNN on different multivariate time-series and compare their performance to alternative approaches. In addition, I evaluate the proposed metric regularization by performing feature extraction using the obtained relevance profile.

Experiments Setup

I use the learned metrics of the proposed LMNN-based methods to classify data using the k-nearest neighbor (kNN) method. For all of our experiments, I use $k = 3$ as the decision parameter of any implemented kNN algorithm. For all LMNN-based implementations, I employ the multiple passes strategy of (Kilian Q. Weinberger and Lawrence K. Saul 2009), which reduces the training's sensitivity to the initial selection of target points. Additionally, the neighborhood size k and weighting parameter μ for LMNN variants are chosen using cross-validation (CV) on the training set. I evaluate the proposed methods based on the classification accuracy $(100 \times [\texttt{\#correct predictions}]/N)$ in a 10-fold cross-validation setting (averaged over 10 repetitions), where N is the total number of test-data. For evaluation and comparison of the proposed approach, I consider the following human motion capture datasets: Walking , Dance , Cricket , and Words , introduced in Section 2.4. The performance of the FDW-LMNN algorithm was also evaluated on vectorial data. The reader can find the respective experiments in (Hosseini and Hammer 2018b).

Alternative Methods

I use the following alternatives as baselines for empirical evaluation of the proposed methods:

kNN : Without applying any metric adaptation, the obtained Euclidean distance-matrix of the training data is directly used to classify the test data with the kNN algorithm. When any two given time-series have different lengths, I calculate their pairwise Euclidean distance by uniformly down-sample the longer sequence according to the length of the shorter one.

DTW: Without applying any metric learning, I directly use the training data's computed DTW distance matrix to classify the test data via the kNN method.

Euc-LMNN: This baseline is obtained by replacing the DTW distance with standard Euclidean metric in DTW-LMNN. More precisely, I use the LMNN formulation (3.10) with

$$\vec{D}^{ij} := (\mathcal{D}_{\text{Euc}}(\vec{x}_i^1, \vec{x}_j^1), \ldots, \mathcal{D}_{\text{Euc}}(\vec{x}_i^d, \vec{x}_j^d)) \in \mathbb{R}^d, \tag{3.19}$$

where $\mathcal{D}_{\text{Euc}}$ denotes the Euclidean distance between any two time series $\mathbf{X}_i$ and $\mathbf{X}_j$.

DTW-SVM: The multi-class support vector machine (SVM) algorithm (Crammer and Singer 2001) with radial basis function kernel calculated based on the DTW distance-matrix and the clip eigenvalue correction

$$\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) = e^{(-\mathcal{D}_{\text{DTW}}(\mathbf{X}_i,\mathbf{X}_j)^2/\delta)},$$

where $\mathcal{D}_{\text{DTW}}(\mathbf{X}_i, \mathbf{X}_j)$ denotes the cumulative DTW distance between $\mathbf{X}_i$ and $\mathbf{X}_j$, i.e. $\sum_{s=1}^{d} \mathcal{D}_{\text{DTW}}(\vec{x}_i^s, \vec{x}_j^s)$. The scalar δ is equal to the average distance for all training data points.

PCA-DTW: For evaluation of the low-rank DTW-LMNN, I compare it to PCA-DTW. After applying the Principle Component Analysis (PCA) (Jolliffe 2002) to the raw time-series, the DTW distance matrix is calculated based on the obtained first 3 principal components of the data, followed by the kNN classifier.

To study the significance of differences in the empirical results, I perform the paired t-test, which tests the hypothesis that two experiments (two CV results) are generated from the same distribution. (M. C. Seiler and F. A. Seiler 1989).

Classification Accuracy

I compare the classification performance of the proposed algorithms DTW-LMNN and FDW-LMNN to the alternative methods. The classification accuracy on the selected datasets is given in Table 3.1, along with their variances and the calculated p-values.

According to the results, DTW-LMNN and FDW-LMNN outperform the Euclidean version of the LMNN algorithm for all datasets. This observation supports the expectation that DTW constitutes a suitable dissimilarity measure for motion data due to its flexibility regarding motions with different lengths. From another point of view, the metric adjustment for both Euc-LMNN and DTW-LMNN provides classification accuracy improvements for all cases compared to the *k*NN and DTW baselines. Interestingly, it even causes a slight superiority of the Euclidean metric over DTW for the Dance dataset when the Euclidean distance matrix benefits from metric adaptation. Based on Table 3.1, FDW-LMNN outperforms the DTW-LMNN algorithm for Dance and Words datasets. We can conclude that for these two datasets (especially for the Dance dataset), the feasibility criteria of FDW-LMNN is effective regarding their class distributions.

Low-rank Matrix Representation

I study the dimensionality reduction performance of DTW-LMNN and FDW-LMNN on the datasets introduced in Section 3.5. The discriminative quality of the obtained low-rank representations is judged based on the resulting classification accuracies. As mentioned in section 3.2, we can have a low-rank solution matrix **M** or **L** for the optimization problem (Equation 3.10). Apart from a compressed representation, this can lead to a significant increase in the time performance of the *k*NN classification in the low-dimensional projection space (Kilian Q Weinberger and Lawrence K Saul 2008). I use a rank 3 matrix **L** corresponding to a projection into the space $\mathbb{R}^3$. For comparison, I also investigate

Table 3.1: Comparison of the algorithms based on the classification accuracy (%) for the four selected datasets. The paired t-test checks the hypothesis that the winner and the runner-up methods are not significantly different. The best result for each dataset is highlighted.

Method	Walking	Dance	Cricket	Words
*k*NN	90.23 ± 1.45	72.48 ± 2.66	92.16 ± 0.51	94.54 ± 2.31
Euc-LMNN	92.32 ± 0.87	80.41 ± 1.49	95.56 ± 0.38	97.30 ± 1.20
DTW	95.44 ± 0.77	77.51 ± 1.51	99.44 ± 0.18	98.61 ± 1.05
DTW-SVM	95.68 ± 0.46	78.53 ± 1.10	$\mathbf{100 \pm 0}$	98.72 ± 1.86
DTW-LMNN	$\mathbf{100 \pm 0}$	90 ± 1.03	$\mathbf{100 \pm 0}$	99.06 ± 1.11
FDW-LMNN	$\mathbf{100 \pm 0}$	$\mathbf{92.4 \pm 1.37}$	$\mathbf{100 \pm 0}$	$\mathbf{99.17} \pm 1.43$
p-value	–	0.02	–	< 0.01

Table 3.2: Comparison of the algorithms' low-rank (lr) implementations based on the classification accuracy (%) for the four selected datasets. Each dataset's best result is highlighted according to a paired t-test at a 5% significance level.

Method	Walking	Dance	Cricket	Words
Euc-LMNN (lr)	86.6±1.10	75±1.52	96.11±0.46	98.60±0.14
PCA-DTW	96.03±1.08	76±1.51	99.44±0.18	94.24±0.25
DTW-LMNN (lr)	**98.8±1.80**	95±0.80	**100±0**	99.12±0.17
FDW-LMNN (lr)	**98.8±1.80**	**97±1.02**	**100±0**	**99.46±0.13**
p-value	< 0.01	0.02	–	0.03

the effect of a rank restriction for the Euclidean version of LMNN, and I investigate the result of classical PCA for dimensionality reduction of the data before classification. The results of these low-rank classification pipelines are reported in Table 3.2.

According to the results, the low-rank versions of both DTW-based LMNN algorithms still have the best classification accuracies compared to other approaches. Furthermore, their accuracies for Words and Dance datasets are even improved compared to their full rank versions (Table 3.1). For these two datasets, the low-rank optimization scheme for DTW-based algorithms provides a more discriminative combination of the original features. In addition, DTW-LMNN and FDW-LMNN algorithms classify the Cricket dataset with 100% accuracy while obtaining a compressed representation as well. In contrast, PCA-DTW and low-rank Euc-LMNN have lower classification accuracies. The latter method decreases the accuracy for two datasets compared to full-rank Euc-LMNN, while PCA-DTW has accuracy improvement only for the Walking dataset. Hence projection directions learned by LMNN low-optimization (with a low-rank solution) can potentially enhance the discriminative aspect of DTW alignments in a low-rank matrix representation.

Regularized Relevance Profiles

In this section, I investigate the resulting relevance profiles for Dance and Walking datasets for the metrics obtained by DTW-LMNN. I use only two of the four previous datasets to focus on the matrix regularization's notable effects. The datasets Words and Cricket are of little interest for this section due to their comparably low-dimensionality (9 and 6 sensors only, without considerable correlations). On the contrary, the two full-body motion datasets (Dance and Walking) have a high number of features (62), with substantial correlations among their joints. Hence we can expect interesting effects when regularizing the learned matrix.

Matrix regularization has different effects: (I) It enables a valid interpretation of the feature relevance profile since it avoids spurious relevance peaks and random effects due to data correlations. I evaluate this effect by an inspection of the sparsity and variance of the relevance profile within cross-validation. (II) It suggests the possible ways to reduce the data dimensionality by eliminating the most irrelevant features according to the found relevance profile. I investigate this effect by evaluating the classification performance when the feature dimensions are iteratively selected (or removed) according to their relevance.

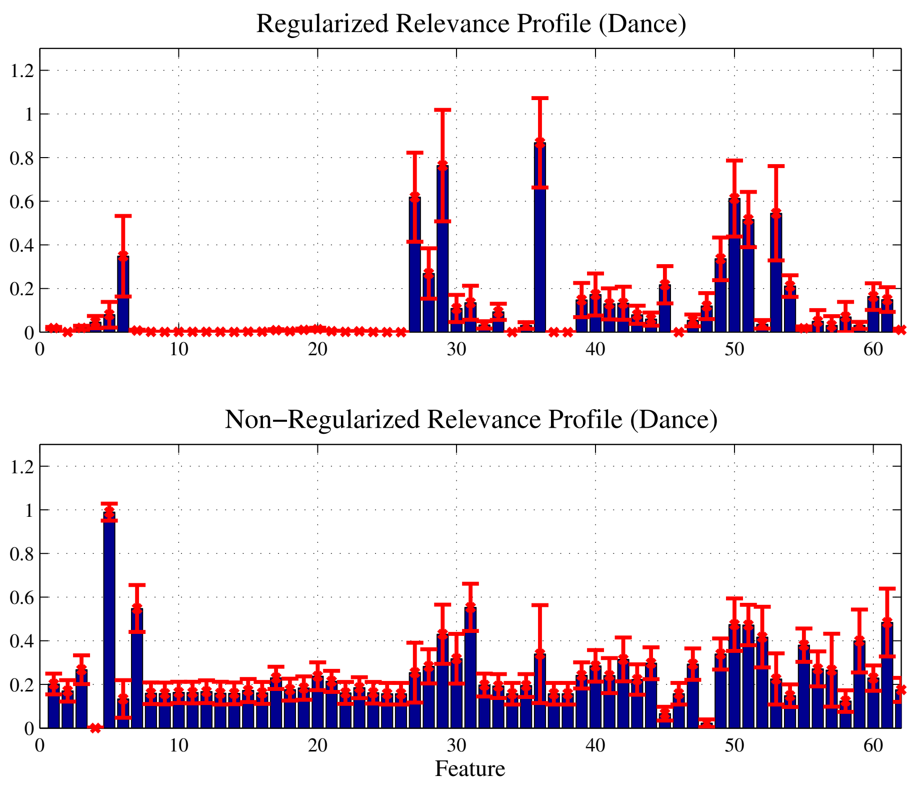


Figure 3.4: Average (blue bars) and deviation (red lines) of the relevance values for features of the Dance dataset calculated according to the normalized diagonal values of $(\mathbf{L}\top\mathbf{L})$. Top: Regularized relevance profile. Bottom: Non-regularized relevance profile.

Dance Dataset

For the Dance dataset, I calculate the relevance values of features as the diagonal entries of $(\mathbf{L}^\top\mathbf{L})$. The transformation matrix $\mathbf{L}$ is obtained via DTW-LMNN, which was applied in section 3.5. The resulting original relevance profile (without any regularization) is displayed in Figure 3.4-bottom, in which I normalized the profiles to the range $[0, 1]$. Since the value of $\mathbf{L}$ varies for different cross-validation splits, I report the average and variance of each diagonal entry over all splits. The total variance of the original relevance profile is 4.47.

In comparison, I regularize matrix $\mathbf{L}$ according to Equation 3.18. To that aim, the eigenvectors $\vec{u}_s$ of matrix $\mathbf{D}\mathbf{D}^\top$ that correspond to its non-zero eigenvalues are determined, where the distance matrix $\mathbf{D}$ is computed based on the training set. Figure A.1 shows how I choose 12 effective dimensions (eigenvectors) based on the corresponding eigenvalue profile of $\mathbf{D}\mathbf{D}^\top$ for the Dance dataset to construct the regularization matrix Φ. The resulting regularized profile, which is obtained from $\tilde{\mathbf{L}}$, is shown in Figure 3.4-top. It is clear that this profile has much fewer high values representing singled out relevant features compared to the original profile (Figure 3.4-bottom). In comparison, also the variance of this profile is reduced to 2.86.

Next, I utilize the learned metric for the benefit of feature selection. To that aim, I sort the input dimensions (features) according to their relevance values in Figure 3.4 in descending order. Then, I select the important features according to this order in the classifier by removing other rows in $\mathbf{L}$ (corresponding to other features) for the

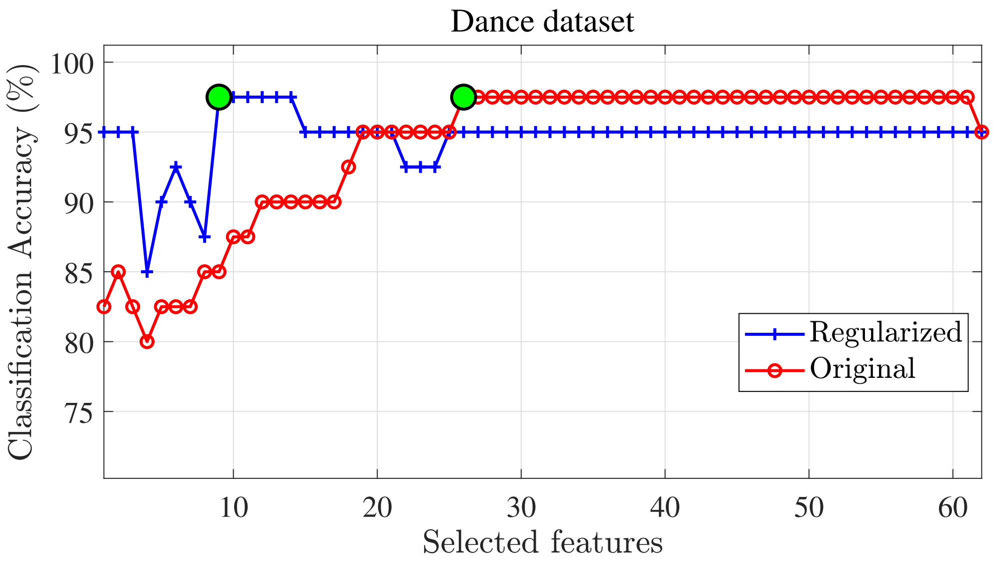


Figure 3.5: Classification accuracy of the row-reduced transformation **LX** for the Dance dataset. Non-zero rows of **L** correspond to the selected features according to the profiles in Figure 3.4.

transformation **LX**. Figure 3.5 shows the result of the above process on the test set's classification accuracy for different numbers of selected features. Interestingly, both relevance profiles in Figure 3.4 let us to remove a large number of sensors without reducing the classification accuracy. This selection process reaches its optimal point where only 9 and 26 features are chosen for the regularized and non-regularized profiles, respectively. Hence, the regularization technique greatly enhances the feature selection characteristic of the learned metric. The relevant body joints to the selected 9 features are depicted on the skeleton structure in Figure A.2. Additionally, the semantic meanings of these features are reported in Table. 3.3.

According to Figure 3.4, the regularization matrix Φ (Equation 3.18) positively affects the relevance profile. It reduces the profile's redundancy and produces a sparse representation for the relevance values of the input's features. Besides, based on Table 3.3, the regularized profile has smaller variances in the feature bars than the original profile and is more reliable regarding feature importances. Furthermore, Figure A.3 illustrates that for a wide range of effective dimensions in Equation 3.18, the classification accuracy for the test data stays at its maximum point.

As a semantic interpretation, it can be concluded from Figure A.2 that `hands` and

Table 3.3: Total variance in the regularized and non-regularized relevance profiles along with the feature selection result for the Dance dataset.

	Value
Profile variance (not-regularized)	4.47
Profile variance (regularized)	2.86
Selected joints (feature IDs)	root(6), rthumb(36), rfemur(49,50,51), rfoot(53), rhumerus(27,28,29)

`feet` are both important discriminative features for this dancing task. From another perspective, this is a difficult task because each class has different subcategories within itself, which account for overlaps with other classes; hence the combination of both (hand and foot) is required to distinguish between the two dance categories. Furthermore, as another interesting interpretation, only the data related to one side of the body (right side) is necessary to achieve the highest classification performance. This interpretation coincides with the fact that dancing is typically a symmetrical whole body movement in which symmetry can be found between the left and right sides of the body.

Walking Dataset

I repeat the previous experimental setting for the Walking dataset as well, upon which I select 14 effective dimensions to form the regularization matrix Φ. The obtained regularized profile is depicted in Figure 3.6, and the total variances of the relevance profiles before and after the regularization are 10.7 and 2.51, respectively. Again, similar to the Dance dataset, I perform feature selections using the relevance profiles of Figure 3.6. Consequently, the resulting classification accuracies and the selected essential joints are provided in Figure 3.7 and Tab. 3.4 respectively.

Similar to the Dance dataset, the regularization of the learned metric results in a sparse representation of the relevance profile and a reduced variance (Figure 3.6). Furthermore, according to Figure 3.7, a classification accuracy of 100% can be achieved while choosing fewer features (7 features) compared to the not-regularized profile (25 selected features).

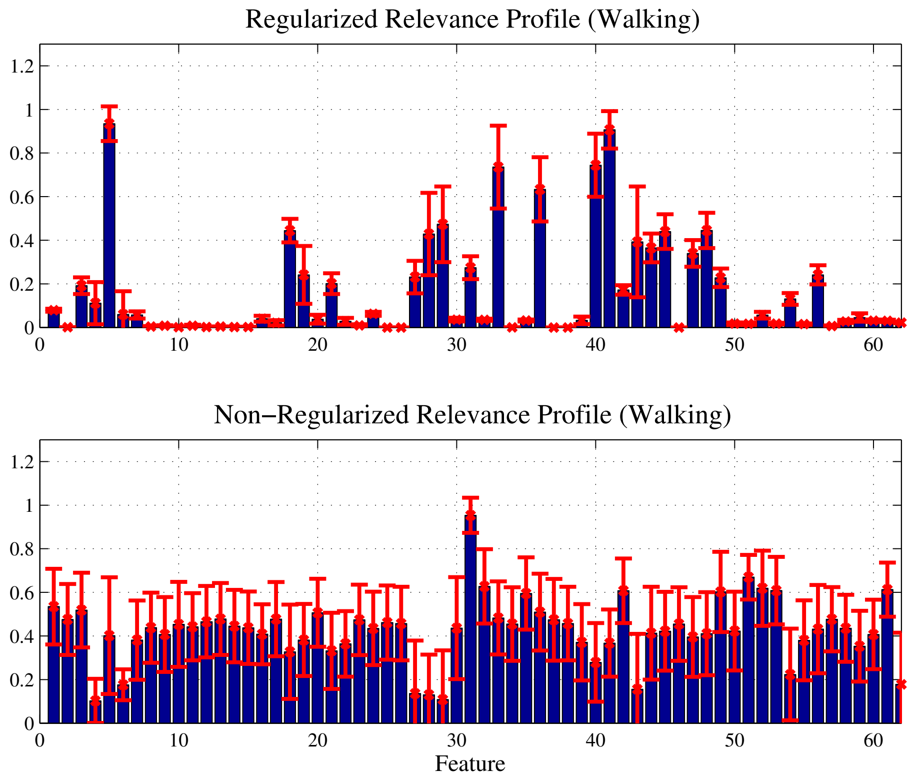


Figure 3.6: Average (blue bars) and deviation (red lines) of the relevance values for features of the Walking dataset calculated according to the normalized diagonal values of $(\mathbf{L}^\top\mathbf{L})$. `Top`: Regularized relevance profile. `Bottom`: Non-regularized relevance profile.

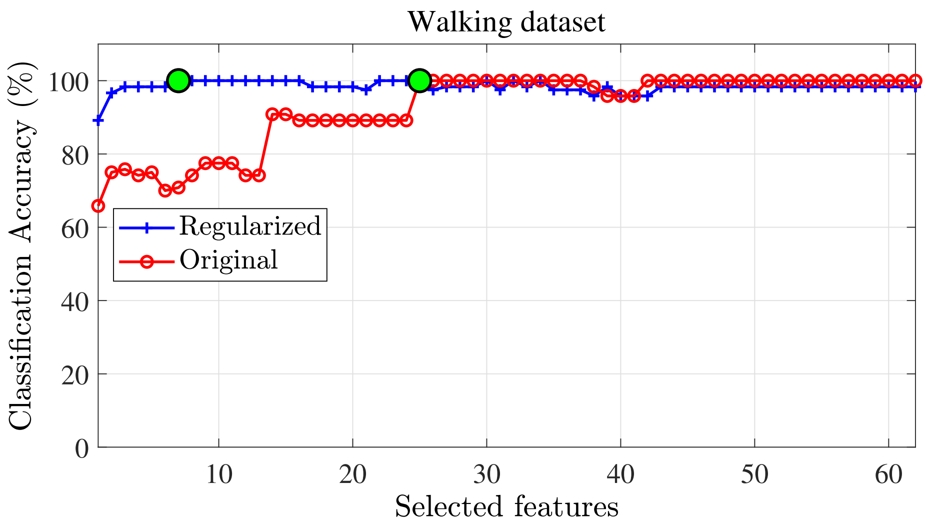


Figure 3.7: Classification performance of Walking dataset based on the selected features according to the regularized profile

Based on the observations from Figure A.4, for this dataset (and this classification task), `hands` are more important than `feet`. In addition, as the classes are very similar (all of them are connected to Walking), the classification algorithm needs to have input features from both sides of the body in order to carry out the classification task with a perfect result. I tested this hypothesis by using `Lhand` instead of `Rhand` or deleting `Rthumb` (since we already have `Lthumb`), but in both cases, the performance decreased (around 3% to 4%), showing that those selected joints are all necessary, even though they may look symmetrical in the skeleton structure.

3.6 CONCLUSION

In this chapter, I proposed a distance-based extension to the popular LMNN metric learning algorithm. This extension enables us to apply LMNN on motion data and other multi-dimensional time-series. By incorporating the DTW dissimilarity measure, which is particularly suited to mocap data analysis, I introduced the DTW-LMNN method. This algorithm benefits from a component-wise DTW-based representation of the distances in the given mocap dataset. Consequently, DTW-LMNN is able to recognize and adjust the relevance of particular joints that their movement pattern can semantically bring similar motions closer while keeping them farther away from other types of motions. In other words, by incorporating this distance-based representation to the LMNN framework,

Table 3.4: Total variance of the regularized and non-regularized relevance profiles and selected features for the Walking dataset.

	Value
Profile variance (not-regularized)	10.70
Profile variance (regularized)	2.51
Selected joints (feature IDs)	root(5), lhumerus(40,41), lowerneck(18), rthumb(36), rhand(33), lthumb (48)

we can efficiently adapt the feature ranking and correlation according to their semantic connection to the classification tasks at hand. In such resulted condensed representation, a distance-based decision making regarding the class of a given motion is highly interpretable according to its nearby motion sequence. Judging the quality of the DTW-LMNN 's learned metric based on K-nearest neighbor classification of real-world motion benchmarks, the proposed approach outperforms its Euclidean version as well as the sole application of DTW distance.

In section 3.5 of this chapter, I showed that the DTW-LMNN algorithm opens up the possibility of transferring auxiliary concepts such as metric regularization to motion data. I devised a method to apply metric regularization - which has been proposed for vectorial data Strickert et al. 2013 - to alignment-based representations. According to the results in section 3.5, this regularization step is a crucial prerequisite to obtain a valid interpretation of the relevance profile. This post-processing step removes the highly correlated dimensions related to the null space of the data correlation matrix. In other words, the regularization process signifies the specific joints which have a strong semantic connection to the defined classification task. As a result, it can enhance the semantic interpretability of the resulting metric. It is essential to mention that the above-proposed regularization step can be applied to any other dissimilarity-based metric framework as well.

In addition to the DTW extension of LMNN, I studied the feasibility of the constraints in LMNN's optimization problem according to selected target neighbors $\mathcal{N}_i^k$ for each data $\vec{x}_i$. I obtained a mathematical method to measure each target point's feasibility based on the distribution of its nearby impostors $\mathcal{I}_i^k$. I showed how some target points could cause tightly-feasible or even infeasible solution sets for LMNN's optimization constraints and therefore affecting the quality of the algorithm's solution metric in a negative way.

Accordingly, I reformulated the optimization problem to select the target points based on their feasibility measure. The proposed FDW-LMNN framework focuses more on targets with less tight constraints and highly feasible solution sets. Hence, it avoids infeasible targets and also has the potential to yield a more discriminant metric $\mathbf{M}$. Empirical results on real Mocap benchmarks showed that applying the feasibility measure can improve the quality of LMNN's metric for different data types by choosing more achievable target points in the dataset. Nevertheless, to use FDW-LMNN , one has to pay the additional computational cost prior to the algorithm's training phase. Therefore, a trade-off can be considered in practice between the complexity and accuracy while deciding between FDW-LMNN and DTW-LMNN .

Relying on the promising results achieved by the proposed DTW-LMNN and FDW-LMNN frameworks, there is considerable potential for future research on dissimilarity-based metric learning: the principle can be transferred to other metric learning methods which are not explicitly linked to the kNN classifier. Furthermore, the feasibility-based concept can be extended to its local application, which fits the local distance variation of the LMNN algorithm (Kilian Q Weinberger and Lawrence K Saul 2008). Another promising research line would be to investigate the application of more advanced regularization techniques (such as Frénay et al. 2014) on DTW-LMNN to achieve further enriched relevance profiles.

In this chapter, I performed an interpretable analysis of motion sequences by transforming the motion data into a new distribution, which magnifies the semantic similarity of motion sequences, and signifies their relevant dimensions to that goal. From another perspective, in the next chapter, I focus on finding interpretable embedding models for motion data. These models bring us sparse vector encodings for motion representation, which are also semantically interpretable w.r.t. the original motion sequences.

4 SPARSE CODING FOR INTERPRETABLE EMBEDDING OF MOTION DATA

Publications: This chapter is partially based on the following publications.

- Hosseini, Babak, Felix Hülsmann, et al. (2016). "Non-negative kernel sparse coding for the analysis of motion data". In: *International Conference on Artificial Neural Networks (ICANN)*. Springer, pp. 506–514.
- Hosseini, Babak and Barbara Hammer (2018a). "Confident kernel sparse coding and dictionary learning". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1031–1036.
- — (2018c). "Non-negative Local Sparse Coding for Subspace Clustering". In: *Advances in Intelligent Data Analysis XVII. (IDA)*. Ed. by Ukkonen A. Duivesteijn W. Siebes A. Vol. 11191. Lecture Notes in Computer Science. Springer, pp. 137–150. DOI: 10.1007/978-3-030-01768-2_12.

Recently, extensive applications of sparse coding (SRC) have been carried out in many areas of data analysis such as action recognition (Yao et al. 2015; W. Ding et al. 2018), text representation (Yogatama et al. 2015; Yang Li and T. Yang 2018), image classification (Jia et al. 2016; A. Li et al. 2018), recommendation systems (Qian et al. 2015; Z. Ji et al. 2019), and noise reduction (KyungHyun Cho 2013; Anada et al. 2019). An SRC algorithm aims to reconstruct an input signal using a weighted combination of a few selected entries from a set of learned base vectors. The vector of weighting coefficients and the set of bases are called the *sparse codes* and the *dictionary*, respectively. Such resulting sparse encoding can capture essential intrinsic characteristics of the dataset (T. Kim, Shakhnarovich, and Urtasun 2010; Rubinstein, Zibulevsky, and Michael Elad 2008), and can reconstruct a large number of data points using a relatively small set of training samples (Duarte-Carvajalino and Sapiro 2009). Focusing on semantic commonalities between similar (motion) data, I hypothesize that the learned dictionary can extract such meaningful entities by relying on natural priors such as sparsity.

In particular, relying on such representation of data, one can expect to observe similar characteristics between the sparse encoding of the observed and unobserved data given that they belong to the same source. This particular view can be extended to the test/train regime as the evaluation basis of many machine learning tasks. Accordingly, several data-driven works benefit from the sparse representation of data by applying other machine learning frameworks to the resulting sparse codes. For example, in (K. Huang and Aviyente 2007; Quan, Bao, and H. Ji 2016), the classification tasks are carried out by applying a support vector machine (SVM) classifier to the obtained sparse codes (K. Huang and Aviyente 2007; Quan, Bao, and H. Ji 2016). Also, in (C.-G. Li, You, and René Vidal 2017), the spectral clustering method is applied to the sparse codes to cluster the input data.

Regarding analyzing sequential data such as motion, some works have applied sparse coding frameworks directly on the raw input time-series (T. Kim, Shakhnarovich, and Urtasun 2010; Jin Wang et al. 2013; Takeishi and Yairi 2014). However, such applications to motion data face the inconsistency in the lengths of input samples as well as the multi-dimensional form of motion sequences. The first issue contradicts using dictionary elements of fixed size as a standard building block of a sparse coding framework. Also, the vectorial solution to the second problem can lead to notably high-dimensional dictionary elements. Therefore, such issues make these applications practically challenging for real-world motion data.

In this chapter, I will therefore take another stance, which enables a treatment of time series of different length by means of a flexible and possibly non-linear sparse decomposition, by relying on the kernel trick of machine learning and similar approaches. Accordingly, by considering an implicit mapping of the data to a high-dimensional feature space, it is possible to formulate SRC using a kernel-based representation of data (H. V. Nguyen et al. 2013; Z. Chen et al. 2015). Such kernels generally denote the pairwise similarity of data points in the given dataset. The resulting kernel sparse coding (K-SRC) methods can notably extend the application domain of SRC to structured data such as video processing (L. Xu et al. 2014), frame extraction (G. Xia et al. 2016), image segmentation (Tong et al. 2019), object recognition (Huaping Liu, D. Guo, and F. Sun 2016), and image classification (S. Gao, I. W.-H. Tsang, and L.-T. Chia 2010).

As observed in Chapter 3, DTW provides an elastic alignment of time series of possibly different lengths according to their semantic similarity. Therefore, DTW distance is a suitable candidate to construct a useful similarity kernel to represent sequential data (Ahmed, Paul, and Gavrilova 2015; Bahlmann, Haasdonk, and Burkhardt 2002; Shi et al. 2019). Fusing DTW-derived motion kernels with a proper kernel-based sparse coding extracts a dictionary from a given mocap data set, such that it enables a sparse vectorial representation of motion sequences (Hosseini, Hülsmann, et al. 2016; Huaping Liu, D. Guo, and F. Sun 2016; Z. Chen et al. 2015). In other words, the application of K-SRC on motion data results in an embedding from the sequential space to the vector space. Although such kernel-based frameworks enable us to obtain sparse encapsulated motion representations, they cannot provide a semantically interpretable encoding in terms of its constituent elements. More specifically, it is desired to see underlying connections between non-zero entries of the resulting encoding and semantically meaningful (motion-wise) parts of the model (the dictionary elements).

It is shown that the non-negative formulation of SRC frameworks can increase the possibility of relating each input signal to its semantically similar resources. In particular, such formulations can result in better classification results while also leading to a better interpretability of the sparse data encoding C. Zhang et al. 2011; Hazan, Polak, and Shashua 2005. Relatively, some works have proposed kernel-based non-negative matrix factorization algorithms, which can be considered as the kernel-based extension of SRC frameworks (Y. Zhang, T. Xu, and J. Ma 2017; D. Zhang, Z.-H. Zhou, and S. Chen 2006; Wenjun Wang et al. 2017; Yifeng Li and Ngom 2012). However, these algorithms do not apply any sparseness to their resulting encoding (even the (Y. Zhang, T. Xu, and J. Ma 2017)). Therefore, such kernel-based encodings are challenging to be interpreted with respect to their building blocks or the entities of their resulted embedded vectors. To address the above concern, I ask this follow-up question related to **RQ2**:

RQ2-a: How can we extend the kernel-based sparse coding frameworks to semantically interpretable embeddings?

From another perspective, it is expected from an interpretable embedding to carry the essential properties of the data, such as commonalities and supervised information. Such embedding constitutes an interface based on which semantic search becomes possible: motions that decompose into the same/similar dictionary elements have considerable semantic overlap. This concept is generally addressed as discriminative sparse codings, which employ discriminant terms in their encoding. Such additional terms usually incorporate the supervised information to project semantic similarities of the data to the resulting embedded sparse vectors. There are several variations of discriminative sparse coding which utilize different supervised terms, such as adding linear discriminants (Z. Jiang, Z. Lin, and Davis 2013; W. Liu et al. 2015), using regression operators (Bahrampour et al. 2015; Julien Mairal, Francis Bach, Jean Ponce, Sapiro, and Zisserman 2008; Julien Mairal, Jean Ponce, et al. 2009), or benefiting from ideas similar to the Fischer Discriminant (K. Huang and Aviyente 2007). Regardless of their achievements in improving the encoding's discriminative representation, they suffer from the lack of consistency between their training and the recall models (Hosseini and Hammer 2018a). While the supervised information plays a significant role in learning enriched sparse codes for training data, lack of that information in the recall phase of the algorithm degrades the quality of the encoding for the test data. Accordingly, another essential follow-up question for **RQ2** is:

RQ2-b: How can we increase the consistency between training and recall models for K-SRC frameworks to result in an enriched supervised embedding of non-observed data?

From another related perspective, another challenging task in many real-world motion datasets is to categorize underlying motion types without having any annotated training data available. In machine learning and data analysis, this concern is generally formulated as a clustering problem (R. Xu and Wunsch 2005), for which unsupervised methods try to discover the hidden structure of the data.

Accordingly, a subset of sparse coding works focus on using the sparse encoding vectors as the information source for the application of common clustering methods such as spectral clustering and k-means algorithms (Y. Yang, Zhangyang Wang, et al. 2014; Y. Yang, J. Feng, et al. 2016; C.-G. Li, You, and René Vidal 2017).

An important group of sparse coding methods for clustering is called sparse subspace clustering algorithms (SSC) (Elhamifar and Rene Vidal 2013). Assuming the data is distributed on a union of linear subspaces, SSC methods focus on obtaining a self-expressive representation, in which each data point is reconstructed by other similar samples from its underlying cluster (subspace) (X. Peng et al. 2018; Guangcan Liu et al. 2013; René Vidal and Favaro 2014).

Kernel-based SSC methods (K-SSC) extend the above idea to structure data as motions by relying on the pairwise similarity of data points. K-SSC methods have shown that such sparse self-expressive encoding can reveal the underlying subspaces in data distribution, in which data samples are semantically similar (Patel and René Vidal 2014; Yin et al. 2016; Xiaoqian Zhang et al. 2019).

Despite the success of K-SSC methods regarding both structured and vectorial data types, it is not easy to interpret the entities of their encoding vectors directly. On the one hand, some of these entities point toward data points in other subspaces based on weak existing similarities, causing negative redundancies in the obtained encoding. On the other, sparse encodings naturally create local sub-clusters within each subspace.

These issues makes the interpretation of these embedded vectors challenging for data types such as motions. As a common observation in motion datasets, many weak

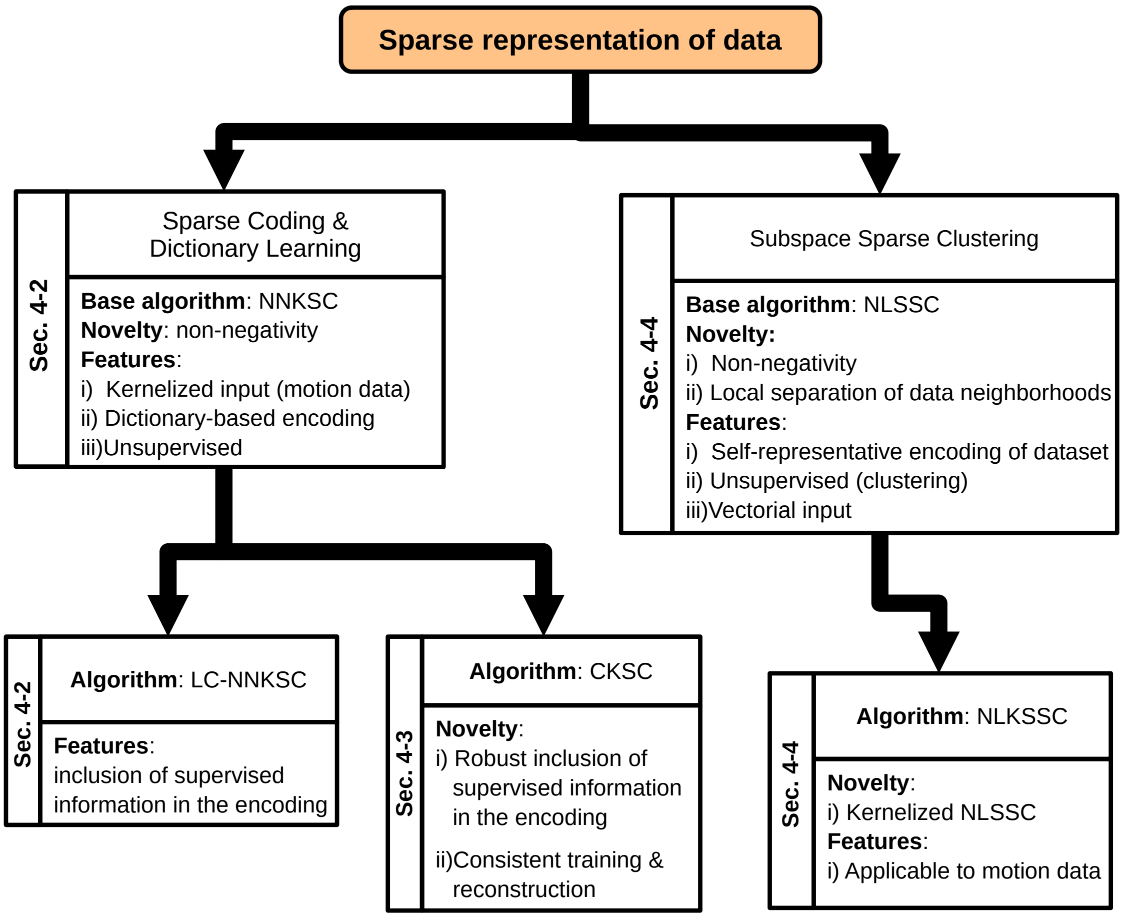

Sparse representation of data
Sec. 4-2
Sparse Coding & Dictionary Learning
Base algorithm: NNKSC
Novelty: non-negativity
Features:
i) Kernelized input (motion data)
ii) Dictionary-based encoding
iii)Unsupervised
Sec. 4-4
Subspace Sparse Clustering
Base algorithm: NLSSC
Novelty:
i) Non-negativity
ii) Local separation of data neighborhoods
Features:
i) Self-representative encoding of dataset
ii) Unsupervised (clustering)
iii)Vectorial input
Sec. 4-2
Algorithm: LC-NNKSC
Features:
inclusion of supervised information in the encoding
Sec. 4-3
Algorithm: CKSC
Novelty:
i) Robust inclusion of supervised information in the encoding
ii)Consistent training & reconstruction
Sec. 4-4
Algorithm: NLKSSC
Novelty:
i) Kernelized NLSSC
Features:
i) Applicable to motion data

Figure 4.1: Summary and hierarchy of proposed algorithms in Chapter 4 for sparse representation of data. The methods are generally divided into two main branches of sparse coding and subspace sparse clustering. Then, they expand into their supervised / unsupervised, kernelized / vectorial, or robust variant formulations.

similarities exist between different motion types, while many motion sequences can be grouped into small sub-clusters. Hence, my next follow-up research question related to **RQ2** is:

RQ2-c: How can we improve the interpretability of self-representative sparse encoding models?

In this chapter, I propose different supervised and unsupervised sparse coding and dictionary learning frameworks to investigate the above-proposed questions as summarized in Figure 4.1. These frameworks are suitable for the sparse encoding of motion data and other structured data given the kernel-based information is available. In particular, these proposed algorithms learn interpretable encodings with both supervised and unsupervised focuses. In addition, I design appropriate optimization algorithms learning the model parameters of the proposed encoding frameworks. In summary, I have the following contributions with respect to the state-of-the-art in kernel-based sparse coding.

- I propose a non-negative sparse coding framework NNKSC that uses similarity-based kernel information to encode motion sequences into sparse vectors. Such embedding represents and reconstructs each motion by using other semantically similar motion sequences. This specific framework provides an interpretable encoding as a suitable basis for further supervised or unsupervised motion data analysis.
- To enrich the proposed non-negative sparse encoding with supervised information, I extend my NNKSC algorithm to the supervised framework LC-NNKSC and its more robust variation CKSC . These algorithms focus on representing each input motion by contributions mostly taken from the same motion class. The outcome encoding is sparse, semantically interpretable, and preserves labeling information.
- I introduce a novel kernel-based sparse subspace clustering algorithm for clustering motion data (and other kernel-based representations). The proposed non-negative K-SSC method results in a sparse self-representative encoding of a motion dataset, where each sequences is mostly connected to other sequences of the similar type. The novel formulation and the post-processing step leads to an interpretable, unsupervised semantic grouping of motion data.

In the next section, I provide the necessary background for sparse coding and dictionary learning, along with their discriminative and kernel-based extensions. Then, the proposed non-negative sparse coding framework and its supervised and unsupervised extensions are explained in the consecutive sections. The chapter is concluded with the empirical evaluations on motion datasets and the summary of achievements.

4.1 STATE OF THE ART

In this section, I briefly study the background and state-of-the-art regarding sparse coding, its Kernel-based extension, its discriminant variant, and subspace sparse subspace clustering, which are related to the main principles of my proposed frameworks.

Sparse Coding

Denoting the training data matrix as $\mathbf{X} = [\vec{x}_1, ..., \vec{x}_N] \in \mathbb{R}^{d \times N}$, sparse coding is the idea of approximating each input signal as $\vec{x}_i \approx \mathbf{D}\vec{\gamma}_i$, where $\mathbf{D} \in \mathbb{R}^{d \times k}$ is the dictionary and $\mathbf{\Gamma} = [\vec{\gamma}_1, \ldots, \vec{\gamma}_N] \in \mathbb{R}^{k \times N}$ is the matrix containing the sparse codes. So, each sparse code $\vec{\gamma}_i$ uses a linear combination of the columns of $\mathbf{D}$ to reconstruct $\vec{x}_i$. Additionally, it is desired to find coefficient vectors $\vec{\gamma}_i$ which use limited resources from $\mathbf{D}$ such that:

$$\min_{\mathbf{\Gamma},\mathbf{D}} \|\mathbf{\Gamma}\|_0 \quad \text{s.t. } \mathbf{X} = \mathbf{D}\mathbf{\Gamma}, \tag{4.1}$$

where $\|.\|_0$ denotes the cardinality of $\mathbf{\Gamma}$. The exact equality is typically relaxed in practice and a suitable loss such as squared loss $\|\vec{x}_i - \mathbf{D}\vec{\gamma}_i\|_2^2$ is employed. This relaxation converts Equation 4.1 into

$$\min_{\mathbf{\Gamma},\mathbf{D}} \|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 \quad \text{s.t. } \|\vec{\gamma}_i\|_0 < T \;\; \forall i, \tag{4.2}$$

where $\|.\|_F$ denotes the Frobenius norm. Optimization of Equation 4.2 w.r.t. $\mathbf{\Gamma}$ is an NP-hard problem. However, its suboptimal solutions can be found using approximate methods like Orthogonal Matching Pursuit (OMP) (Pati, Rezaiifar, and Krishnaprasad 1993) or by relaxing the l_0-norm (cardinality) via using other norms such as $\|\mathbf{\Gamma}\|_1$ (Tibshirani 1996). On the other hand, optimizing (4.1) w.r.t. $\mathbf{D}$ is called dictionary learning (DL) for which various optimization strategies have been suggested such as the K-SVD algorithm (M. Aharon, M. Elad, and Bruckstein 2006), the Lagrangian method (H. Lee et al. 2006) and the online DL (Julien Mairal, Francis Bach, Jean Ponce, and Sapiro 2009). Generally, the problem of Equation 4.2 or its extensions are solved by an alternating optimization scheme which divides it into the following two individual optimization problems

$$\begin{aligned} &\text{Problem A: } \min_{\mathbf{\Gamma}} \|\mathbf{X} - \mathbf{D\Gamma}\|_F^2 \quad \text{s.t. } \mathcal{J}_{\mathbf{\Gamma}}(\mathbf{\Gamma}) \\ &\text{Problem B: } \min_{\mathbf{D}} \|\mathbf{X} - \mathbf{D\Gamma}\|_F^2 \quad \text{s.t. } \mathcal{J}_{\mathbf{D}}(\mathbf{D}). \end{aligned} \tag{4.3}$$

In Equation 4.3, the terms $\mathcal{J}_{\mathbf{\Gamma}}(\mathbf{\Gamma})$ and $\mathcal{J}_{\mathbf{D}}(\mathbf{D})$ denote the specific constraints applied to the sparse codes and the dictionary, respectively. These constraints vary based on the application or the algorithm (M. Aharon, M. Elad, and Bruckstein 2006; X. Lu et al. 2012; Vu and Monga 2017; M. Yang, H. Chang, and Luo 2017; Zhao Zhang et al. 2017)

Kernel-based SRC

In a vectorial sparse coding framework such as Equation 4.2, the input $\vec{x}_i$ is a vector in $\mathbb{R}^d$. However, as discussed in Section 2.1, a motion sequence is represented by a matrix $\mathbf{X} \in \; \times \mathbb{N}$. This raw representation is not consistent with the vectorial framework of Equation 4.2. In methods such as (T. Kim, Shakhnarovich, and Urtasun 2010; Jin Wang et al. 2013; Takeishi and Yairi 2014) the sparse coding problem is applied to the temporal axis of $\mathbf{X}$. This is achievable by synchronizing all $\mathbf{X}$ in the training set $\mathcal{X}$ regarding their temporal length. It is also required to reduce $\mathbf{X}$ to a 1D sequence by dimension reduction methods such as PCA or by concatenating all dimensions. Nevertheless, the application of such solutions for real-world motions, with different lengths, is challenging and inefficient.

On the other hand, it is shown that by incorporating kernel representation of data into the sparse coding framework, we can extend it to non-linear and non-vectorial domains (Li Zhang et al. 2011; X.-T. Yuan, X. Liu, and Shuicheng Yan 2012). Denote $\Phi : \mathbb{R}^d \to \mathbb{R}^h$ as an implicit non-linear mapping that can transfer data to a reproducing kernel Hilbert space (RKHS) with $d \ll h$. Therefore, we can use the kernel function $\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$ in the input space, which is associated with the implicit mapping Φ such that

$$\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) = \langle \Phi(\mathbf{X}_i), \Phi(\mathbf{X}_j) \rangle, \tag{4.4}$$

where $\langle . \rangle$ is the inner product operator (Cortes and Vapnik 1995). Computing $\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) \forall i, j$ gives us the kernel Gram matrix $\mathcal{K}(\mathcal{X}, \mathcal{X})$, which its entries are determined by the values $\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$. An example of such $\mathcal{K}(\mathcal{X}, \mathcal{X})$ is a similarity matrix, which describes the pairwise similarity of data points. Typically, a practical similarity kernel can be computed using a Gaussian kernel

$$\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) = exp(-\mathcal{D}(\mathbf{X}_i, \mathbf{X}_j)/\delta),$$

or a polynomial kernel

$$\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) = \mathcal{D}(\mathbf{X}_i, \mathbf{X}_j)^c,$$

where $\mathcal{D}(\mathbf{X}_i, \mathbf{X}_j)$ is the distance (dissimilarity) between a pair of $\{\mathbf{X}_i, \mathbf{X}_j\}$. Also, δ and c are considered as the kernel parameters. According to Section 2.3, the DTW distance measure is a practical choice for $\mathcal{D}(\mathbf{X}_i, \mathbf{X}_j)$ to construct a similarity kernel for motion data.

Therefore, by incorporation Equation 4.4 into Equation 4.2, we can extend it to its kernel-based variation:

$$\min_{\mathbf{\Gamma},\mathbf{D}} \|\Phi(\mathcal{X}) - \Phi(\mathbf{D})\mathbf{\Gamma}\|_F^2 \quad \text{s.t. } \|\vec{\gamma}_i\|_0 < T \ \ \forall i \tag{4.5}$$

in which $\Phi(\mathbf{D})$ is a dictionary that is generally defined in the feature space. In exceptional cases such as (Bahrampour et al. 2015), the kernel functions are explicitly computed based on vectorial inputs, and hence $\Phi(\mathbf{D})$ is traceable in the input space. However, for structured data such as motion, it is difficult to interpret a general $\Phi(\mathbf{D})$ and its constituent entities due to the lack of direct access to the implicit feature space.

As a workaround to the above issue, it is possible to define a dictionary in the feature space in the form of $\Phi(\mathbf{D}) = \Phi(\mathcal{X})\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{N\times c}$ (H. V. Nguyen et al. 2013). This dictionary structure results in the following K-SRC formulation:

$$\min_{\mathbf{\Gamma},\mathbf{U}} \|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 \quad \text{s.t. } \|\vec{\gamma}_i\|_0 < T \ \ \forall i \tag{4.6}$$

Each column of the dictionary matrix $\mathbf{U}$ contains a linear combination of data points in the feature space. Therefore, to its advantage over $\Phi(\mathbf{D})$, the reconstruction term in (4.5) can be rephrased in terms of the Gram matrix $\mathcal{K}(\mathbf{X}, \mathbf{X})$

$$\begin{aligned} &\|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 = \\ &\mathcal{K}(\mathbf{X}, \mathbf{X}) + \mathbf{\Gamma}^\top\mathbf{U}^\top\mathcal{K}(\mathbf{X}, \mathbf{X})\mathbf{U}\mathbf{\Gamma} - 2\mathcal{K}(\mathbf{X}, \mathbf{X})\mathbf{U}\mathbf{\Gamma}, \end{aligned} \tag{4.7}$$

which is traceable in the input space. Furthermore, to optimize the dictionary, we can directly optimize the entries of $\mathbf{U}$. In Section 4.2, I show how we can benefit from this specific structure to efficiently interpret the entities of the dictionary model as well as the sparse encodings of motion data.

Discriminant Sparse Coding

As pointed out before, we are interested in having sparsely encoded vectors that can also carry supervised information about the original motion sequences. In the sparse coding family of algorithms, such a concept is generally addressed by designing discriminative sparse coding frameworks. Considering the label matrix $\mathbf{H}$ (as defined in Section 2.2), discriminative SRC methods focus on designing discriminant objective terms in Equation 4.2. These terms help the SRC framework to learn an efficient dictionary $\mathbf{D}$, based on which the sparse codes $\mathbf{\Gamma}$ can also represent the labeling information of $\mathbf{H}$. Generally, we can categorize discriminative SRC algorithms into disjoint dictionary learning and discriminant-based formulation.

Disjoint Dictionary Learning

In a group of sparse coding works (F. Bach et al. 2008; Sivalingam et al. 2011; Jenatton et al. 2010), the dictionary $\mathbf{D}$ is split into C individual class-specific sub-dictionaries $\mathbf{D} = [\mathbf{D}_1\mathbf{D}_2 \dots \mathbf{D}_C]$. Each $\mathbf{D}_i$ is separately trained to reconstruct only the i-th class of data via assuming no existing correlation between different classes. At first sight, these sub-dictionaries appear highly interpretable, such that each dictionary atom $\vec{d}_i$ can be associated with only one class of data. Nevertheless, these methods confront problems when different data classes are close to each other, and some data points from one class can also be expressed by dictionary atoms related to another class. Such issues reduce the discriminative quality of their representation. Furthermore, they cannot model sub-class structures, especially if there is an overlap between the classes. Therefore, as a typical observation, a test input $\vec{z}$ would be reconstructed by selecting atoms from multiple sub-dictionaries $\mathbf{D}_i$ related to multiple classes. Such behavior damages the semantic interpretation of the corresponding $\vec{\gamma}$, and more often, it includes noisy supervised information in sparse encoding. As the workarounds, some researchers tried to mitigate the above limitation by learning an additional dictionary module $\tilde{\mathbf{D}}$, which is shared among all of the classes to take care of the class overlaps, or via making the disjoint dictionaries orthogonal to each other in order to reduce their coherency (Ramirez, Sprechmann, and Sapiro 2010; N. Zhou et al. 2012; S. Kong and D. Wang 2012). As another improved strategy, (M. Yang, Lei Zhang, et al. 2011; Vu and Monga 2017) train all sub-dictionaries together as one unique problem in favor of the reconstruction and discrimination purposes together. However, the main focus of all the mentioned methods is improving the classification accuracy without any effort toward the interpretability of the encoding frameworks. In addition, these frameworks require to have the sub-dictionary sizes manually defined in advance. As another limitation, such a requirement relies on an unrealistic assumption that all the classes have similar local and global distributions.

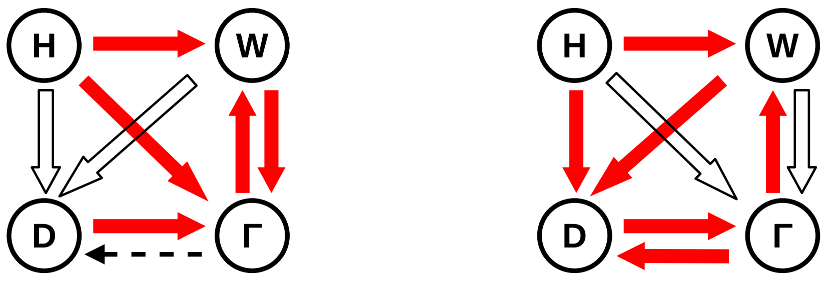
Discriminant-based Formulation

Another branch of discriminative sparse coding algorithms such as (Mairal, F. Bach, and Ponce 2012; Z. Jiang, Z. Lin, and Davis 2013; W. Liu et al. 2015; Quan, Y. Xu, et al. 2016) focus on adding a particular objective(s) to the optimization scheme of Equation 4.2, which also encodes the label information of the input $\vec{x}$ into $\vec{\gamma}$. One particular example of such frameworks is the LC-KSVD algorithm (Z. Jiang, Z. Lin, and Davis 2013) and its variations as

$$\begin{array}{ll} \min\limits_{\mathbf{\Gamma},\mathbf{D},\mathbf{W}} & \|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 + \alpha\|\mathbf{H} - \mathbf{W}\mathbf{\Gamma}\|_F^2 + \lambda\|\mathbf{W}\|_F^2 \\ \text{s.t.} & \|\vec{\gamma}_i\|_0 < T \ \forall i, \end{array} \tag{4.8}$$

where $\mathbf{W}$ is the discriminant's matrix of parameters, and α, λ are the trade-off scalar and the regularization factor, respectively. Here, the term $\|\mathbf{H} - \mathbf{W}\mathbf{\Gamma}\|_F^2$ is a linear discriminant that tries to map the encodings $\mathbf{\Gamma}$ to the labeling space spanned by columns of $\mathbf{H}$. In other words, the resulting discriminant model is the combination of two mappings, $\mathbf{D} : \vec{x} \in \mathbf{R}^d \mapsto \vec{\gamma} \in \mathbf{R}^k$ and $\mathbf{W} : \vec{\gamma} \in \mathbf{R}^k \mapsto \vec{h} \in \mathbf{R}^C$. Such a mapping chain assumes an optimal $\mathbf{D}^*$ exists that makes the classes linearly separable in $span(\mathbf{D}^*)$.

One specific issue that I want to address in the above-mentioned frameworks is the lack of an integrated optimization scheme in them. As mentioned in Section 4.1, such frameworks' parameters are typically optimized in an alternating scheme similar to Equation 4.3. Therefore, when we optimize one parameter and fix the others, parts



(a) LC-KSVD (b) Task-driven

Figure 4.2: The influence of parameters on each other during the optimization loop of discriminative sparse coding. A red arrow toward a specific parameter indicates which other parameters influence it during its optimization. Accordingly, the dashed and empty arrows show partial and missing links, respectively. LC-KSVD: The values of **W** and **H** have no direct effect on the optimization of **D**. Task-Driven: The optimization of **Γ** is not directly affected by the value of **W** and **H**.

of the compound objective function that are constant w.r.t. that parameter are set to zero. Under this elimination, the parameters cannot be optimized according to a unified objective, which can either disturb the convergence path or the quality of the convergence point in the optimization loop. For example, when optimizing Equation 4.8 w.r.t. **D**, the discriminant $\|\mathbf{H} - \mathbf{W}\mathbf{\Gamma}\|_F^2$ and the term $\|\mathbf{W}\|_F^2$ are removed as constants terms. However, those terms are functions of **W** and **Γ**. Hence, the current value of **W** does not have a direct effect in the optimization of matrix **D**. Nevertheless, it is expected from such a framework to train **D** by also considering the role of the **W** and **H** in the whole framework. This issue is visualized in Figure 4.2-left, where the arrows show the existing effect of parameters on each other during the optimization phase. According to the figure, **D** is not directly influenced by **W** and **H** in the optimization loop, while it is partially influenced by the value of **Γ**. Although the task-driven algorithm (Mairal, F. Bach, and Ponce 2012) tries to optimize **D** directly coupled with the values of **W** and **H**, the updating process of **Γ** still occurs in a disjointed framework (Fig.4.2-right). In contrast to the mentioned discriminative sparse coding frameworks, the proposed methods in this chapter have an integrated optimization scheme.

Consistency Between Training and Recall

Considering the optimization scheme in Equation 4.8 as a typical form of discriminative sparse coding, the discriminative quality of the learned $\mathbf{\Gamma}^*$ and the mapping $\mathbf{D} : \mathbf{X} \mapsto \mathbf{\Gamma}^*$ is greatly influenced by the role of **H** in the framework. However, **H** is not available in the recall phase, which is related to the reconstruction of the test data $\vec{z}$ similar to

$$\min_{\vec{\gamma}} \|\vec{z} - \mathbf{D}\vec{\gamma}\|_2^2 \quad \text{s.t.} \|\vec{\gamma}\|_0 < T \tag{4.9}$$

Therefore, due to the redundancy of the learned **D**, it is highly probable that reconstructing $\vec{z}$ using only Equation 4.9 results in $\vec{z} \mapsto \vec{\gamma}$ such that $\vec{\gamma} \notin \text{span}\{\mathbf{\Gamma}^*\}$ even if we have $\vec{z} \in \text{span}\{\mathbf{X}\}$ which reduces the discriminative quality of $\vec{\gamma}$. To my knowledge, the only discriminative sparse coding algorithm that merely aims for such consistency is the Fisher discriminative sparse coding (M. Yang, Lei Zhang, et al. 2011). That algorithm aims to encode the test data $\vec{z}$ by a sparse vector close enough to $\bar{\vec{\gamma}} = \frac{1}{N}\sum_{i=1}^{N} \vec{\gamma}_i$, which is the representative vector for all encoded training data from the presumed class of $\vec{z}$. This method tries all the possible classes for $\vec{z}$ to find the best fitting solution. However, in contrast to its base assumption, it is convenient for an SRC model to obtain distributed clusters of sparse codes, even though they are related to one class. In Section 4.3, I propose a more consistent sparse coding framework, which incorporates the supervised training information also in the encoding of test data. Hence, its recall phase provides a more efficient discriminant mapping for the test data.

Sparse Subspace Clustering

For vectorial data with a matrix of training samples $\mathbf{X} = [\vec{x}_i]_{i=1}^{N}$, we can assume $\mathbf{X}$ lies in the union of n linear subspaces $\cup_{l=1}^{n} \mathcal{S}_l$ with corresponding dimensions of $\{m_l\}_{l=1}^{n}$. Subspace clustering tries to categorize data into separate clusters such that each cluster j contains samples lying in one individual subspace $\mathcal{S}_j$. By assuming that each subspace holds a semantic similarity between its constituent samples, each data point $\vec{x}_i \in \mathcal{S}_j$ can be represented by $\bar{\mathbf{X}}_i$ as other samples in $\mathcal{S}_j$ with a linear combination $\vec{x}_i \approx \bar{\mathbf{X}}_i \vec{\gamma}_i$. Focusing on the sparseness of the coding vectors $\vec{\gamma}_i$, subspace sparse clustering (SSC) (Elhamifar and Rene Vidal 2013) formulates the above concept as

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_0 \quad s.t. \;\; \mathbf{X} = \mathbf{X}\mathbf{\Gamma}, \gamma_{ii} = 0 \;, \; \forall i \tag{4.10}$$

where the constraint on γ_{ii} prevents the data points from self-reconstrution as the trivial solution to Equation 4.10.

An SSC model relies on the assumption that applying a sparsity prior to the encoding vector $\vec{\gamma}$ represents $\vec{x}$ by using only (or mostly) data points semantically similar to $\vec{x}$, which are expected to lie in the same subspace. Therefore, computing an affinity matrix

$$\mathbf{A} = |\mathbf{\Gamma}|^{\top} + |\mathbf{\Gamma}| \tag{4.11}$$

signifies the pairwise similarities of data points in the input space, based on which graph-based methods such as spectral clustering can reveal the underlying clusters.

The SSC problem in Equation 4.10 is NP-hard to solve in its original format (Elhamifar and Rene Vidal 2013). As a solution, $\|.\|_0$ can be relaxed into other norms. For instance, (Elhamifar and Rene Vidal 2013; Patel and René Vidal 2014; Bian, F. Li, and X. Ning 2016; S. Gao, I. W.-h. Tsang, and L.-t. Chia 2012) use the l_1-norm to achieve sparse $\mathbf{\Gamma}$, while (You, D. Robinson, and René Vidal 2016) aims for the approximate solution of Equation 4.10 with having $\|\vec{\gamma}_i\|_0 \leq T$. Another group of SSC methods (René Vidal and Favaro 2014; S. Xiao et al. 2016; Guangcan Liu et al. 2013; Zhuang et al. 2012) focuses on shrinking the nuclear norm $\|\mathbf{\Gamma}\|_*$ and making $\mathbf{\Gamma}$ low-rank to better represent the global structure of $\mathbf{X}$. Among SSC algorithms, (Elhamifar and Rene Vidal 2013; Patel and René Vidal 2014) enforced $\mathbf{\Gamma}$ to provide affine representations by using the constraint $\mathbf{\Gamma}^{\top}\vec{\mathbf{1}} = \vec{\mathbf{1}}$ based on the idea of having the data points lying on an affine combination of subspaces. Despite continuous improvements in clustering results of aforementioned SSC methods, there is no direct connection between the quality of the encoding model and the subsequent clustering task. Consequently, they suffer from performance variation across different datasets and high sensitivity of their results to the parameters choice.

To mitigate the above issue, another group of algorithms called Laplacian sparse coding encourage the sparse coefficient vectors $\vec{\gamma}_i$ related to each cluster to be as similar as possible (S. Gao, I. W.-h. Tsang, and L.-t. Chia 2012; Y. Yang, Zhangyang Wang, et al.

2014). In their SSC formulation (Equation 4.12) they employ a similarity matrix **W** in which each w_{ij} measures the pairwise similarity between a pair $(\vec{x}_i, \vec{x}_j)$:

$$\min_{\mathbf{\Gamma}} \quad \|\mathbf{X} - \mathbf{X}\mathbf{\Gamma}\|_F^2 + \lambda\|\mathbf{\Gamma}\|_1 + \tfrac{1}{2}\sum_{i,j} w_{ij}\|\vec{\gamma}_i - \vec{\gamma}_j\|_2^2 \qquad s.t.\ \gamma_{ii} = 0\ ,\ \forall i. \tag{4.12}$$

Nevertheless, optimization frameworks like Equation 4.12 suffer from two main issues:

1. Columns of **Γ** are forced to become similar to each other while the similarity matrix **W** is used as the weighting coefficients. Hence, due to this heavy bias, the sparse codes $\vec{\gamma}_i$ obtain a distribution similar to the neighborhoods in **W**. Consequently, their performance is at best comparable to kernel-based clustering with direct use of **W** as the kernel information.
2. Although Equation 4.12 tries to decrease the intra-cluster distances, the inter-cluster structure of data is ignored in such frameworks. However, typically both of these terms have to be adopted when focusing on the separability of clusters.

Investigating the interpretability of SSC algorithms, some works benefit from the non-negative formulation of the SSC framework (X. Li, Cui, and Dong 2017; S. Xiao et al. 2016; Zhuang et al. 2012). These methods impose non-negative constraints on the entries of **Γ**, which enforce it to represent each $\vec{x}$ with other data points that are semantically more similar to $\vec{x}$. Such non-negative formulation has the potential to considerably reduce the overlapping columns of **Γ** w.r.t. the underlying subspaces. Nevertheless, these methods still suffer from the above-mentioned issues regarding their optimization formulations. In addition, such representation more often leads to categorizing the dataset into many local, distinct sub-clusters within each subspace. Although these sub-clusters are still similar, they do not have corresponding connections in the affinity matrix **A** (Equation 4.11). Those links correspond to redundancies that are drastically removed due to the significantly sparse form of the columns in **Γ**.

It is possible to extend most of the mentioned SSC algorithms to their kernel-based sparse subspace coding (K-SSC) variation by using the dot-product rule (Equation 4.4) in their mathematical formulation (Bian, F. Li, and X. Ning 2016; S. Xiao et al. 2016; Patel and René Vidal 2014). These K-SSC algorithms make the subspace clustering concept applicable to structured data such as motion sequences or video. Benefiting from the hidden information in a similarity kernel, K-SSC can reveal the underlying semantic relations between the data points without the need to employ any supervised information. In Section 4.4, I propose a novel SSC algorithm (and its kernel-based extension for motion sequences), which mitigates the addressed limitations of SSC frameworks, especially for interpretation of the resulted encoding. My post-processing technique can also contribute to non-negative SSC methods to improve their latent representations by reviving their broken data links.

In the following sections of this chapter, I discuss my non-negative algorithms for the interpretable sparse encoding of motion data. I propose my non-negative kernel sparse coding frameworks in the next section.

4.2 NON-NEGATIVE KERNEL SPARSE CODING

As discussed before, we can obtain an embedding for motion data from the sequential space to the vector space by means of a kernel sparse coding framework. A K-SRC model gives us a sparse encapsulated vector of representation for a motion sequence according to its semantic similarity to other sequences in a mocap dataset. In Chapter 3, we learned that the DTW is an intuitive distance to construct such a similarity-based kernel. However, in the context of this dissertation, we are interested in achieving embeddings that are interpretable in terms of the constituent elements of the resulting encoded vector. As a contribution of this section, I extend the existing K-SRC framework to its non-negative variation, which adds the above-desired property to the resulting sparse embedding. More specifically, the non-negative framework signifies meaningful connections between different elements of its model (dictionary and the sparse code) and semantically relevant information (motion sequences). As a result, the proposed method of this section can encode a motion sequence into a sparse vector, in which its non-zero entries can be easier interpreted and understood by a practitioner.

Proposed Non-negative Framework

As discussed in Section 4.1, we can obtain a kernel-based representation of a motion dataset by computing the pairwise similarity between its sequences. As a practical choice, we can use the DTW distance to derive the similarity-based kernel $\mathcal{K}(\mathbf{X}, \mathbf{X})$, which holds the dot-product rule of Equation 4.4. Such representation particularly suits the variations of motion sequences in the temporal length. Using $\mathcal{K}(\mathbf{X}, \mathbf{X})$, we can construct the dictionary matrix $\Phi(\mathbf{D}) = \Phi(\mathcal{X})\mathbf{U}$ as suggested by (H. V. Nguyen et al. 2013). Although the dictionary matrix $\Phi(\mathbf{D})$ is defined in the feature space, we can update the dictionary elements by adjusting the corresponding entries of matrix $\mathbf{U} \in \mathbb{R}^{N \times k}$ in the input space. More precisely, each vector $\vec{u}_i$ constructs a corresponding dictionary atom $\Phi(\vec{d}_i)$ by linear combinations of the training samples from $\mathcal{X}$ in the feature space. Such structure gives us the reconstruction loss term $\|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$ similar to Equation 4.7.

Proposition 4.1. *If $rank(\Phi(\mathcal{X})) < N$, there exist $\mathbf{U}^* \in \mathbb{R}^{N \times k}, \mathbf{\Gamma}^* \in \mathbb{R}^{k \times N}$ $k < N$ such that $\Phi(\mathcal{X})$ can be reconstructed as $\Phi(\mathcal{X}) = \Phi(\mathcal{X})\mathbf{U}^*\mathbf{\Gamma}^*$.*

Proof. Refer to Appendix A.4. □

This proposition supports the rationale behind choosing the above dictionary structure. In particular, Proposition 4.1 tells us that via properly defining $\mathbf{U}$ as the dictionary matrix, each $\Phi(\vec{x}_i)$ can be optimally reconstructed by the weighted contributions from the training set $\mathbf{\Gamma}$. Also, by considering that the test data $\Phi(\vec{z})$ is sufficiently similar to the training samples such that $\Phi(\vec{z}) \in span\{\Phi(\mathcal{X})\}$, we can derive the same conclusion for test data $\Phi(\mathbf{Z})$.

In accordance with the **RQ2** research question in Chapter 1, we want to obtain a semantically interpretable encoding for motion sequences. This objective requires $\mathbf{U}$ and $\mathbf{\Gamma}$ to be consequently interpretable from that perspective. To be more specific, we would like to have a dictionary $\Phi(\mathcal{X})\mathbf{U}$ such that each of its columns carries characteristics of a particular motion type. Therefore, a dictionary atom that is a positive linear combination of input data ($u_{ij} \geq 0$) naturally selects semantically similar sequences from $\mathcal{X}$. Relevantly, we need to learn a dictionary such that its representative matrix $\mathbf{U}$ uses as few elements from $\Phi(\mathbf{Y})$ as possible (small $\|\vec{u}_i\|_1$). In other words, it leads to using fewer signals from

$\mathcal{X}$ for representing the motion dataset. Such structure results in a sparse, interpretable dictionary, each of which atoms can be linked to one type of motion sequence.

Having such a dictionary, the encoding vector $\vec{\gamma}$ links each motion data $\mathbf{X}$ to one or more meaningful motion-based dictionary atoms. Therefore, if non-zero entries of $\vec{\gamma}$ reconstruct $\mathbf{X}$ using similar elements of $\Phi(\mathcal{X})\mathbf{U}$, the meaningful content of $\vec{\gamma}$ (and consequently $\mathbf{X}$) can be more easily interpreted. To enforce such potential, I formulate the K-SRC problem such that the motion signal would be encoded with a non-negative coefficient vector as $\vec{\gamma}_{ij} \geq 0$.

Therefore, based on the above structural proposals, I extend the formulation of Equation 4.6 to the following novel non-negative kernel sparse coding (NNKSC) framework:

$$\begin{aligned} \min_{\mathbf{\Gamma},\mathbf{U}} \quad & \|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 + \lambda\|\mathbf{U}\|_1^2 \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 \leq T, \quad u_{ij}, \gamma_{ij} \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{4.13}$$

In Equation 4.13, parameters λ and T are the scalars that enforce sparseness on the dictionary matrix $\mathbf{U}$ and the encoding matrix $\mathbf{\Gamma}$, respectively.

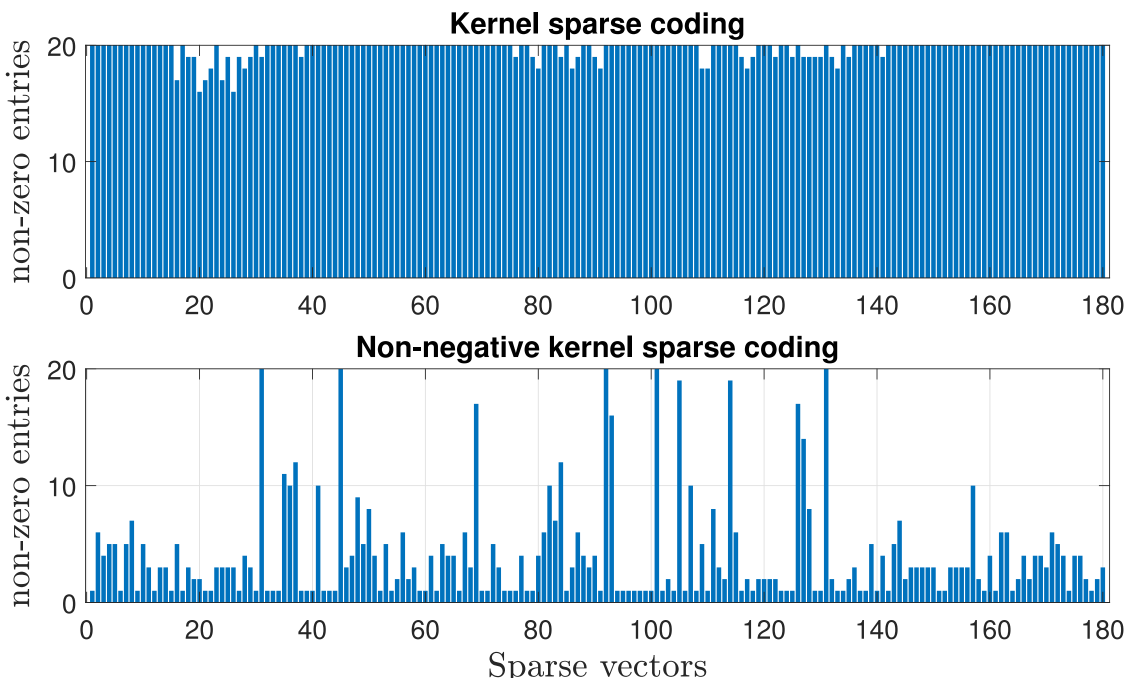
To observe the direct effect of the non-negative constraint in NNKSC , we can compare it with the kernel KSVD algorithm (K-KSVD) from (H. V. Nguyen et al. 2013). K-KSVD has the same optimization formulation as in Equation 4.13 except the non-negativity constrains on $(\mathbf{\Gamma}, \mathbf{U})$ and the sparseness objective $\|\mathbf{U}\|_1$ on the dictionary. As depicted in Figure 4.3, these extra terms in the optimization scheme of NNKSC result in a noticeable sparseness and compactness of its encoded vectors in $\mathbf{\Gamma}$ (Figure 4.3-a). Primarily, the non-negativity constraint prevents any redundant combination of columns of $\Phi(\mathcal{X})\mathbf{U}$ for the reconstruction of a given $\vec{x}$ in the feature space, leading to a considerably compact $\vec{\gamma}$.

On the other hand, the K-KSVD method uses almost all possible dictionary resources for its reconstruction. That is why in Figure 4.3-a, almost all encoded vectors $\vec{\gamma}$ used their maximum allowed budget from $\mathbf{U}$ (here is $T = 20$). In the same way, each column of the trained nA in K-KSVD generously uses contributions from training samples $\Phi(\mathcal{X})$ to shape dictionary atoms.

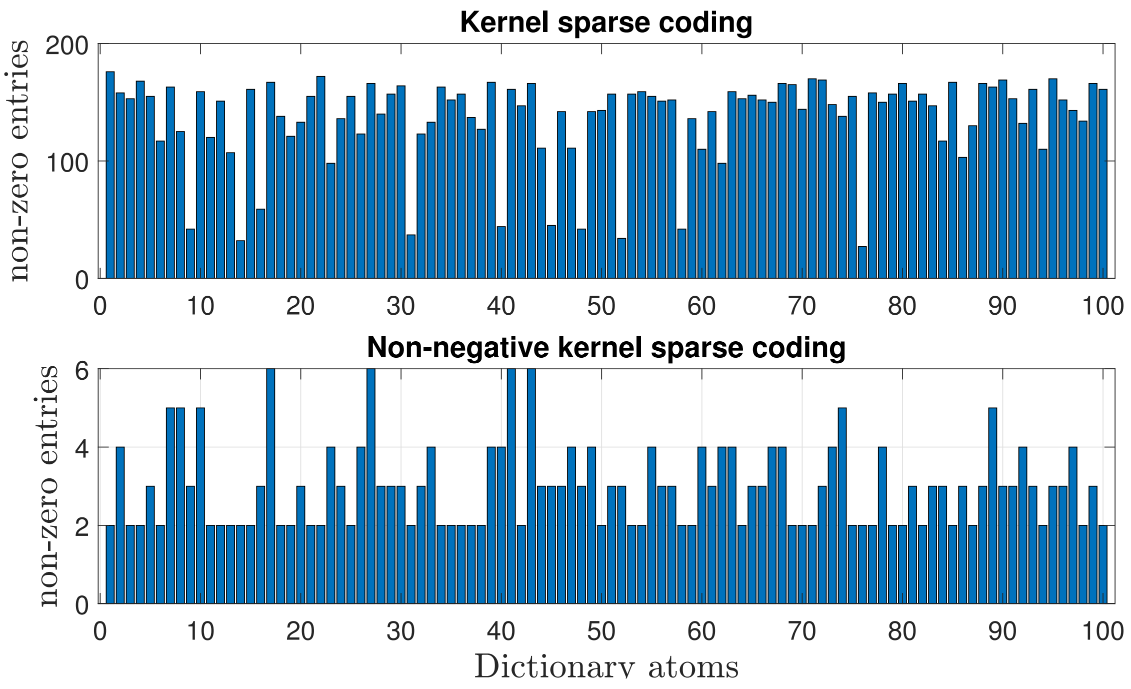
Considering the number of connections between each $\vec{\gamma}$ and the training samples in K-KSVD makes it difficult (or almost not feasible) to interpret the resulting encoding in most columns of $\mathbf{\Gamma}$. However, the considerably sparse form of $\mathbf{\Gamma}$ in NNKSC lets us semantically relate each $\Phi(\mathbf{X}_i)$ to the few dictionary atoms from which it is reconstructed. We can have a similar interpretation for the resulting sparse columns of $\mathbf{U}$, which form a connection between each dictionary atom and its few constructing training sources. In the ideal case, we prefer to have each atom $\Phi(\mathcal{X})\vec{u}_i$ to be related to one type of motion (one class from $\mathbf{X}$). In the experiment part of this chapter (section 4.5), I numerically evaluate and measure the above interpretability concept for my proposed methods using mathematical evaluating measures.

Optimization Framework

In order to solve the optimization problem of Equation 4.13, I use an alternating optimization scheme similar to Equation 4.3. In the optimization loop of NNKSC , each of the matrices $(\mathbf{U}, \mathbf{\Gamma})$ becomes updated in a separate step while the other one is fixed (Algorithm 4.3). In the following, I discuss the specific optimization steps for updating the dictionary and the sparse codes, as well as the proposed methods to approach individual optimization sub-problems.



(a) The number of non-zero entries in columns of $\mathbf{\Gamma}$.



(b) The number of non-zero entries in columns of $\mathbf{U}$.

Figure 4.3: Comparing the sparseness of NNKSC to that of K-KSVD for the UTKinect dataset. (a): NNKSC has more zero entries in its nX (a more sparse encoding) than K-KSVD. (b): Columns of the dictionary matrix $\mathbf{U}$ for NNKSC have fewer contributions from training data compared to $\mathbf{U}$ of K-KSVD. Both models trained up to the same reconstruction error $\|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$.

Algorithm 4.1 The NN-KOMP algorithm: finds an approximate solution to Equation 4.14 as a non-negative sparse encoding of a data sample in the feature space under the cardinality constraint.

```
1: Input: Dictionary matrix $\mathbf{U}$, sparseness limit $T$, kernel matrix $\mathcal{K}(\mathbf{X},\mathbf{X})$
2: Output: Approximate solution $\vec{\gamma}$ to
       $\arg\min_{\vec{\gamma}} \|\Phi(\vec{z}) - \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}\|_2^2$
       s.t. $\gamma_j \geq 0\ \forall j, \quad \|\vec{\gamma}\|_0 \leq T$
3: Initialize variables as $\vec{\gamma} = 0,\ I = \emptyset$.
4: while $\|\vec{\gamma}\|_0 < T$ do
5:     $\tau_i = [\mathcal{K}(\vec{z},\mathbf{X}) - \vec{\gamma}^\top \mathbf{U}_I^\top \mathcal{K}(\mathbf{X},\mathbf{X})]\vec{u}_i,\ \forall i \notin I$
6:     $i_{max} = \arg\max_i |\tau_i|,\ \forall i \notin I$
7:     $I = I \cup i_{max}$
8:     $\vec{\gamma} = \arg\min_{\vec{\gamma}} \|\Phi(\vec{z}) - \Phi(\mathcal{X})\mathbf{U}_I\vec{\gamma}\|_2^2$ s.t $\gamma_j \geq 0,\ \forall j$ using K-NNLS (Algorithm A.1).
9: end while
```

Updating the Matrix of Sparse Codes $\mathbf{\Gamma}$

For the sparse coding part of Equation 4.13, I estimate each individual non-negative sparse vector $\vec{\gamma}_i\ \forall i = 1, \ldots, N$ based on the current solution of the dictionary matrix $\mathbf{U}$ using Equation 4.14. This optimization problem focuses on the non-negative reconstructing of each motion signal $\mathbf{X}_i$ in the feature space using non-negative contributions from other training motion samples in $\mathcal{X}$.

$$\begin{aligned} \vec{\gamma}_i = \quad & \arg\min_{\vec{\gamma}_i} \quad \|\Phi(\mathbf{X}_i) - \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}\|_2^2 \\ & \text{s.t.} \quad \vec{\gamma}_i \geq 0, \quad \|\vec{\gamma}\|_0 \leq T \end{aligned} \tag{4.14}$$

To solve Equation 4.14, I propose the NN-KOMP algorithm (Algorithm 4.1) as the non-negative extension of the KOMP algorithm from (H. V. Nguyen et al. 2013). In step 8 of the algorithm, the non-negative vector $\vec{\gamma}_I$ corresponding to the currently selected dictionary atoms $\mathbf{U}_I$ is estimated by Algorithm A.1 as the kernel-based non-negative least square method (K-NNLS). I propose the K-NNLS algorithm by kernelizing the active set fast non-negative least square optimization method (FNNLS) from (Bro and De Jong 1997). According to Algorithm 4.1, NN-KOMP takes $\mathcal{O}((5T + T^2)\frac{N}{2} + T^{4.3})$ steps to find a solution for each $\vec{\gamma}_i$ from Equation 4.14 and also requires $\mathcal{O}(kN^2 + k^2N)$ computation operations to update all $\mathbf{\Gamma}$.

Updating the Dictionary Matrix $\mathbf{U}$

As the second part of my NNKSC algorithm, I want to find the best dictionary $\Phi(\mathcal{X})\mathbf{U}$ which minimizes (Equation 4.13) while using the obtained coefficients $\mathbf{\Gamma}$ as the output of NN-KOMP in the previous section. Based on (H. V. Nguyen et al. 2013), the error function $\|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$ can be reformulated as:

$$\|\Phi(\mathcal{X})\mathbf{E}_i - \Phi(\mathcal{X})\vec{u}_i\vec{\gamma}^i\|_F^2, \qquad \mathbf{E}_i = (I - \sum_{j\neq i}\vec{u}_j\vec{\gamma}^j). \tag{4.15}$$

In Equation 4.15, $\Phi(\mathcal{X})\mathbf{E}_i$ is the reconstruction error using all the dictionary columns except $\vec{u}_i$, and $\vec{\gamma}^i$ is the corresponding coefficients in the i-th row of the updated $\mathbf{\Gamma}$.

Algorithm 4.2 The NN-KFISTA algorithm: updates a dictionary atom based on Equation 4.16 in the presence of a non-negativity constraint.

```
1: Input: function $f(\vec{u}, \mathcal{K}(\mathbf{X},\mathbf{X}), \mathbf{E})$ from Equation 4.17, sparseness weight $\lambda$
2: Output: non-negative sparse dictionary atom $\vec{u}$ as the approximate solution to (Equation 4.16)
3: Initialize variables as $m = 0,\ t = 1,\ 0 < \eta < 1,\ \alpha \geq 0,\ \delta$
4: for $m \in \mathbb{N}$ do
5:     $i = \arg\min_i i$
       s.t. $\alpha_m = \eta^i \alpha_{m-1}$
            $\vec{u}^{m+1} = \tau_{\alpha_m\lambda}(\vec{u}^m - \alpha_m \nabla f(\vec{u}^m))$
            $f(\vec{u}^{m+1}) - f(\vec{u}^m) > (\vec{u}^{m+1} - \vec{u}^m)^\top \nabla f(\vec{u}^m) - \frac{\|\vec{u}^{m+1} - \vec{u}^m\|_2^2}{2\alpha_m}$
            $i \in \mathbb{N}$
6:     if $f(\vec{u}^{m+1}) < \delta$ then
7:         Stop the algorithm.
8:     else
9:         $t_{m+1} = (1 + \sqrt{1 + 4t_m^2})/2.$
10:        $\vec{u}^{m+1} \leftarrow \vec{u}^{m+1} + (\vec{u}^{m+1} - \vec{u}^m)(t_m - 1)/t_{m+1}.$
11:    end if
12: end for
```

Therefore, dictionary columns can be updated through solving Equation 4.15 for $\{\vec{u}_i\}_{i=1}^k$. As an essential constraint, we have to take into account that the optimal dictionary should be used along with non-negative coefficients $\mathbf{\Gamma}$. According to Equation 4.15, we are looking for the solution of the following optimization problem:

$$\begin{aligned} \vec{u}_i = \ & \arg\min_{\vec{u}_i} \ \|\Phi(\mathcal{X})\mathbf{E}_i - \Phi(\mathcal{X})\vec{u}_i\vec{\gamma}^i\|_F^2 + \lambda\|\vec{u}_i\|_1 \\ & \text{s.t.} \quad \vec{u}_i \geq 0 \end{aligned} \tag{4.16}$$

In order to solve the problem of Equation 4.16, I propose the non-negative kernel FISTA algorithm (NN-KFISTA), which is a combination of the projected gradient technique (C.-J. Lin 2007) and the first order Shrinkage-Thresholding method (Beck and Teboulle 2009). In steps 5 and 6 of Algorithm 4.2, values of $f(\vec{u}_i)$, $\nabla f(\vec{u}_i)$, and entries of $\tau_l(\vec{u}_i)$ are calculated as:

$$\begin{aligned} & f(\vec{u}_i) = \|\Phi(\mathcal{X})\mathbf{E}_i - \Phi(\mathcal{X})\vec{u}_i\vec{\gamma}^i\|_F^2 = \text{tr}[(\mathbf{E}_i - \vec{u}_i\vec{\gamma}^i)^\top \mathcal{K}(\mathbf{X},\mathbf{X})(\mathbf{E}_i - \vec{u}_i\vec{\gamma}^i)] \\ & \nabla f(\vec{u}_i) = -2\mathcal{K}(\mathbf{X},\mathbf{X})(\mathbf{E}_i - \vec{u}_i\vec{\gamma}^i)\vec{\gamma}^{i\top} \\ & \tau_l(u_{ij}) = (u_{ij} - l)(sign(u_{ij} - l) + 1)/2, \quad \forall j, \end{aligned} \tag{4.17}$$

where tr(.) denotes the trace operator. The convergence analysis of the NN-KFISTA algorithm mainly follows the same principles as presented in (Beck and Teboulle 2009) for the FISTA optimization method, in which it is proven that the FISTA algorithm (and consequently NN-KFISTA) has quadratic convergence. Regarding the run-time complexity of Algorithm 4.2, when excluding the precomputation parts, NN-KFISTA in the worst case needs $\mathcal{O}(m[8N^2 + (5 + 7s)N + 3s])$ steps to converge, where m and s denote the total iteration for outer-loop and the step 5 of the algorithm, respectively. Additionally, NN-KFISTA requires $\mathcal{O}(2kN^2)$ precomputation to update all the columns in $\mathbf{U}$.

Note: When computing $\mathbf{E}_i$ to update $\vec{u}_i$, matrix $\mathbf{U}$ should be used with its recently updated columns to improve the optimization loop's convergence speed. For example,

Algorithm 4.3 The NNKSC algorithm: learns a non-negative dictionary and sparse code matrices as the approximate solution to the sparse coding problem in Equation 4.13. The algorithm applies cardinality constraint on the sparse encoding vectors.

```
1: **Input:** Sparseness parameters $(T, \lambda)$, Kernel matrix $\mathcal{K}(\mathbf{X}, \mathbf{X})$, stopping threshold $\delta$.
2: **Output:** Approximate solutions $(\mathbf{\Gamma}, \mathbf{U})$ to Equation 4.13
3: Initialization: $\mathbf{U} \leftarrow$ a random matrix with single-entry binary columns.
4: while [objective of Equation 4.13] $> \delta$ do
5:     Updating $\mathbf{\Gamma}$ based on Equation 4.14 using NN-KOMP (Algorithm 4.1).
6:     Updating $\mathbf{U}$ based on Equation 4.15 using NN-KFISTA (Algorithm 4.2).
7: end while
```

when successively updating columns of $\mathbf{U}$ in Step 6 of Algorithm 4.3, matrix $\mathbf{E}_i$ is computed based on the current update of $\mathbf{U}$ as:

$$\mathbf{U} = \{\vec{u}_1^t, \vec{u}_2^t, ..., \vec{u}_{i-1}^{(t-1)}, \vec{u}_i^{(t-1)}, ..., \vec{u}_k^{(t-1)}\},$$

where $\vec{u}_i^t$ is the value of $\vec{u}_i$ at iteration t of NNKSC . Moreover, directly after updating each $\vec{u}_i$ via the NN-KFISTA algorithm, it should be normalized as $\vec{u}_i \leftarrow \frac{\vec{u}_i}{\|\Phi(\mathcal{X})\vec{u}_i\|_2}$ to prevent degeneracy during the optimization loop. This step applies the constraint $\|\Phi(\mathcal{X})\vec{u}_i\|_2^2 = 1$ as a bound on l_2-norm of the dictionary columns, which is a typical step to prevent the solution of Equation 4.13 from becoming degenerated (Michael Elad and Michal Aharon 2006).

To sum up, the optimization loop of the NNKSC algorithm consists of solving the two main optimization problems (Equations 4.14 and 4.15) until a convergence criterion is reached (Algorithm 4.3).

Label Consistent NNKSC Extension

As discussed in Chapter 1, another research goal is to enrich the resulting sparse motion embedding with supervised information, i.e., a sparse $\vec{\gamma}$ should also encode the labeling information about the input motion sequence $\mathbf{X}$. In addition, by aiming for interpretation of the resulted embedding, we want to learn a $\vec{\gamma}$ that can clearly relate $\mathbf{X}$ to other sequences of its type. As already discussed in the previous subsection, the specific structure of the dictionary and the sparse codes in the proposed non-negative K-SRC framework (Equation 4.13) have the required basis to achieve the above. Nevertheless, the common inter-class overlaps in real data results in the encoding of some data samples by columns of $\mathbf{D}$ (or $\Phi(\mathcal{X})\mathbf{U}$) that belong to a different data class. Therefore, it is required to add discriminative characteristics to the structure of this framework.

Accordingly, I extend my NNKSC algorithm to appropriate discriminative frameworks, which incorporate the supervised information in order to project semantic similarities of the data (labeling information $\mathbf{H}$) to the resulting embedded sparse vectors $\mathbf{\Gamma}$.

One straightforward approach to obtain the above goal is to use the idea of "Label Consistent SC" from (Z. Jiang, Z. Lin, and Davis 2013) and apply it to our non-negative model NNKSC . This idea was already kernelized in (Z. Chen et al. 2015) for the K-KSVD algorithm. To that aim, I extend the optimization problem of Equation 4.13 using the label matrix **H** related to training data:

$$\begin{aligned} \min_{\mathbf{\Gamma},\mathbf{U}} \quad & \|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 + \alpha\|\mathbf{H} - \mathbf{H}\mathbf{U}\mathbf{\Gamma}\|_F^2 + \lambda\|\mathbf{U}\|_1^2 \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 \leq T, \ \forall i = 1,\dots,N, \quad u_{ij}, \gamma_{ij} \in \mathbb{R}_{\geq 0}, \end{aligned} \tag{4.18}$$

where the parameter α is chosen with a trade-off between the reconstruction error and the classification accuracy. Denoting the augmented kernel matrix as $\widetilde{\mathcal{K}}(\mathbf{X}_i, \mathbf{X}_j) = \mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) + \alpha\langle \vec{h}_i, \vec{h}_j \rangle$, the optimization problem in Equation 4.18 can be solved by the proposed Algorithm 4.3.

After training the dictionary matrix **U**, a test data **Z** can be encoded by applying NN-KOMP (Algorithm 4.1) on Equation 4.14. Afterward, the resulting sparse code $\vec{\gamma}$ would be used to determine the label of **Z** as

$$l = \arg\min_i |1 - \vec{h}^i \mathbf{U}\vec{\gamma}|, \tag{4.19}$$

where $\vec{h}^i$ denotes the i-th row of **H**.

In the experiments (Section 4.5), we observe that the LC-NNKSC extension benefits from both the favorable compact encoding of NNKSC and the class-based interpretability of the resulting model. Nevertheless, the LC-NNKSC algorithm still lacks consistency between its training and recall phases as a common issue of discriminative sparse coding frameworks, which was pointed out in Section 4.1. In the next section, I propose a more robust extension of the NNKSC to a discriminative framework, which mitigates the above concern.

4.3 CONFIDENCE BASED KERNEL SPARSE CODING

In Section 4.2, the designed NNKSC algorithm can encode motion sequences to sparse interpretable encoded vectors. Also, LC-NNKSC can extend that algorithm to a discriminative framework, which provides class-based interpretability for the encoding. Nevertheless, as pointed out in the previous section, the LC-NNKSC framework, similar to other discriminative SRC algorithms, suffers from inconsistency between its training and test models. In this section, I propose a novel confident K-SRC and dictionary learning algorithm (CKSC), which employs the supervised information in both of the training and recall models. Such formulation provides more consistency in the CKSC framework compared to other discriminative sparse codings, which improves the class-based encoding of the test data as well as its interpretability.

I propose a novel kernel-based discriminative sparse coding algorithm with the following training framework

$$\begin{aligned} \textit{Train}: \quad \min_{\mathbf{\Gamma},\mathbf{U}} \quad & \mathcal{R}(\mathcal{X}, \mathbf{\Gamma}, \mathbf{U}) + \alpha\mathcal{F}(\mathbf{H}, \mathbf{\Gamma}, \mathbf{U}) \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T, \ \|\Phi(\mathcal{X})\vec{u}_i\|_2^2 = 1, \\ & \|\vec{u}_i\|_0 \leq T, \ u_{ij}, \gamma_{ij} \in \mathbb{R}_{\geq 0} \ \forall ij \end{aligned} \tag{4.20}$$

and its relevant recall framework as

$$\begin{aligned} \textit{Recall}: \quad \min_{\vec{\gamma}} \quad & \mathcal{R}(\mathcal{X}, \mathbf{Z}, \mathbf{U}, \vec{\gamma}) + \alpha\mathcal{G}(\mathbf{H}, \vec{\gamma}, \mathbf{U}) \\ \text{s.t.} \quad & \|\vec{\gamma}\|_0 < T, \ \gamma_i \in \mathbb{R}_{\geq 0} \ \forall i \end{aligned} \tag{4.21}$$

In these two frameworks, $\mathcal{R}$ is the same reconstruction loss function as in NNKSC sparse coding algorithm (Equation 4.13), and $\{\mathcal{F}, \mathcal{G}\}$ are the novel discriminative loss terms I introduce in this section for the CKSC algorithm. Parameter T applies the l_0-norm sparsity constraint on the columns of $\{\mathbf{U}, \mathbf{\Gamma}\}$, and α is the control factor between the reconstruction and the discriminant terms. In the following sub-sections, I discuss the mathematical detail of these objective terms (as presented in Equation 4.27 and Equation 4.28) and explain my particular design choice motivations.

Reconstruction Term $\mathcal{R}(\mathcal{X}, \mathbf{U}, \mathbf{\Gamma})$

Similar to the K-SRC framework of NNKSC , I define the reconstruction objective of Equation 4.20 as

$$\mathcal{R}(\mathcal{X}, \mathbf{\Gamma}, \mathbf{U}) = \|\Phi(\mathcal{X}) - \Phi(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2. \tag{4.22}$$

Based on Proposition 4.1 and discussion of Section 4.2, $\Phi(\mathcal{X})\mathbf{U}$ is a proper structure for an interpretable kernel-based dictionary.

Proposition 4.2. *Using the dictionary structure of Equation 4.22, sparse reconstruction of sequence $\Phi(\mathbf{X})$ necessitates to bound the value of $\|\vec{u}_i\|_0$.*

Proof. Refer to Appendix A.6. □

Proposition 4.2 justifies the need to have a constraint on $\|\vec{u}_i\|_0$, which leads to the sparse representation of each $\mathbf{X}$ in terms of other samples in $\mathcal{X}$. Such constraint facilitates the interpretation of the encoding vector $\vec{\gamma}$. Accordingly, I choose $\|\vec{u}_i\|_0 \leq T$ in Equation 4.20, which places a more specific bound on the sparseness of $\mathbf{U}$ compared to NNKSC . Nevertheless, due to this constraint in Equation 4.20, I use another optimization algorithm to update columns of $\mathbf{U}$ rather than Algorithm 4.2. Also, similar to the optimization of $\mathbf{U}$ in NNKSC , the constraint $\|\Phi(\mathcal{X})\vec{u}_i\|_2^2 = 1$ prevents the solution of (4.20) from becoming degenerated (Michael Elad and Michal Aharon 2006).

Discriminative Objective $\mathcal{F}(\mathbf{H}, \mathbf{U}, \mathbf{\Gamma})$*:*

Before discussing the mathematical content of $\mathcal{F}(\mathbf{H}, \mathbf{U}, \mathbf{\Gamma})$, I explain the motivation behind my specific choice of $\mathcal{F}$ as the discriminant term. If $\Phi(\mathbf{X})$ is reconstructed as $\Phi(\mathbf{X}) = \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}$, then the entries of $\mathbf{H}\mathbf{U}\vec{\gamma} \in \mathcal{R}^C$ show the share of each class in the reconstruction of $\Phi(\mathbf{X})$. Hence, as an extreme case, if we assume that $\mathbf{X}$ belongs to the class q and $\Phi(\mathbf{X})$ is lying on the subspace of class q in the feature space, we have $\vec{h}^s\mathbf{U}\vec{\gamma} = 0 \ \forall s \neq q$. Vector $\vec{h}^s$ denotes the s-th row of the label matrix $\mathbf{H}$. Proposition 4.3 generalizes this extreme case to a more realistic condition that $\Phi(\mathbf{X})$ is lying on a union of subspaces.

Proposition 4.3. *If sequence $\Phi(\mathbf{X})$ belongs to the class q and is lying on a union of subspaces with arbitrarily small contributions from the subspaces $s \neq q$, then the non-negative discriminant combination $\{\mathbf{U}, \vec{\gamma}\}$ can reconstruct $\Phi(\mathbf{X})$ such that*

$$\frac{\sum_{s \neq q} \vec{h}^s \mathbf{U}\vec{\gamma}}{\vec{h}^q \mathbf{U}\vec{\gamma}} \leq \epsilon$$

for an arbitrarily small ϵ.

Proof. Refer to Appendix A.7. □

Proposition 4.3 provides the following remarks.

Remark 4.1. Via focusing on $\mathbf{U}\vec{\gamma}$, it is possible to obtain a discriminative reconstruction of $\mathbf{X}$ in the feature space by using data points from all of the classes, as long as the class of $\mathbf{X}$ holds the largest share of contributions. This relaxation results in a more flexible dictionary $\mathbf{U}$ regarding both the reconstruction and classification purposes.

Remark 4.2. From a classification point of view, I denote $\mathbf{H}\mathbf{U}\vec{\gamma}$ as the decision vector. The largest entry of $\mathbf{H}\mathbf{U}\vec{\gamma}$ represents the class that its subspace reconstructs the biggest part of $\Phi(\mathbf{X})$ in the feature space and can be considered the class to which $\mathbf{X}$ belongs.

In addition, Proposition 4.3 provides the rationale for having the non-negative constraints on $\mathbf{U}, \mathbf{\Gamma}$ in Equation (4.20). According to the above discussion, I define

$$\mathcal{F}(\mathbf{H}, \mathbf{U}, \vec{\gamma}) = \sum_{s \neq q} \vec{h}^s \mathbf{U}\vec{\gamma},$$

which is the sum of contributions from other classes. Hence, for all $\mathbf{\Gamma}$ we have

$$\mathcal{F}(\mathbf{H}, \mathbf{U}, \mathbf{\Gamma}) = \sum_i \mathcal{F}(\mathbf{H}, \mathbf{U}, \vec{\gamma}_i) = \operatorname{tr}((\mathbf{1} - \mathbf{H}^\top)\mathbf{H}\mathbf{U}\mathbf{\Gamma}) \tag{4.23}$$

where $\mathbf{1} \in \mathbb{R}^{N \times C}$ is a matrix of one, and $\operatorname{tr}(.)$ denotes the matrix trace. Function $\mathcal{F}(\mathbf{H}, \mathbf{U}, \mathbf{\Gamma})$ is a linear term respecting each $\vec{\gamma}_i$ and $\vec{u}_i$ individually. Therefore, it does not violate the convexity of the optimization problem in Equation 4.20. Considering the optimization framework of Equation 4.27, $\mathcal{F}$ is employed along with the additional term $\beta\|\mathbf{U}\mathbf{\Gamma}\|_F^2$. This term preserves the consistency between the training and the recall models and is explained in the next subsection.

Remark 4.3. In contrast to other discriminative sparse coding frameworks such as (Mairal, F. Bach, and Ponce 2012; Z. Jiang, Z. Lin, and Davis 2013; W. Liu et al. 2015), my proposed discriminant term is a compact function of $(\mathbf{H}, \mathbf{U}, \mathbf{\Gamma})$. This structure directly involves the supervised information $\mathbf{H}$ in the optimization of both $\mathbf{U}$ and $\mathbf{\Gamma}$ (Figure 4.4), which improves the discriminative quality of the learned dictionary.

Discriminative Recall Term $\mathcal{G}(\mathbf{H}, \vec{\gamma}, \mathbf{U})$:

For the encoding of a test sequence $\mathbf{Z}$, we use the optimization problem of Equation 4.21. In that case, the reconstruction loss is employed as

$$\mathcal{R}(\mathcal{X}, \mathbf{Z}, \mathbf{U}, \vec{\gamma}) = \|\Phi(\mathbf{Z}) - \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}\|_F^2. \tag{4.24}$$

We can assume that $\Phi(\mathbf{Z}) \in span\{\Phi(\mathcal{X})\}$ and belongs to the class q such that its projection on subspace q as $\|\Phi(\mathbf{Z})^q\|_2$ is arbitrarily larger than $\|\Phi(\mathbf{Z})\|_2 - \|\Phi(\mathbf{Z})^q\|_2$. Therefore, based on Preposition 4.3 and via using the learned $\mathbf{U}$ from Equation 4.20, there exists a $\vec{\gamma}$ that reconstructs the test data as $\Phi(\vec{z}) = \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}$ with more contributions chosen from the class q. Consequently, the class label $\vec{h}_\mathbf{Z}$ is predicted as a zero vector with only $\vec{h}_\mathbf{Z}(j) = 1$, where

$$j = \operatorname*{argmax}_j \vec{h}^j \mathbf{U}\vec{\gamma} \tag{4.25}$$

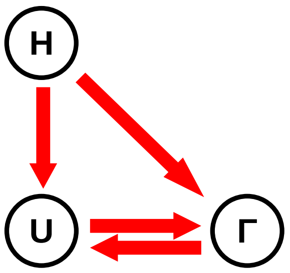


Figure 4.4: The influence of parameters on each other during the optimization loop of CKSC algorithm. An arrow toward a specific parameter indicates other parameters that influence it during its optimization step. The variables **Γ** and **U** have direct effects on each other in their individual optimization steps. They are also directly influenced by the value of **H** in the optimization framework.

In other words, $\vec{h}_z$ is determined by the class of data that has the most contribution to the reconstruction of **Z**.

Since we do not have access to the labeling information for the test data, I propose a cross-entropy-like loss for Equation 4.21 as

$$\mathcal{G}(\mathbf{H}, \vec{\gamma}, \mathbf{U}) = \sum_i \Big(\sum_{s \neq i} \pi_s\Big)\pi_i \quad \text{where } \pi_i := \vec{h}^i \mathbf{U}\vec{\gamma} \tag{4.26}$$

Proposition 4.4. *The proposed term $\mathcal{G}$ in Equation 4.26 is non-convex and has a non-negative gradient.*

Proof. Refer to Appendix A.8. □

Although Proposition 4.4 shows that $\mathcal{G}$ is non-convex, having a non-negative model of $\{\vec{\gamma}, \mathbf{U}\}$ results in a non-negative $\mathcal{G}$, which can have its global optima where $\mathcal{G}(\vec{\gamma}^*) = 0$. Denoting $\vec{\pi}^* = \mathbf{H}\mathbf{U}\vec{\gamma}^*$, besides the trivial solution of $\vec{\pi}^* = 0$, the loss term reaches its global optima when $\vec{\pi}^*$ contains only one non-zero value in its i-th entry. This is equivalent to finding $\vec{\gamma}^*$ such that it reconstructs **Z** using contributions only from one class of data.

Consequently, the non-trivial minima of both regularization terms in Equation 4.23 and Equation 4.26 occur at similar points where the decision vector $\mathbf{H}\mathbf{U}\vec{\gamma}$ has approximately a crisp form regarding only one of its entries. Therefore, adding $\mathcal{G}$ increases the consistency between training and the test frameworks.

Proposition 4.5. *Define*

$$\mathbf{V} := \mathcal{K}(\mathcal{X}, \mathcal{X}) + \alpha \mathbf{H}^\top (\mathbf{1} - \mathbf{I}_{C \times C})\mathbf{H}$$

and $\beta := -\min_i \lambda_i$, with $\{\lambda_i\}_{i=1}^N$ as the eigenvalues of $\mathbf{V}$. Adding $\beta \|\mathbf{U}\vec{\gamma}\|_2^2$ to the objective term $\mathcal{G}$ (Equation 4.26) makes Equation 4.21 a convex optimization problem.

Proof. Refer to Appendix A.9. □

In order to preserve the consistency between the test and training models, I also add the term $\beta\|\mathbf{U}\boldsymbol{\Gamma}\|_F^2$ to the discriminant loss $\mathcal{F}$ of Equation 4.23, which results in the complete training framework of Equation 4.27. By doing so, we want to make sure the dictionary $\mathbf{U}$ has a consistent role in both train and test optimization problems. Furthermore, parameter β is independent of the test data and is computed only once and prior to the optimization phase.

Remark 4.4. The added term $\|\mathbf{U}\vec{\gamma}\|_2^2$ to Equation 4.21 has a similar effect to the l_2-norm regularization term used in the elastic net formulation from (Zou and Hastie 2005) in the vectorial case:

$$\min_{\vec{\gamma}} \|\vec{x} - \mathbf{X}s\|_2^2 + \beta\|s\|_2^2 + \alpha\|s\|_1.$$

Hence, this term relaxes the model's sparseness according to the weighting scalar β, which is similar to the grouping effect in the above elastic net problem (Zou and Hastie 2005). Nevertheless, this added term does not apply any restriction regarding the discriminative structure of $\vec{\gamma}$.

Distinguishing CKSC from the Related Work

Due to the extensive discriminative sparse coding frameworks existing in the literature, I want to compare the structure of my proposed CKSC algorithm to the related work from the following explicit aspects:

- Compared to the methods such as (F. Bach et al. 2008; N. Zhou et al. 2012; S. Kong and D. Wang 2012), which mainly define multiple isolated dictionaries for each class of data, CKSC uses a single seamless dictionary for all classes. However, each dictionary column is still formed based on the contributions mostly from one class of data. This structure makes the dictionary efficient also for the reconstruction of the inter-class overlaps in the data.

- Some algorithms, such as (Z. Jiang, Z. Lin, and Davis 2013; W. Liu et al. 2015; Quan, Y. Xu, et al. 2016), employ $\mathbf{H}$ in their discriminant models via using additional parameters (such as $\mathbf{W}$ in Equation 4.8). However, as explained in Section 4.1, the optimization of $\mathbf{U}$ is not directly influenced by $\mathbf{H}$ (Figure 4.2-a). In contrast, the linear discriminant term in the CKSC directly involves the value of $\mathbf{H}$ in the optimization of the dictionary. This formulation incorporates the underlying formation of the classes into the structure of the learned dictionary.

- Different from algorithms such as (Mairal, F. Bach, and Ponce 2012; Z. Jiang, Z. Lin, and Davis 2013; W. Liu et al. 2015; Quan, Y. Xu, et al. 2016), my proposed method considers a discriminative term also for the recall phase to influence the resulting $\vec{\gamma}$ toward achieving a more confident discriminative representation.

In the next section, I explain the optimization steps regarding frameworks of Equation 4.20 and Equation 4.21.

Optimization Scheme

By rewriting Equation 4.20 and Equation 4.21 using the provided descriptions of $\mathcal{F}$ and $\mathcal{G}$ in the previous section, we obtain the following optimization framework of training

$$\begin{array}{lll} \textit{Train}: & \min\limits_{\mathbf{\Gamma},\mathbf{U}} & \|\Phi(\mathcal{X})-\Phi(\mathcal{X})\mathbf{U\Gamma}\|_F^2+\beta\|\mathbf{U\Gamma}\|_F^2 \\ & & +\alpha\mathrm{tr}\{(\mathbf{1}-\mathbf{H}^\top)\mathbf{HU\Gamma}\} \\ & \text{s.t.} & \|\vec{\gamma}_i\|_0<T,\ \ \|\Phi(\mathcal{X})\vec{u}_i\|_2^2=1, \\ & & \|\vec{u}_i\|_0\le T,\ \ u_{ij},\gamma_{ij}\in\mathbb{R}_{\ge0},\ \ \forall ij \end{array} \tag{4.27}$$

and its relevant recall problem as

$$\begin{array}{lll} \textit{Test}: & \min\limits_{\vec{\gamma}} & \|\Phi(\mathbf{Z})-\Phi(\mathcal{X})\mathbf{U}\vec{\gamma}\|_F^2+\beta\|\mathbf{U}\vec{\gamma}\|_F^2 \\ & & +\alpha(\vec{\gamma}^\top\mathbf{U}^\top\mathbf{H}^\top(\mathbf{1}-\mathbf{I})\mathbf{HU}\vec{\gamma}) \\ & \text{s.t.} & \|\vec{\gamma}\|_0<T,\ \gamma_i\in\mathbb{R}_{\ge0}\ \ \forall i \end{array} \tag{4.28}$$

Although the optimization problem of Equation 4.27 is not convex w.r.t. $\{\mathbf{U},\mathbf{\Gamma}\}$ together, we can train the discriminantive sparse coding model in 2 alternating convex optimization steps. At each step, I update one of the parameters while fixing the other one, as presented in Algorithm 4.4.

Update of the Sparse Codes $\mathbf{\Gamma}$

The entire objective function in Equation 4.27 has a column-separable structure w.r.t. $\mathbf{\Gamma}$, and it can be optimized for each $\vec{\gamma}_i$ individually. Therefore, after removing the constant terms, Equation 4.27 is rewritten w.r.t. each $\vec{\gamma}_i$ as

$$\begin{array}{ll} \min\limits_{\vec{\gamma}_i} & \vec{\gamma}_i^\top\left[\mathbf{U}^\top(\mathcal{K}+\beta\mathbf{I})\mathbf{U}\right]\vec{\gamma}_i \\ & +\left[\alpha(\mathbf{1}-\vec{h}_i^\top)\mathbf{HU}-2\mathcal{K}(\vec{x}_i,\mathbf{X})\mathbf{U}\right]\vec{\gamma}_i \\ \text{s.t.} & \|\vec{\gamma}_i\|_0<T,\ \ \gamma_{ij}\in\mathbb{R}_{\ge0}\ \ \forall ij, \end{array} \tag{4.29}$$

where $\mathcal{K}$ stands for $\mathcal{K}(\mathcal{X},\mathcal{X})$. This optimization framework is a non-negative quadratic programming problem with the constraint $\|\vec{\gamma}_i\|<T$. Furthermore, $\mathcal{K}+\beta\mathbf{I}$ is a PSD matrix, and consequently Equation 4.29 is a convex problem. In order to optimize such problems, I propose the Non-negative Quadratic Pursuit (NQP) algorithm, which is a particular generalization of the Matching Pursuit approach (M. Aharon, M. Elad, and Bruckstein 2006). NQP is presented in Algorithm 4.5 and discussed in a later section.

Update of the Dictionary $\mathbf{U}$

Similar to Equation 4.29, it is also possible to reformulate the objective terms of Equation 4.27 w.r.t. each dictionary column $\vec{u}_i$ separately. Its reconstruction part $\mathcal{R}(\mathcal{X},\mathbf{\Gamma},\mathbf{U})$ can be rewritten as Equation 4.15 for NNKSC using the additional matrix $\mathbf{E}_i$. Likewise, the rest of the objective parts in Equation 4.27 can be written in terms of $\vec{u}_i$ as

$$\beta\vec{u}_i^\top(\vec{\gamma}^i\vec{\gamma}^{i\top}\mathbf{I})\vec{u}_i+\alpha\vec{\gamma}^i(\mathbf{1}-\mathbf{H}^\top)\mathbf{L}\vec{u}_i+2\beta\vec{\gamma}^i(\mathbf{I}-\mathbf{E}_i^\top)\vec{u}_i,$$

Algorithm 4.4 The CKSC algorithm: finds an approximate solution to Equation 4.20 as a non-negative sparse encoding of motion sequences in the feature space under the cardinality constraint and while preserving supervised information.

```
1: Parameters: discriminant weight $\alpha$, sparseness limit $T$, and stopping threshold $\delta$.
2: Input: Labels $\mathbf{H}$, kernel matrix $\mathcal{K}(\mathcal{X},\mathcal{X})$.
3: Output: Sparse coefficients $\mathbf{\Gamma}$, discriminant dictionary $\mathbf{U}$.
4: Initialization: Computing $\beta$ based on Proposition 4.5.
5: while $\mathcal{R}(\mathbf{X},\mathbf{\Gamma},\mathbf{U}) + \alpha\mathcal{F}(\mathbf{H},\mathbf{\Gamma},\mathbf{U}) > \delta$ in Equation 4.20 do
6:     Update $\mathbf{\Gamma}$ based on Equation 4.29 using NQP.
7:     Update $\mathbf{U}$ based on Equation 4.30 using NQP.
8: end while
```

where the constant parts are eliminated. So, by using the kernel function $\mathcal{K}(\mathcal{X},\mathcal{X})$, I reformulate Equation 4.27 for updating $\vec{u}_i$ as

$$\begin{aligned} \min_{\vec{u}_i} \quad & \beta\vec{u}_i^\top(\vec{\gamma}^i\vec{\gamma}^{i\top}(\mathcal{K}+\beta\mathbf{I}))\vec{u}_i \\ & +\vec{\gamma}^i\left[\alpha(\mathbf{1}-\mathbf{H}^\top)\mathbf{L}+2\beta(\mathbf{I}-\mathbf{E}_i^\top)-2\mathbf{E}_i^\top\mathcal{K}\right]\vec{u}_i \\ \text{s.t.} \quad & \|\Phi(\mathcal{X})\vec{u}_i\|_2^2=1,\ \|\vec{u}_i\|_0\le T\ ,\ u_{ij}\in\mathbb{R}_{\ge 0}\ \ \forall j \end{aligned} \tag{4.30}$$

Similar to Equation 4.29, the above framework has the non-negative quadratic form with the cardinality constraint $\|\vec{u}_i\|_0 \le T$ and is a convex problem as well. Consequently, its solution can be approximated using the proposed NQP method (Algorithm 4.5).

Note: The same considerations regarding computing $\mathbf{E}_i$ based on the updated value of $\mathbf{U}$ in each step and normalization each $\Phi(\mathcal{X})\vec{u}_i$ should be taken similar to the NNKSC algorithm (Section 4.2).

Update of the Recall Phase $\vec{\gamma}$:

To reconstruct the test sequence $\mathbf{Z}$, its corresponding sparse code $\vec{\gamma}$ is approximated via expanding Equation 4.28 as follows

$$\begin{aligned} \min_{\vec{\gamma}} \quad & \vec{\gamma}^\top\mathbf{U}^\top\left[\mathcal{K}+\alpha\mathbf{H}^\top(\mathbf{1}-\mathbf{I})\mathbf{H}+\beta\mathbf{I}\right]\mathbf{U}\vec{\gamma} \\ & -2\mathcal{K}(\vec{z},\mathbf{X})\mathbf{U}\vec{\gamma} \\ \text{s.t.} \quad & \|\vec{\gamma}\|_0 < T,\ \ \gamma_j\in\mathbb{R}_{\ge 0}\ \ \forall j \end{aligned} \tag{4.31}$$

The convexity of this optimization problem is guaranteed based on Proposition 4.5, and can be approximately solved by the NQP algorithm similar to the update of $\mathbf{\Gamma}$ and $\mathbf{U}$.

Non-negative Quadratic Pursuit (NQP)

Consider a quadratic function $f(\vec{\gamma}) := \frac{1}{2}\vec{\gamma}^\top\mathbf{Q}\vec{\gamma}+\vec{c}^\top\vec{\gamma}$, in which $\vec{\gamma}\in\mathbb{R}^n$, $\vec{c}\in\mathbb{R}^n$, and $\mathbf{Q}\in\mathbb{R}^{n\times n}$ is a hermitian positive semidefinite matrix. Non-negative quadratic pursuit algorithm (NQP) is an extended form of the Matching Pursuit problem (M. Aharon, M. Elad, and Bruckstein 2006) and is inspired by by (H. Lee et al. 2006). Its objective is to approximately minimize $f(\vec{\gamma})$ in an NP-hard optimization problem similar to

$$\begin{aligned} \vec{\gamma} = \quad & \arg\min_{\vec{\gamma}} \tfrac{1}{2}\vec{\gamma}^\top\mathbf{Q}\vec{\gamma}+\vec{c}^\top\vec{\gamma} \\ & s.t. \quad \|\vec{\gamma}\|_0\le T\ ,\ \gamma_i\ge 0\ \ \forall i, \end{aligned} \tag{4.32}$$

where at most $T \ll n$ elements from $\vec{\gamma}$ are permitted to be positive while all other elements are forced to be zero. As presented in Algorithm 4.5, I compute $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ at each iteration of NQP to guess the next promising dimension of $\vec{\gamma}$ (denoted as γ_j), leading to the largest decrease in the current value of $f(\vec{\gamma}_{\mathcal{I}})$, where $\mathcal{I}$ denotes the set of currently chosen dimensions of $\vec{\gamma}$ based on the previous iterations. I look for a $\vec{\gamma} \geq 0$ solution in each iteration. Also, the entries of $\vec{\gamma}$ corresponding to new possible dimensions are all initially zero. Therefore, similar to the Gauss-Southwell rule in coordinate descent optimization (Nesterov 2012), I choose the dimension j that is related to the smallest negative entry of $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ as

$$j = \underset{j \in S}{\arg\min}\, \vec{q}_j^\top \vec{\gamma} + c_j \qquad s.t.\ \vec{q}_j^\top \vec{\gamma} + c_j < 0. \tag{4.33}$$

In Equation 4.33, $\vec{q}_j$ is the j-th column of $\mathbf{Q}$. Then, by adding j to $\mathcal{I}$, the resulting unconstrained quadratic problem can be solved using the closed-form solution $\vec{\gamma}_{\mathcal{I}} = -\mathbf{Q}_{\mathcal{I}\mathcal{I}}{}^{-1}\vec{c}_{\mathcal{I}}$. Accordingly, I repeat this process until reaching $\|\vec{\gamma}\|_0 = T$ stopping criterion. Corresponding to the set $\mathcal{I}$, notations $\mathbf{Q}_{\mathcal{I}\mathcal{I}}$ and $\vec{c}_{\mathcal{I}}$ indicate the principal submatrix of $\mathbf{Q}$ and the subvector of $\vec{c}$, respectively.

In order to preserve non-negativity of the solution $\vec{\gamma}$ in each iteration t of NQP, in case of having a negative entry in $\vec{\gamma}^t_{\mathcal{I}}$, a simple line search is performed between $\vec{\gamma}^t_{\mathcal{I}}$ and $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$. The line search chooses the nearest zero-crossing point to $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$ on the connecting line between $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$ and $\vec{\gamma}^t_{\mathcal{I}}$.

In addition, to reduce the computational cost, I use the Cholesky factorization $\mathbf{Q}_{\mathcal{I}\mathcal{I}} = \mathbf{L}\mathbf{L}^\top$ (Van Loan 1996) to compute $\vec{\gamma}$ with a back-substitution process. Furthermore, because matrix $\mathbf{Q}$ in equations (4.32) is PSD, its principal sub-matrix $\mathbf{Q}_{\mathcal{I}\mathcal{I}}$ should be either PD or PSD theoretically (Johnson and H. A. Robinson 1981), where the first case is a requirement for the Cholesky factorization. However, by choosing $T << rank(Q)$ in practice, the optimization loop has never encountered a singular condition. Nevertheless, to avoid such rare conditions, I do a non-singularity test for the selected dimension j by examining $q_{jj} \neq v^\top v$ after obtaining v (Step 12 in Algorithm 4.5). In case the resulting v does not fulfill that condition, I choose another j based on Equation 4.33.

The Convergence of NQP

NQP does not guarantee the global optimum as it is a greedy selection of rows/columns of matrix $\mathbf{Q}$ to provide a sparse approximation of the NP-hard problem in Equation 4.32; nevertheless, its convergence to a local optimum point is guaranteed.

Theorem 4.1. *The Non-negative Quadratic Pursuit algorithm (Algorithm 4.5) converges to a local minimum of Equation 4.32 in a limited number of iterations.*

Proof. Refer to Appendix A.10. □

The Computational Complexity of NQP

We can calculate the computational complexity of NQP by considering its individual steps. Iteration t contains computing $\mathbf{Q}\vec{\gamma} + \vec{c}$ ($nt + t$ operation), finding minimum of $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ w.r.t. the negative constraint ($2n$ operations), computing v (t^2 operation for the $t \times t$ back-substitution), computing $\vec{\gamma}^t_{\mathcal{I}}$ (two back-substitutions resulting in $2t^2$ operation),

Algorithm 4.5 The non-negative quadratic pursuit algorithm: finds an approximate solution to the optimization problem of Equation 4.32, which is a non-negative quadratic problem in the presence of a carnality constraint.

```
1: Parameters: Carnality limit $T$, stopping threshold $\epsilon$.
2: Input: $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\vec{c} \in \mathbb{R}^n$ from Equation 4.32.
3: Output: An approximate solution $\vec{\gamma}$.
4: Initialization: $\vec{\gamma} = 0$ , $\mathcal{I} = \{\}$ , $\mathcal{S} = \{1, ..., n\}$ , $t = 1$.
5: do
6:     $j = \arg\min_{j \in S} \vec{q}_j^\top \vec{\gamma} + c_j \qquad s.t.\ \vec{q}_j^\top \vec{\gamma} + c_j < 0$
7:
8:     if $j = \varnothing$ then Convergence.
9:
10:    $\mathcal{I} := \mathcal{I} \cap j$;
11:
12:    $\vec{q}_{\mathcal{I}j}$ := created via selecting rows $\mathcal{I}$ and column $j$ of matrix $\mathbf{Q}$.
13:
14:    $\vec{c}_{\mathcal{I}}$ := a subvector of $c$ based on selecting entries $\mathcal{I}$ of vector $\vec{c}$.
15:
16:    if $t > 1$ then
17:        $v$ := Solve for $v$ $\{\mathbf{L}v = \vec{q}_{\mathcal{I}j}\}$;
18:
19:        $\mathbf{L} := \begin{bmatrix} \mathbf{L} & 0 \\ v^\top & \sqrt{q_{jj} - v^\top v} \end{bmatrix}$
20:
21:    else
22:        $\mathbf{L} = q_{jj}$
23:    end if
24:    $\vec{\gamma}_{\mathcal{I}}^t$ := Solve for $x$ $\{\mathbf{L}\mathbf{L}^\top x = \vec{c}_{\mathcal{I}}\}$;
25:
26:    if $\exists j \in \mathbb{N};\ (\gamma_j^t < 0)$ then
27:        $\vec{\gamma}_{\mathcal{I}}^t$ := the nearest zero-crossing to $\vec{\gamma}_{\mathcal{I}}^{(t-1)}$ via a line search.
28:
29:        $\mathcal{S} := \mathcal{S} - \{$zeros entries in $\vec{\gamma}_{\mathcal{I}}^t\}$
30:    end if
31:    $\mathcal{S} := \mathcal{S} - j$
32:
33:    $t = t + 1$
34: while $(\mathcal{S} \neq \{\}) \wedge (\|\vec{\gamma}\|_0 < T) \wedge (\frac{1}{2}\vec{\gamma}^\top Q \vec{\gamma} + c^\top \vec{\gamma} \geq \epsilon)$
```

and checking negativity of entries of $\vec{\gamma}_{\mathcal{I}}^t$ along with the probable line-search which has $3t$ operations in total. Hence, the total runtime of iteration t is bounded by

$$\mathbf{T}_t = (n+4)t + 2n + 3t^2,$$

, and the total runtime of the algorithm is

$$\begin{aligned} \mathbf{T}_{NQP} &= \textstyle\sum_{t=1}^{T} (n+4)t + 2n + 3t^2 \\ &= 2nT + T(T+1)[\tfrac{n+2T+5}{2}]. \end{aligned} \tag{4.34}$$

Hence, its computational complexity is bounded by $\mathcal{O}(n(2T + \frac{T^2}{2}) + \frac{T^3}{2})$. When one uses NQP for estimating the sparse codes, its run-time would be linear in terms of the dictionary size k. In contrast, the NN-KNOP method's run-time complexity (Algorithm 4.1) is $\mathcal{O}((5T + T^2)\frac{N}{2} + T^{4.3})$ which is linear in terms of data size N.

Although both algorithms look for maximum T elements to estimate $\vec{\gamma}$, due to the non-negativity constraint and convexity of the problem, both algorithms converge in a small number of iterations (usually smaller than 20), which is independent of the dictionary or data size or the value of T. Hence, without investigating a rigorous mathematical analysis, both algorithms show superlinear convergence empirically.

In addition, as it is discussed in Section 4.5, we can choose k based on T and C in practice. Therefore, dictionary size k is not dependent on data size N and more often, it is saturated to a specific value for large N. This facts results in having a fixed upper bound for run-time complexity of NQP when estimating $\vec{\gamma}$, while NN-KOMP computations are linearly affected by the data size (even for each encoding vector $\vec{\gamma}$).

From another point of view, when updating a dictionary atom $\vec{u}_i$, the NN-KFISTA method for NNKSC and LC-NNKSC algorithms (Section 4.2) has $\mathcal{O}(N^2)$ runtime complexity for big values of N. In comparison, updating each $\vec{u}_i$ using NQP optimization has an $\mathcal{O}(N)$ complexity. Furthermore, since NQP is applicable to quadratic problems, we can assume that NN-KOMP addresses a subset of the NQP application domain. Based on the above facts, I can claim that NQP is more efficient than both NN-KOMP and NN-KFISTA in terms of run-time complexity and the application domain.

4.4 MOTION CLUSTERING USING NON-NEGATIVE KERNEL SPARSE CODING

As discussed before, K-SSC methods can obtain a particular sparse encoding of non-vectorial datasets. This encoding reveals the underlying categories of data given that no supervised information is provided. The K-SSC sparse coding models are constructed upon the self-representation of the data distribution. For example, for mocap data, each motion sequence would be directly represented by other sequences based on a given underlying semantic similarity (kernel information). As cited in Section 4.1, K-SSC methods' non-negative variations enhance the interpretation of the self-representative encoding vectors for non-vectorial data such as motion sequences. In this section, I propose a novel K-SSC that improves the quality of the self-representation encoding w.r.t. revealing the underplaying subspaces in the data distribution. To that aim, I propose a novel optimization framework for the K-SSC problem and a post-processing algorithm to enhance the obtained encoding for the clustering purpose.

In the following, I first introduce my proposed SSC algorithm, non-negative local subspace sparse clustering (NLSSC), for the general case of vectorial input data. Later, I extend this algorithm to its kernel-based variation NKLSSC, which is also applicable to other input types such as motion sequences.

Non-negative Local Subspace Sparse Clustering

Considering vectorial data representation $\vec{x}$ and the dataset matrix $\mathbf{X}$, I formulate my non-negative local SSC algorithm (NLSSC) using the following self-representative framework:

$$\begin{array}{ll} \min\limits_{\mathbf{\Gamma}} & \|\mathbf{\Gamma}\|_* + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{\Gamma}\|_F^2 + \mu\mathcal{E}_{lsp}(\mathbf{\Gamma}, \mathbf{X}) \\ \text{s.t.} & \mathbf{\Gamma}^\top\vec{\mathbf{1}} = \vec{\mathbf{1}}, \gamma_{ij} \geq 0, \quad \gamma_{ii} = 0 \ \ \forall ij, \end{array} \tag{4.35}$$

where $\gamma_{ii} = 0$ prevents data points from being represented by their own contributions as trivial solutions. The constraint $\mathbf{\Gamma}^\top \vec{\mathbf{1}} = \vec{\mathbf{1}}$ focuses on the affine reconstruction of data points, which coincides with having the data lying in an affine union of subspaces $\cup_{l=1}^{n} \mathcal{S}_l$. The nuclear norm regularization term $\|\mathbf{\Gamma}\|_* = trace(\sqrt{\mathbf{\Gamma}^* \, \mathbf{\Gamma}})$ is employed to ensure the sparse coding representations are low-rank. This term specifically helps the sparse model to capture the global structure of data distribution more appropriately.

The non-negativity constraint of Equation 4.35 on γ_{ij} is employed to enforce the data combinations to happen mostly between similar samples. In other words, this term aids the sparse encoding to become more interpretable regarding the semantic meaning of its entries. The novel term $\mathcal{E}_{lsp}(\mathbf{\Gamma}, \mathbf{X})$ is a loss function that focuses on the local separation of data points in the coding space according to columns of $\mathbf{\Gamma}$. Accordingly, scalars λ and μ are constant weights, which control the contribution of these objective terms.

The goal of minimizing $\mathcal{E}_{lsp}(\mathbf{\Gamma}, \mathbf{X})$ in the SSC model is to reduce intra-cluster distance and increase inter-cluster distance. To that aim in an unsupervised setting, I define:

$$\mathcal{E}_{lsp}(\mathbf{\Gamma}, \mathbf{X}) := \frac{1}{2} \sum_{i,j} \left[w_{ij} \|\vec{\gamma}_i - \vec{\gamma}_j\|_2^2 + b_{ij} (\vec{\gamma}_i^\top \vec{\gamma}_j) \right], \tag{4.36}$$

in which the binary regularization weighting matrices $\mathbf{W}$ and $\mathbf{B}$ are computed as

$$w_{ij} = \begin{cases} 1, & \text{if } \vec{x}_j \in \mathcal{N}_i^k \\ 0, & \text{otherwise} \end{cases}, \qquad b_{ij} = \begin{cases} 1, & \text{if } \vec{x}_j \in \mathcal{F}_i^k \\ 0, & \text{otherwise} \end{cases} \tag{4.37}$$

The two sets $\mathcal{N}_i^k$ and $\mathcal{F}_i^k$ refer to the k-nearest and k-farthest data points to $\vec{x}_i$. These sets are determined via computing Euclidean distance $\|\vec{x}_i - \vec{x}_j\|_2$ between each $\vec{x}_i$ and $\vec{x}_j$. Defining

$$\begin{array}{l} \mathcal{J}(\mathbf{W}, \mathbf{\Gamma}) := \sum_{i,j} w_{ij} \|\vec{\gamma}_i - \vec{\gamma}_j\|_2^2 \\ \mathcal{S}(\mathbf{B}, \mathbf{\Gamma}) := \sum_{i,j} b_{ij} (\vec{\gamma}_i^\top \vec{\gamma}_j), \end{array} \tag{4.38}$$

minimizing $\mathcal{J}(\mathbf{W}, \mathbf{\Gamma})$ part reduces the distance between $(\vec{\gamma}_i, \vec{\gamma}_j)$ if they belong to $\mathcal{N}_i^k$, while minimizing $\mathcal{S}(\mathbf{B}, \mathbf{\Gamma})$ reduces the incoherency of each pair of $(\vec{\gamma}_i, \vec{\gamma}_j)$ if they are members of $\mathcal{F}_i^k$.

We can compare $\mathcal{J}(\mathbf{W}, \mathbf{\Gamma})$ to the last loss term in Equation 4.12. However, the similarity matrix $\mathbf{W}$ in Equation 4.12 defines a global distribution of similarity values between each $\vec{x}_i$ and the rest of dataset. Hence, using such $\mathbf{W}$ as the weighting scheme in Equation 4.12 stretches the neighborhoods in the space spanned by columns of $\mathbf{\Gamma}$, which naturally leads to overlapping $\vec{\gamma}_i$ vectors among different subspaces in such space. On the other hand, my proposed $\mathbf{W}$ in Equation 4.37 locally connects similar data samples in $\mathbf{X}$. Employing such $\mathbf{W}$ in $\mathcal{J}(\mathbf{W}, \mathbf{\Gamma})$ to localize the neighborhoods in $\mathbf{\Gamma}$, which better projects the underlying existing data subspaces. Adding to the above, decreasing $\mathcal{S}(\mathbf{B}, \mathbf{\Gamma})$ in $\mathcal{E}_{lsp}$ generally increases $\mathcal{J}(\mathbf{B}, \mathbf{\Gamma})$ as the distance between distant neighborhoods, which globally makes the neighborhoods in the encoding space more separated. As a result, minimizing the loss term $\mathcal{E}_{lsp}$ results in having localized and condense neighborhoods in the sparse codes $\mathbf{\Gamma}$ by making the sparse codes of the neighboring samples more similar (identical in the ideal case) while making those of faraway points incoherent (orthogonal in the ideal case). It also provides the desired condition by which the local neighborhoods in $\mathbf{\Gamma}$ can better respect the class labels $\vec{l}$ and leading to a better alignment between $\mathbf{\Gamma}$ and the underlying subspaces.

Algorithm 4.6 The Link-Restore algorithm: performs post-processing on the encoded sparse vectors from Equation 4.35, which revives the broken links in the representation graph corresponding to $\mathbf{\Gamma}$.

```
1: Input: Sparse code $\vec{\gamma}$, data matrix $\mathbf{X}$, threshold $\tau \in [0,1]$.
2: Output: Corrected $\vec{\gamma}$ by restoring its connections to other data points.
3: Initializing variables $I = \{i \mid \gamma_i \neq 0\}$ (except index of $\vec{x}$)
4: for all $i \in I$ do
5:     $\hat{\vec{\gamma}} = \vec{\gamma}$.
6:     $\bar{I} := \{s \mid (\vec{x}_s^\top \vec{x}_s - 2\vec{x}_i^\top \vec{x}_s) < (\tau - 1)\vec{x}_i^\top \vec{x}_i \ , \ \gamma_s = 0\}$.
7:     $\hat{\gamma}_i = \gamma_i(\vec{x}_i^\top \vec{x}_i / \sum_{s \in \{\bar{I} \cup i\}} \vec{x}_i^\top \vec{x}_s)$.
8:     $\hat{\gamma}_s = \hat{\gamma}_i(\vec{x}_i^\top \vec{x}_s / \vec{x}_i^\top \vec{x}_i) \ , \ \forall s \in \bar{I}$.
9:     $\vec{\gamma} = \hat{\vec{\gamma}}, \quad I = I \backslash \{i\}$.
10: end for
```

Clustering based on $\mathbf{\Gamma}$

Similar to other SSC algorithms, the resulting sparse coefficient matrix $\mathbf{\Gamma}$ is used to construct an adjacency matrix

$$\mathbf{A} = \mathbf{\Gamma} + \mathbf{\Gamma}^\top, \tag{4.39}$$

which defines a sparse representation graph $\mathcal{G}$. This undirected graph consists of non-negative weighted connections between pairs of $(\vec{x}_i, \vec{x}_j)$, representing the local connections of data points in the input space. Therefore, we can use $\mathbf{A}$ as the affinity matrix in the spectral clustering algorithm (Y. Yang, Zhangyang Wang, et al. 2014) to find the data clusters.

Link-Restore

After constructing the affinity matrix based on $\mathbf{\Gamma}$, it is desired to observe positive weights in the representation graph $\mathcal{G}$ between every two points of a given data cluster. However, it is possible to see non-connected nodes (broken links) even inside condense clusters in practice. This happens due to the redundancy issue related to sparse coding algorithms. In Equation 4.35, $\mathbf{X}$ is used as an over-complete dictionary to reconstruct each $\vec{x}_i$. Therefore, we can assume $\vec{x}_i \approx \mathbf{X}\vec{\gamma}_i$. Nevertheless, as a common observation in sparse coding models, the solution for the value of $\vec{\gamma}_i$ is suboptimal because of the utilized $\|\vec{\gamma}_i\|_p$ relaxations. Thus for $\vec{x}_s$ as a close data point to $\vec{x}_i$, it is possible to have $\vec{x}_s \approx \mathbf{X}\vec{\gamma}_s$, but with a big $\vec{\gamma}_i^\top \vec{\gamma}_s$. This observation means $\vec{\gamma}_i$ and $\vec{\gamma}_s$ are not similar in the entries. Consequently, a_{ij} can be small, resulting from distinct $\vec{\gamma}_i$ and $\vec{\gamma}_s$, albeit $\vec{x}_i$ and $\vec{x}_s$ are very similar.

As a workaround to the mentioned issue, I propose the Link-Restore method (Algorithm 4.6) as an effective step regarding these situations. It acts as a post-processing step on the obtained $\mathbf{\Gamma}$ before the application of spectral clustering. Link-restore corrects entries of each $\vec{\gamma}$ by restoring the broken connections between $\vec{x}$ and other points in the dataset. To do so, it first obtains the current set of data points connected to $\vec{x}$ as $I = \{i \mid \gamma_i \neq 0\}$, where γ_i denotes the i-th entry in vector $\vec{\gamma}$. Then for each $\vec{\gamma}_i$ that $i \in I$, the algorithm collects the indices $\bar{I}$ of data points close to $\vec{x}_i$ but not used in the sparse code of $\vec{x}$ (line 6). To that aim, for each $\vec{x}_s \in \bar{I}$ the criterion $\|\vec{x}_i - \vec{x}_s\|_2^2 / \|\vec{x}_i\|_2^2 < \tau$ should be fulfilled, where $0 \leq \tau \leq 1$. Then, to incorporate members of $\bar{I}$ into $\vec{\gamma}$, the entry γ_i is projected to $\bar{I} \cup i$ based on the value of $\frac{\vec{x}_i^\top \vec{x}_s}{\vec{x}_i^\top \vec{x}_i} \ \ \forall s \in \bar{I}$ while also maintaining the affinity constraint on $\vec{\gamma}$ (lines 7-8). It is essential to point out that the pre-assumption for the above is that $\gamma_i \geq 0 \ \ \forall i$. Therefore the link-restore method can be assumed as a proper post-processing method for *non-negative* subspace clustering algorithms.

Kernel Extension of NLSSC

In order to apply the proposed NLSSC to non-vectorial data such as motion, we can use the kernel representation of data as $\mathcal{K}(\mathcal{X}, \mathcal{X})$, where the set $\mathcal{X}$ contains motion sequences. Even using such representation for vectorial data, we can benefit from the non-linear characteristics of this implicit mapping to obtain better representation for the data. Accordingly, I can reformulate my NLSSC method (Equation 4.35) into its kernel extension as the non-negative local kernel SSC algorithm (NLKSSC):

$$\begin{array}{ll} \min\limits_{\mathbf{\Gamma}} & \|\mathbf{\Gamma}\|_* + \frac{\lambda}{2}\|\Phi(\mathbf{X}) - \Phi(\mathbf{X})\mathbf{\Gamma}\|_F^2 + \mu\mathcal{E}_{lsp}(\mathbf{\Gamma}, \Phi(\mathbf{X})) \\ \text{s.t.} & \mathbf{\Gamma}^\top\vec{\mathbf{1}} = \vec{\mathbf{1}}, \ \gamma_{ij} \geq 0, \ \ \gamma_{ii} = 0 \ , \ \forall ij \end{array} \tag{4.40}$$

Comparing to Equation 4.35, the second term in the objective of Eq.4.40 means a self-representation in the feature space, and the local-separability term ($\mathcal{E}_{lsp}$) is equivalent to the one used in 4.35. However, $\mathbf{W}$ and $\mathbf{B}$ in $\mathcal{E}_{lsp}$ are computed based on the entries $\mathcal{K}(\vec{x}_i, \vec{x}_j)$ which directly indicate the pairwise similarity of each data $\vec{x}_i$ to its surrounding neighborhood. The benefit of having a kernel representation of $\mathbf{X}$ is that a proper kernel function leads to the more efficient role of $\mathcal{E}_{lsp}$ in Equation 4.35. As we see in Sec. 4.4, we can use the same optimization regime for both NLSSC and NLKSSC . Also, to kernelize the link-restore algorithm, simply the lines 6-8 of the Algorithm 4.6 would be modified by replacing any $\vec{x}_i^\top \vec{x}_j$ with $\mathcal{K}(\vec{x}_i, \vec{x}_i)$ according to the above dot-product rule.

Optimization Scheme of NLKSSC

By putting Equation 4.36 into Equation 4.40, the following optimization framework is derived

$$\begin{array}{ll} \min\limits_{\mathbf{\Gamma}} & \|\mathbf{\Gamma}\|_* + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{\Gamma}\|_F^2 + \frac{\mu}{2}\sum_{i,j}\left[w_{ij}\|\vec{\gamma}_i - \vec{\gamma}_j\|_2^2 + b_{ij}(\vec{\gamma}_i^\top\vec{\gamma}_j)\right] \\ \text{s.t.} & \mathbf{\Gamma}^\top\vec{\mathbf{1}} = \vec{\mathbf{1}}, \ \gamma_{ij} \geq 0, \ \ \gamma_{ii} = 0 \ , \ \forall ij \end{array} \tag{4.41}$$

To simplify the third loss term in (4.41), I symmetrize $\mathbf{W}$ as $\mathbf{W} \leftarrow \frac{\mathbf{W}+\mathbf{W}^\top}{2}$ and do the same for $\mathbf{B}$. Then, I compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ according to (Von. Luxburg 2007), where $\mathbf{D}$ is a diagonal matrix such that $d_{ii} = \sum_j w_{ij}$. Then, we can rewrite $\mathcal{E}_{lsp}(\mathbf{\Gamma}, \mathbf{X}) = \text{tr}(\mathbf{\Gamma}\mathbf{L}\mathbf{\Gamma}^\top) + \frac{1}{2}\text{tr}(\mathbf{\Gamma}\mathbf{B}\mathbf{\Gamma}^\top)$ with simple algebraic operations and reformulate Equation 4.41 as:

$$\begin{array}{ll} \min\limits_{\mathbf{\Gamma}} & \|\mathbf{\Gamma}\|_* + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{\Gamma}\|_F^2 + \mu\text{tr}(\mathbf{\Gamma}\hat{\mathbf{L}}\mathbf{\Gamma}^\top) \\ \text{s.t.} & \mathbf{\Gamma}^\top\vec{\mathbf{1}} = \vec{\mathbf{1}}, \ \gamma_{ij} \geq 0, \ \ \gamma_{ii} = 0 \ , \ \forall ij \end{array} \tag{4.42}$$

where tr(.) is the trace operator and $\hat{\mathbf{L}} = (\mathbf{L} + \frac{1}{2}\mathbf{B})$. The objective of Equation 4.42 is the sum of convex functions (trace, inner-product, and convex norms). Hence, the optimization problem is a constrained convex problem and can be solved using the alternating direction method of multipliers (ADMM) (Boyd et al. 2011) as presented in Algorithm 4.7. Optimizing Equation 4.42 coincides with minimizing the following

Algorithm 4.7 Optimization Scheme of NLSSC

```
1: Input: $\mathbf{X}, \lambda, \mu, k, \Delta_\rho = 0.1, \rho_{max} = 10^6$.
2: Output: Sparse coefficient matrix $\mathbf{\Gamma}$
3: Initialization: Compute $\{\mathbf{W}, \mathbf{B}, \hat{\mathbf{L}}\}$. Set all $\{\mathbf{\Gamma}_+, \mathbf{\Gamma}, \mathbf{U}, \alpha_+, \alpha_U, \vec{\alpha}_\mathbf{1}\}$ to zero
4: do
5:     Updating $\mathbf{\Gamma}$ by solving Equation 4.44.
6:     Updating $\mathbf{U}$ based on (J.-F. Cai, Candès, and Z. Shen 2010)(Equation 2.2).
7:     Updating $\mathbf{\Gamma}_+, \alpha_+, \alpha_U, \vec{\alpha}_\mathbf{1}$ based on Equation 4.45.
8: while Convergence criteria of Equation 4.46 is not met
```

augmented Lagrangian, which is derived by adding its constraints as penalty terms in the objective function.

$$\begin{aligned}\mathcal{L}_\rho \quad & (\mathbf{\Gamma}, \mathbf{\Gamma}_+, \mathbf{U}, \alpha_+, \alpha_U, \vec{\alpha}_\mathbf{1}) = \|\mathbf{U}\|_* + \lambda\mathcal{E}_{rep}(\mathbf{X}, \mathbf{\Gamma}) + \mu\mathcal{E}_{lsp}(\mathbf{X}, \mathbf{\Gamma}) \\ & + \tfrac{\rho}{2}\|\mathbf{\Gamma} - \mathbf{\Gamma}_+\|_F^2 + \mathrm{tr}(\alpha_+^\top(\mathbf{\Gamma} - \mathbf{\Gamma}_+)) + \tfrac{\rho}{2}\|\mathbf{\Gamma} - \mathbf{U}\|_F^2 \\ & + \mathrm{tr}(\alpha_U^\top(\mathbf{\Gamma} - \mathbf{U})) + \tfrac{\rho}{2}\|\mathbf{\Gamma}^\top\vec{\mathbf{1}} - \vec{\mathbf{1}}\|_2^2 + \langle\vec{\alpha}_\mathbf{1}, \mathbf{\Gamma}^\top\vec{\mathbf{1}} - \vec{\mathbf{1}}\rangle,\end{aligned} \tag{4.43}$$

in which $\mathcal{E}_{rep} := \frac{1}{2}\|\mathbf{X} - \mathbf{X}\mathbf{\Gamma}\|_F^2$, and $(\mathbf{\Gamma}_+, \mathbf{U})$ are auxiliary matrices related to the non-negativity constraint and the term $\|\mathbf{\Gamma}\|_*$. Eq 4.43 contains the Lagrangian multipliers $\alpha_+, \alpha_U \in \mathbb{R}^{N\times N}$ and $\vec{\alpha}_\mathbf{1} \in \mathbb{R}^N$, and the penalty parameter $\rho \in \mathbb{R}^+$. Minimizing $\mathcal{L}_\rho$ Equation 4.43 is carried out in an alternating optimization framework, such that at each step of the optimization, all of the parameters $\{\mathbf{\Gamma}, \mathbf{\Gamma}_+, \mathbf{U}, \alpha_+, \alpha_U, \vec{\alpha}_\mathbf{1}\}$ are fixed except one. Therefore, the updating steps are described as follows.
Updating $\mathbf{\Gamma}$: At iteration t of ADMM, via fixing $(\mathbf{\Gamma}_+^t, \mathbf{U}^t, \alpha_+^t, \alpha_U^t, \vec{\alpha}_\mathbf{1}^t)$, the matrix $\mathbf{\Gamma}^{t+1}$ is updated as the solution to this Sylvester linear system of equations (Kirrinnis 2001)

$$[2\lambda\mathbf{X}^\top\mathbf{X} + 2\rho\mathbf{I} + \vec{\mathbf{1}}\vec{\mathbf{1}}^\top]\mathbf{\Gamma}^{t+1} + \mathbf{\Gamma}^{t+1}[2\mu\hat{\mathbf{L}}] = \rho[\mathbf{\Gamma}_+^t + \mathbf{U}^t + \vec{\mathbf{1}}\vec{\mathbf{1}}^\top] - \alpha_\mathbf{U}^t - \alpha_+^t - \vec{\mathbf{1}}\vec{\alpha}_\mathbf{1}^{t\,\top} \tag{4.44}$$

Updating $\mathbf{U}$: Updating $\mathbf{U}^{t+1}$ which is associated with $\|\mathbf{\Gamma}\|_*$ can be done via fixing other parameters and using the singular value thresholding method (J.-F. Cai, Candès, and Z. Shen 2010) as $\mathbf{U}^{t+1} = \mathcal{T}_{1/\rho}(\mathbf{\Gamma})$ where term $\mathcal{T}(.)$ is the thresholding operator from (J.-F. Cai, Candès, and Z. Shen 2010)(Equation 2.2).
Updating $\mathbf{\Gamma}_+, \alpha_+, \alpha_U, \vec{\alpha}_\mathbf{1}, \rho$: The matrix $\mathbf{\Gamma}_+$ and the multipliers are updated using the following projected gradient descent and gradient ascent steps

$$\begin{aligned} \mathbf{\Gamma}_+^{t+1} &= \max(\mathbf{\Gamma} + \tfrac{1}{\rho}\alpha_+^t, 0), & \alpha_+^{t+1} &= \alpha_+^t + \rho(\mathbf{\Gamma} - \mathbf{\Gamma}_+) \\ \vec{\alpha}_\mathbf{1}^{t+1} &= \vec{\alpha}_\mathbf{1}^t + \rho(\mathbf{\Gamma}^\top\vec{\mathbf{1}} - \vec{\mathbf{1}}), & \rho^{t+1} &= \min(\rho^t(1 + \Delta_\rho), \rho_{max}) \end{aligned} \tag{4.45}$$

in which $(\Delta_\rho, \rho_{max})$ are the update step and higher bound of ρ, respectively.
Convergence Criteria: The algorithm reaches its convergence point when for a fixed $\epsilon > 0$ we have

$$\begin{aligned} \|\mathbf{\Gamma}^t - \mathbf{\Gamma}^{t-1}\|_\infty &\leq \epsilon, & \|\mathbf{\Gamma}_+^t - \mathbf{\Gamma}^t\|_\infty &\leq \epsilon \\ \|\mathbf{U}^t - \mathbf{\Gamma}^t\|_\infty &\leq \epsilon, & \|\mathbf{\Gamma}^{t\,\top}\vec{\mathbf{1}} - \vec{\mathbf{1}}\|_\infty &\leq \epsilon \end{aligned} \tag{4.46}$$

Optimizing NLKSSC:

As mentioned in Section 4.4, the kernel-based extension of the NLSSC model (NLKSSC) is also optimized by using Algorithm 4.7. However, the kernel trick $\Phi(\vec{x}_i)^\top\Phi(\vec{x}_j) = \mathcal{K}(\vec{x}_i, \vec{x}_j)$ should be applied in advance to replace $\mathbf{X}^\top\mathbf{X}$ of the optimization steps by matrix $\mathcal{K}(\mathcal{X}, \mathcal{X})$ in Equation 4.44.

4.5 EXPERIMENTS

In this section, I evaluate the performance of my proposed Kernel-based sparse coding frameworks respecting the interpretability and discriminative quality of the resulting encodings. I implement my approaches on real-world motion datasets, where LC-NNKSC and CKSC are evaluated in a supervised setting while the performance of NLKSSC is determined by unsupervised (clustering) measures.

Supervised Setting

In this section, I empirically evaluate the performance of the proposed supervised sparse coding frameworks LC-NNKSC and its improved version CKSC w.r.t. to the enriched encoding of mocap data.

For experiments, I consider the following datasets: Schunk , DynTex++ , Dance , UTKinect , HDM05 , and CMU-9, introduced in Section 2.4. Except for the last two datasets, which are generally multi-dimensional time-series, other selected datasets are human motion capture benchmarks. For these datasets, I compute the kernel representations based on the pairwise DTW distance between motion sequences. Specifically, I employ the global alignment kernel (GAK) (Cuturi et al. 2007), which guarantees the resulting $\mathcal{K}(\mathcal{X},\mathcal{X})$ is PSD without performing any manual eigenvalue correction to the resulting Gramm matrix. Due to this change of kernel function, the results could be in some cases better than the reported ones in my relevant publication (Hosseini and Hammer 2018a). Exceptionally, the kernel for DynTex++ is computed as described in (Quan, Bao, and H. Ji 2016). Also, prior to the application of GAK on `Utkiect`, I use the preprocessing from (Vemulapalli, Arrate, and Chellappa 2014) to obtain the Lie Group representation.

Parameters Tuning

In order to tune the parameters α and T related to the optimization frameworks of CKSC in (Equation 4.20 , Equation 4.21) and LC-NNKSC in Equation 4.18, I perform 5-fold cross-validation using train and validation sets. I carry out the same procedure for the baselines to find their optimal choice of parameters. The parameter k (dictionary size) is determined as :

$$k = \{\#\ \text{classes}\} \times T$$

However, as a working parameter setting for CKSC and LC-NNKSC in practice, we can choose a value around $\alpha = 0.1$. Parameter T (and the dictionary size) generally depends on class distributions and the complexity of the dataset. However, based on empirical evidence choosing big T does not improve the performance of CKSC and just increases the dictionary redundancy.

Alternative Methods

To properly analyze the performance of my kernel-based sparse coding algorithms (LC-NNKSC and CKSC), I select the following alternatives among kernel-based discriminative sparse codings: K-KSVD (H. V. Nguyen et al. 2013), JKSR (Huaping Liu, D. Guo, and F. Sun 2016), LC-KKSVD (Z. Chen et al. 2015), LP-KSVD (W. Liu et al. 2015), KGDL (Harandi et al. 2013), and EKDL (Quan, Bao, and H. Ji 2016). These algorithms are

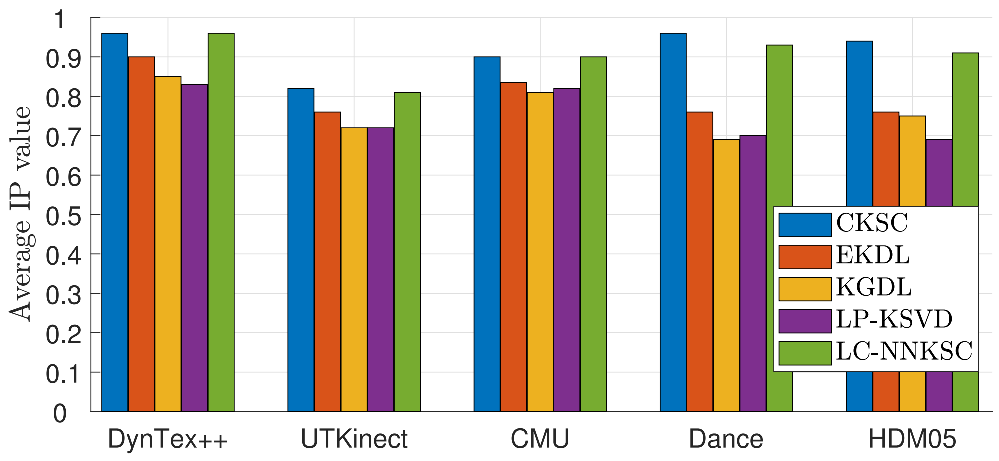


Figure 4.5: Average *IP* value for UTKinect , DynTex++ , CMU-9, UTKinect , and HDM05 datasets.

known as state-of-the-art kernel-based sparse coding algorithms, which learn dictionaries for sparse and discriminative data representation provided the input's kernel-based representation.

I empirically compare the proposed methods based on the interpretability of sparse encodings and their discriminative quality, and for the basis of evaluations, I use 10-fold cross-validation averaged over 10 repetitions. It is important to emphasize that the purpose of my sparse coding frameworks is to obtain a **discriminative sparse encoding** of motion sequences based on its precomputed **kernel representation**. Therefore, instead of comparing the results to all the available top classifiers in the literature such as Deep Neural Networks and others, I only select the recent kernel-based alternatives which fit the above description.

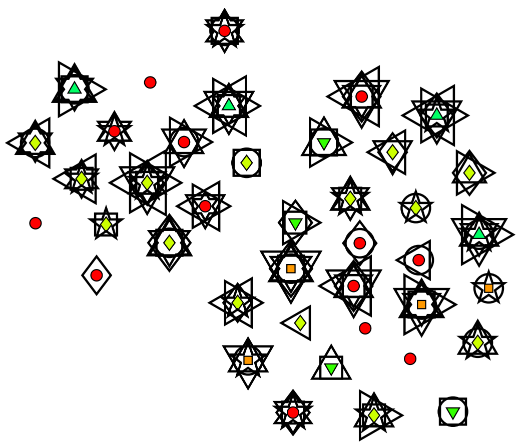
Interpretability of the Encoding

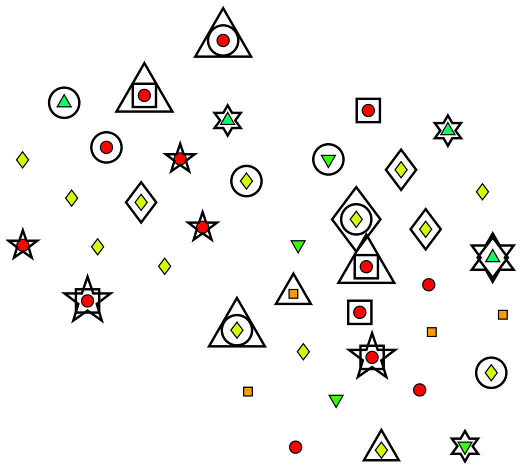
As discussed before, by using non-negativity constraints, I aimed to obtain an encoding model that is more interpretable in terms of its constituent building blocks. Already in Figure 4.3, we observed the effect of the non-negativity constraint on the sparseness of the resulting model. Although both NNKSC and its constraint-free counterpart K-KSVD were trained to obtain the same $\|\Phi(\mathbf{X}) - \Phi(\mathbf{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$ as the reconstruction error, NNKSC uses less training samples and dictionary atoms to shape the encoding vectors. Therefore, it would be much easier to trace back from an encoded motion vector $\vec{\gamma}_i$ to other motion samples to which $\mathbf{X}_i$ is related.

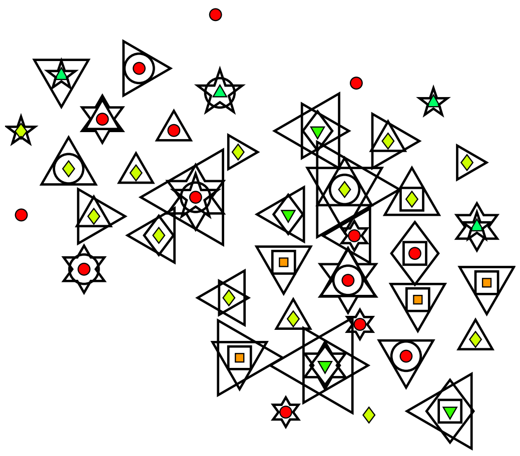
Apart from the sparseness level of the model, it is desired to have dictionary atoms that could be assigned to mostly one class of data. Having such a model, we can interpret an end encoding $\vec{\gamma}_i$ based on the motion type(s) to which $\mathbf{X}_i$ belongs. To measure such characteristic in the sparse coding models, I define the interpretability measure IP_i for each $\vec{u}_i$ as

$$IP_i = \max_j (\vec{\rho}_j^\top \vec{u}_i) / (\vec{\mathbf{1}}^\top \mathbf{H} \vec{u}_i),$$

where $\vec{\mathbf{1}} \in \mathbb{R}^C$ is a vector of ones. IP_i becomes 1 if $\vec{u}_i$ uses data instances related only to one specific class. Figure 4.5a presents the IP value for the algorithms CKSC, EKDL, LC-NNKSC , KGDL, and LP-KSVD related to their implementations on the datasets DynTex++ , CMU-9, UTKinect , UTKinect , and HDM05 . Based on the results, CKSC and


(a) The Kernel-KSVD model.


(b) The NNKSC model.


(c) The EKDL model.

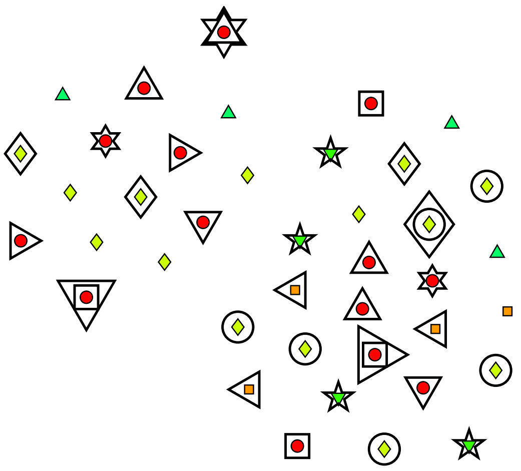

(d) The CKSC model.

Figure 4.6: Studying the contribution of training samples in the formation of 9 dictionary atoms for (a) K-KSVD, (b) NNKSC , (c) EKDL, and (d) CKSC algorithms on a dense neighborhood in the HDM05 dataset. Each small shape is a training sample related to one class of data, and each type of big shape represents one dictionary atom $\Phi(\mathcal{X})\vec{u}_i$ and indicates the training samples on which it is built corresponding to the non-zero entries of $\vec{u}_i$.

LC-NNKSC achieved the highest *IP* values as they use similar non-negative constraints in their training frameworks. Among other methods, EKDL presents better interpretability results due to the incoherency term it uses between the dictionary atoms $\vec{u}_i$.

Figure 4.6 visualizes the formation of dictionary atoms based on non-zero entries of columns of $\mathbf{U}$ for the HDM05 dataset. I already zeroed the relatively small entries of each $\vec{u}_i$ such that the remaining coefficients point toward significant training samples for the given dictionary atom. As we can observe for the K-KSVD model (Figure 4.6-a), each $\vec{u}_i$ uses several data points from various classes and with both positive and negative coefficients. The *IP* value for this model on the HDM05 dataset is 0.32. Hence, we cannot semantically relate any of $\Phi(\mathcal{X})\vec{u}_i$ atoms to an specific class of data (motion type).

On the other hand, dictionary vectors of NNKSC model (Figure 4.6-b) are considerably sparse and compact. Each $\vec{u}_i$ is formed mostly from 4 to 6 specific training samples, which makes the dictionary atom highly transparent regarding its constituent resources. This dictionary formation corresponds to the IP value of 0.87 for NNKSC . Despite that quality, the $\vec{u}_i$ vector takes contributions from 2 or 3 different motion classes in several cases. This observation happens because NNKSC uses no supervised information for the optimization of its dictionary atoms. Hence, we may observe contributions from samples of different motion classes that are in a relatively close neighborhood of each other. For example, in the NNKSC model, one $\vec{u}_i$ may use samples from 2 or 3 close leg-movement motion classes. In contrast, this combination in the K-KSVD model typically occurs between leg-movement, hand-gesture, and dance classes.

In comparison, the dictionary vectors of the EKDL method have more overlap regarding their contributing classes (Figure 4.6-c). This cross-class contribution for the formation of each $\vec{u}_i$ is still smaller compared to K-KSVD due to the discriminative structure of EKDL, resulting in an IP value of 0.76. However, the dictionary atoms are generally not interpretable as they are for NNKSC . On average, each $\vec{u}_i$ is connected to data samples from 3 different classes, and almost all the sequences are used to construct the dictionary matrix $\mathbf{U}$.

As depicted in Figure 4.6-d, the CKSC model provides dictionary vectors $\vec{u}_i$ with a similar sparseness level as NNKSC . However, its dictionary atoms are mostly related to one type of motion data among the training samples. As a result, when a $\vec{\gamma}_i$ encodes a motion sample $\Phi(\mathbf{X}_i)$ using the *square*, *hexagram*, and *upright triangle* dictionary atoms, we can interpret that $\Phi(\mathbf{X}_i)$ has the characteristic of the *circle* class of motion. Accordingly, the IP value of CKSC for this dataset is 0.94, which is the highest among other methods.

Discriminative Quality of the Encoding

In addition to the sparseness of the encoded vectors and the interpretability of dictionary atoms, we are interested to see how the sparse encoding respects the labeling of data. In other words, in an interpretable sparse model, the encoded vector $\vec{\gamma}_i$ would represent the motion sample $\mathbf{X}_i$ using resources (columns of $\mathbf{U}$) that links $\mathbf{X}_i$ majorly to training data of its own type (class).

By the specific way that I determined the label of each test data in Equation 4.25, we can use the classification accuracy of test data to measure the above characteristic of the sparse models directly. To that aim, the classification accuracy of the implemented sparse coding approaches is measured as $Acc = (100 \times [\texttt{\#correct predictions}]/N)$ and reported in Table 4.1, where N is here the total number of test data.

According to the given results, my CKSC algorithm obtains the highest classification performance for all of the benchmarks, which shows that the designed frameworks of Equation 4.20 and Equation 4.21 provide better discriminative representations compared to other K-SRC algorithms.

CKSC, EKDL, LP-KSVD, and LC-NNKSC can be considered the runner-up group with competitive classification accuracy. This observation is due to the embedded labeling information in their discriminant terms, which improves their discriminative representations in contrast to K-KSVD, JSRC, and KGDL. Nevertheless, there is a variation in the comparison results of these methods. I can conclude that although they use different strategies to obtain a discriminant model, their recall model does not necessarily comply with their training model. In contrast, CKSC demonstrated a more efficient embedding of the supervised information in both the training and the recall framework.

Table 4.1: Average classification accuracies (%) ± standard deviations for the selected datasets.

Datasets	K-KSVD	JKSRC	LC-NNKSC	LP-KSVD
Schunk	83.42±0.35	87.49±0.57	89.96±0.64	89.62±0.51
CMU-9	82.62±1.02	83.68±0.79	90.94±0.67	90.21±0.57
DynTex++	89.22±0.47	89.95±0.35	93.22±0.37	93.12±0.47
Dance	92.26±0.78	91.43±0.69	96.46±0.68	96.51±0.71
UTKinect	84.38±0.31	85.67±0.44	88.43±0.30	89.35±0.32
HDM05	83.91±0.92	87.44±0.56	88.12±0.45	87.46±0.41
	KGDL	EKDL	**CKSC**	
Schunk	88.17±0.43	88.39±0.24	**91.42±0.34**	
CMU-9	87.34±0.87	90.88±0.71	**92.68±0.79**	
DynTex++	92.83±0.31	93.51±0.46	**94.36±0.32**	
Dance	94.74±0.58	95.37±0.76	**97.75±0.53**	
UTKinect	88.18±0.29	89.02±0.27	**90.97±0.24**	
HDM05	88.85±0.58	88.31±0.30	**91.87±0.43**	

The best result (**bold**) is according to a two-valued t-test at a 5% significance level.

LC-NNKSC uses a non-negative framework similar to the basis of CKSC's structure; additionally, by comparing the results of LC-NNKSC to those of LP-KSVD and EKDL, we observe that its non-negative framework obtains a competitive performance. Although LP-KSVD and EKDL employ extra objective terms in their models, the non-negative structure of LC-NNKSC can achieve a similar outcome accuracy without the need to use such extra terms. Relevantly, CKSC benefits from this non-negative optimization framework as a basis for its confidence-based model, leading to its superior performance compared to other baselines.

Putting the above results next to the interpretability performance of the methods (Figure 4.5), I conclude that the CKSC algorithm learns a highly interpretable dictionary atoms $\vec{u}_i$, while also providing an efficient discriminative encoding of motion sequences.

Effect of the Parameter Setting

In order to study the sensitivity of CKSC to the parameter settings, I carry out experiments via changing the algorithm's parameters (α, T). Implementing on the Schunk dataset, I apply CKSC in 2 individual settings via changing one parameter throughout each experiment when the other one is fixed. As observed in Figure 4.7-a, the right choice for α lies in the interval $[0.1, 0.4]$. However, the discriminative objective can outweigh the reconstruction part when α is close to 1, and it results in over-fitting and performance reductions.

Regarding the dictionary size, I increase T from 1 to 20 with a step-size of 1, which changes the size of $\mathbf{U}$ in the range $[20, 400]$ with step-size 20 (average number of data samples per class). According to Figure 4.7-b, having T between 4 and 8 keeps the performance of CKSC at an optimal level for the Schunk dataset. As it is clear, small values of T put a tight limit on the number of available atoms $\vec{u}_i$ for reconstruction purpose which reduces the accuracy of the method. On the other hand, larger values of T increase dictionary redundancy and loosen up the sparseness bound on $\mathbf{\Gamma}$; nevertheless, NQP algorithm and the non-negativity constraints intrinsically incur sparse characteristics

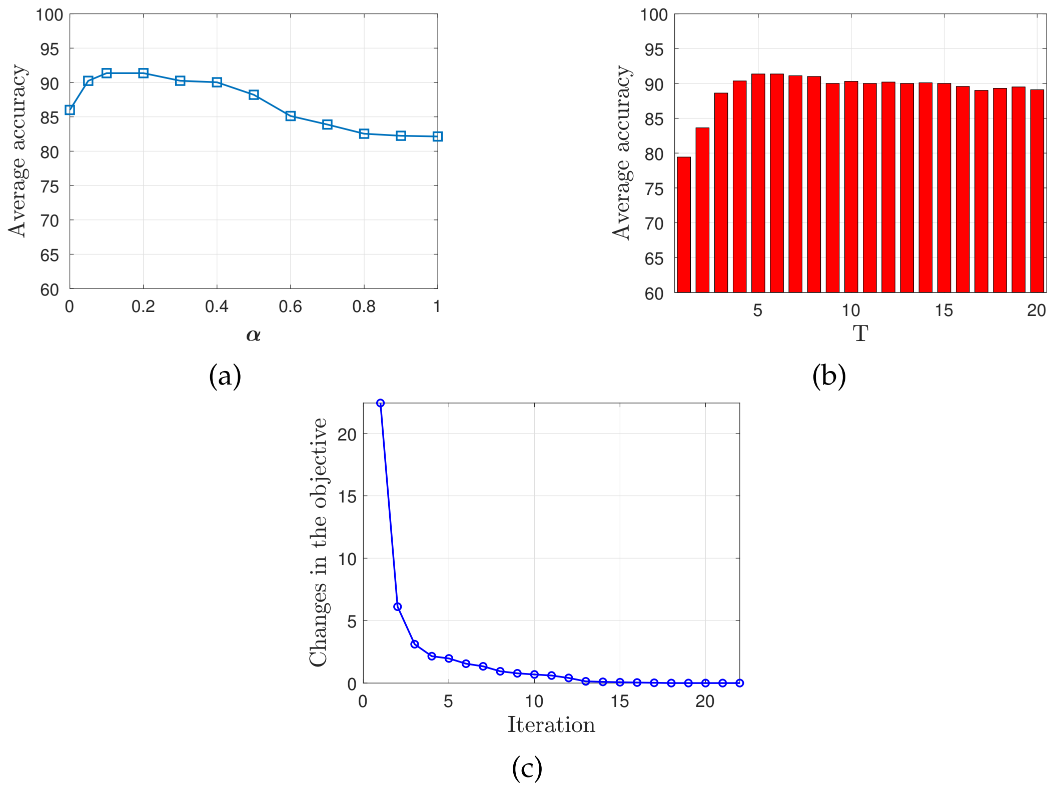


Figure 4.7: Effect of changes in CKSC 's hyper-parameters α (a) and T (b), and the convergence curve (c) of the algorithm for Schunk dataset.

to $\mathbf{\Gamma}$ and $\mathbf{U}$ via combining together only the most similar resources. Therefore, increasing T does not degrade the performance of CKSC in a dramatic way.

Complexity and Convergence of CKSC

To calculate the computational complexity of CKSC per iteration, I analyze the update of $\{\mathbf{\Gamma}, \mathbf{U}\}$ separately. In each iteration, $\mathbf{\Gamma}$ and $\mathbf{U}$ are optimized using the NQP algorithm, which has the computational complexity of $\mathcal{O}(n\frac{T^2}{2})$ (based on section 4.3), where T and n are the sparsity limit and size of $\mathbf{Q}$ in Algorithm 4.5. Therefore, optimizing $\mathbf{\Gamma}$ and $\mathbf{U}$ in each iteration takes $\mathcal{O}(kN\frac{T^2}{2} + (2k+C)N^2 + kCN)$ and $\mathcal{O}(kN\frac{T^2}{2} + (8k+C)N^2)$ steps, respectively. In the above computation, N, k, and C denote the number of training samples, the dictionary size, and the number of classes, respectively.

As shown in Figure 4.7b, we can practically choose $T \leq 10$, and also we have $C \ll k$. Therefore, the dominant run-time complexity for one iteration of Algorithm 4.4 is $\mathcal{O}(8kN^2)$, which is mainly due to the matrix multiplications as the pre-optimization step for updating $\mathbf{U}$ in Equation 4.29. Nevertheless, for datasets that N/C is relevantly large, the size of $\mathbf{U}$ should be chosen such that $k \ll N$. Otherwise, it increases the redundancy in the dictionary without having any added-value.

On the other hand, the optimization framework of CKSC in Equation 4.27 is non-convex when considering $\{\mathbf{U}, \mathbf{\Gamma}\}$ together. However, each of the sub-problems defined in Equation 4.29 and Equation 4.30 are convex. Therefore, the alternating optimization scheme in Algorithm 4.4 converges in a limited number of steps. Consequently, in practice, the approximate computational complexity of CKSC becomes $\mathcal{O}(N^2)$, especially for large datasets. Using the information given in Section 4.2 about the computational complexity of NN-KOMP and NN-KFISTA in optimizing the LC-NNKSC algorithm (as my primary supervised K-SRC model), we can show that it has a similar computational complexity of $\mathcal{O}(N^2)$ under the above implementation setting.

For instance, Figure 4.7c shows the convergence curve of CKSC for the Schunk dataset based on the changes in the value of the objective term in Equation 4.29. According to this curve, the algorithm reaches a stationary point in a reasonable number of iterations (15 total iterations).

Unsupervised Encoding

In this section, I evaluate the performance of my NLKSSC algorithm w.r.t. the quality of the resulting self-representative encoded vectors in revealing the underlying subspaces in the mocap dataset.

For the implementation of NLKSSC , I select the following datasets: Schunk , Cricket , UTKinect , Words , and CMU-9, which are explained in Section 2.4. All selected datasets are human motion capture benchmarks (full body or partial body) except the first one, which is a general multi-dimensional time-series. Additionally, the performance of NLSSC respecting vectorial data is reported in my relevant publication (Hosseini and Hammer 2018c) by its application on other real benchmark datasets. Similar to the supervised experiments, the kernel matrix $(\mathcal{X}, \mathcal{X})$ is computed based on pairwise DTW distance between motion sequences.

Parameter Setting

In order to tune the parameters λ, μ, k for NLKSSC , I utilize a grid-search method. I do the search for λ in the range of $\{1, 1.5, ..., 7\}$, for μ in the range of $\{0.1, 0.2, ..., 1\}$ and k in $\{3, 4, ..., 8\}$. I implement a similar parameter search for the baselines to find their best settings. Although for the link-restore parameter, $\tau = 0.2$ generally works well, one can do a separate grid-search for τ.

Alternative Methods

I compare my algorithms' performance to baseline methods KSC (Von. Luxburg 2007), NNKSC (Section 4.2), KSSC (Patel and René Vidal 2014), KSSR (Bian, F. Li, and X. Ning 2016) and RKNNLRS (S. Xiao et al. 2016). These algorithms are selected from major sparse coding-based clustering approaches applied to kernel information.

The evaluation basis is the clustering error as $CE = \frac{\text{\# of miss-clustered samples}}{\text{Total samples}}$ using the posterior labeling of the clusters (Hammer, Hasenfuss, et al. 2007) and the normalized mutual information (NMI) (Ana and Jain 2003). For each method, an average CE is calculated over 10 runs of the algorithm. As an external measure, CE can be approximately compared to $1 - \frac{Acc}{100}$ values of the supervised experiments in the previous sub-section. NMI measures the amount of information shared between the clustering result and the ground-truth which lies in the range of $[0, 1]$ with the ideal score of 1. As mentioned in Section 4.4, the label information of the dataset has no role in the decision-making process of the NLKSSC algorithm. However, I only use that information as the ground truth to evaluate the method's clustering performance.

Table 4.2: Average clustering error (CE) and NMI for CMU, Words, Cricket, UTKinect, and Schunk datasets.

Dataset	KSC		KSSC		KSSR	
	CE	NMI	CE	NMI	CE	NMI
CMU-9	0.2715	0.7206	0.2266	0.7429	0.2454	0.7355
Words	0.1921	0.8436	0.1564	0.8725	0.1873	0.8359
Cricket	0.3042	0.7442	0.2542	0.7041	0.2616	0.7228
UTKinect	0.3315	0.6724	0.2717	0.7318	0.3156	0.6776
Schunk	0.3194	0.6652	0.2974	0.7273	0.3066	0.7374
	RKNNLRS		**NNKSC (Algorithm 4.3)**		**NLKSSC (proposed)**	
	CE	NMI	CE	NMI	CE	NMI
CMU-9	0.2016	0.7516	0.2287	0.8150	**0.1723**	**0.8623**
Words	0.1424	0.8636	0.1613	0.8022	**0.0947**	**0.8725**
Cricket	0.2478	0.7984	0.2846	0.7083	**0.2073**	**0.8066**
UTKinect	0.2732	0.7138	0.2841	0.7624	**0.2314**	**0.8011**
Schunk	0.2621	0.7741	0.2962	0.7456	**0.1929**	**0.7954**

The best result (**bold**) is according to a two-valued t-test at a 5% significance level.

As explained in Section 4.4, each algorithm's sparse codes are used to construct the corresponding sparse representation graph $\mathcal{G}$ with weighted connections (Equation 4.39). However, I use $\mathbf{A} = \mathbf{\Gamma}^\top \mathbf{\Gamma}$ for the NNKSC method as its $\mathbf{\Gamma}$ is not symmetric. The spectral clustering step of the baselines is performed via using the correct number of clusters. For KSC as the kernel-based spectral clustering baseline, I specifically use the kernel matrix $\mathcal{K}(\mathcal{X}, \mathcal{X})$ directly instead of $\mathbf{\Gamma}$ in Equation 4.39 to compute the adjacency matrix.

Clustering Results

According to the results summarized in Tables. (4.2), the proposed subspace clustering algorithm NLKSSC outperformed the benchmarks regarding the clustering error. This result supports my claim regarding the effect of the specific clustering-based formulation I proposed in Equation 4.40 in finding the underlying subspaces in the data. The KSC algorithm obtains the minimum accuracy for all datasets and is treated as the baseline for evaluating other methods. The clustering result of my NNKSC algorithm shows that the non-negative dictionary-based structure of this framework can also be effective compared to the KSSR method or KSSC (for CMU and Schunk).

On the other hand, RKNNLRS performance shows that its non-negative model is more practical for the clustering goal compared to the NNKSC method. This evidence suggests that having a subspace-based structure and low-rank characteristic is more successful than a dictionary-based representation in an unsupervised setting. By comparing NLSSC (the proposed algorithm) to other algorithms with low-rank regularizations in their models, I can conclude that the proper combination of the locality term and the affine constraints has aided NLKSSC to obtain higher performance. Also, it is clear from Table. 4.2 that the KSSR algorithm's accuracy is close to the baseline method KSC for all datasets. This weak performance is due to the lack of any strong regularization term in its model regarding the subspace structure of data.

Since CE is chosen as an external clustering measure, one can compare the perfor-

Table 4.3: Application of the link-restore method on the non-negativity based approaches.

Dataset	RKNNLRS		**NNKSC$_{\text{(proposed)}}$**		**NLKSSC $_{\text{(proposed)}}$**	
	CE	*NMI*	*CE*	*NMI*	*CE*	*NMI*
CMU	0.1875	0.7684	0.2036	0.8117	**0.1572**	**0.8355**
Words	0.1369	0.8203	0.1658	0.8357	**0.0926**	**0.9536**
Cricket	0.2234	0.7830	0.2574	0.7870	**0.1846**	**0.8004**
UTKinect	0.2473	0.7522	0.2656	0.7393	**0.2185**	**0.7527**
Schunk	0.2592	0.7706	0.2993	0.7366	**0.1914**	**0.7629**

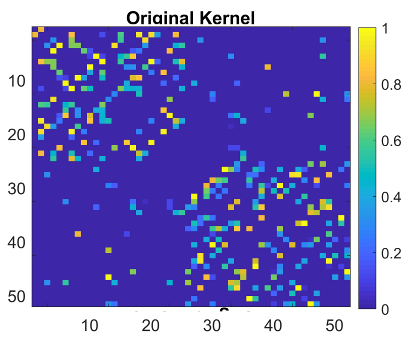
The best result (**bold**) is according to a two-valued t-test at a 5% significance level.

mance of unsupervised methods to that of supervised approaches from Table 4.1. By loosely considering $CE \approx 1 - \frac{Acc}{100}$, we observe that all supervised methods (Table 4.1) outperform the clustering methods of Table 4.2 for CMU-9, Schunk , and UTKinect datasets due to their access to the supervise information during their training phase. Nevertheless, we can conclude that the proposed NLKSSC clustering method and the supervised K-KSVD algorithm have comparable accuracies for CMU-9 dataset.

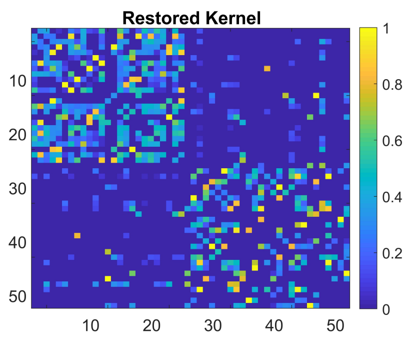
On the other hand, both NNKSC and K-KSVD are unsupervised methods, and NNKSC has a better class-specific dictionary formation (Figure 4.6). However, the reason behind the wide distances between the classification performance of these two methods is their labeling strategy. While the label assignment for NNKSC in experiment of Table 4.2 is performed by applying spectral clustering on the sparse codes, the K-KSVD results of Table 4.1 are based on applying the SVM classifier on the sparse vectors, which is heavily biased by the supervised information of the training set.

Effect of Link-Restore

To investigate the effect of the proposed link-restore algorithm, I apply it to the non-negative K-SSC methods RKNNLRS, NNKSC , and NLKSSC as a post-processing step. This selection is based on the fact that link-restore is designed based on the non-negativity assumption about columns of $\mathbf{\Gamma}$. According to Table 4.3, the application of link-restore was effective regarding in all cases. It reduced the clustering error of all the relevant methods to some extent, demonstrating its ability to correct broken links in the representation



(a)



(b)

Figure 4.8: A subset of the affinity matrix resulted from the implementation of NLKSSC on the Words dataset: (a) Before application of link-restore. (b) After the application of link-restore.

graph $\mathcal{G}$.

Nevertheless, its effect on the NLKSSC method varies among the utilized datasets. For Words and Schunk datasets, the post-processing method did not add any non-trivial link to the graph $\mathcal{G}$, which consequently did not change the value of CE. However, for CMU, Cricket, and UTKinect datasets, the amount of decreases in CE shows the effectiveness of link-restore in correcting the missing connections in $\mathcal{G}$.

In addition, Figure 4.8 visualizes the affinity matrix for the implementation of NKLSSC on the Words dataset. The figure is zoomed in on clusters showing that the representation graph contains more intra-cluster connections after applying link-restore (figure 4.8-b). It is clear from this figure that the encoding vectors are more interpretable after reviving more significant connections by the application of link-restore.

Sensitivity to the Parameter Settings

I study the sensitivity of NLKSSC to the choice of parameters for the UTKinect dataset (with the highest CE in Table. 4.2 by implementing 3 different experiments. In each experiment, I fix two parameters from $\{\lambda, \mu, k\}$, and change the other one while study the effect of this variation on clustering error (CE). Based on Figure 4.9, the algorithm sensitivity to λ is acceptable when $2 \leq \lambda \leq 4.5$. Having $\lambda \geq 6$ does not change CE since it makes the loss term $\mathcal{E}_{rep} := \|\Phi(\mathbf{X}) - \Phi(\mathbf{X})\mathbf{\Gamma}\|_F^2$ more dominant in the optimization problem of Equation 4.40.

By choosing $0.25 \leq \mu \leq 0.5$, the algorithm's performance does not change drastically. However, NLKSSC shows a considerable sensitivity if μ goes beyond 0.6. High values of μ weaken the role of $\mathcal{E}_{rep}$ (the primary loss term) in the sparse coding model.

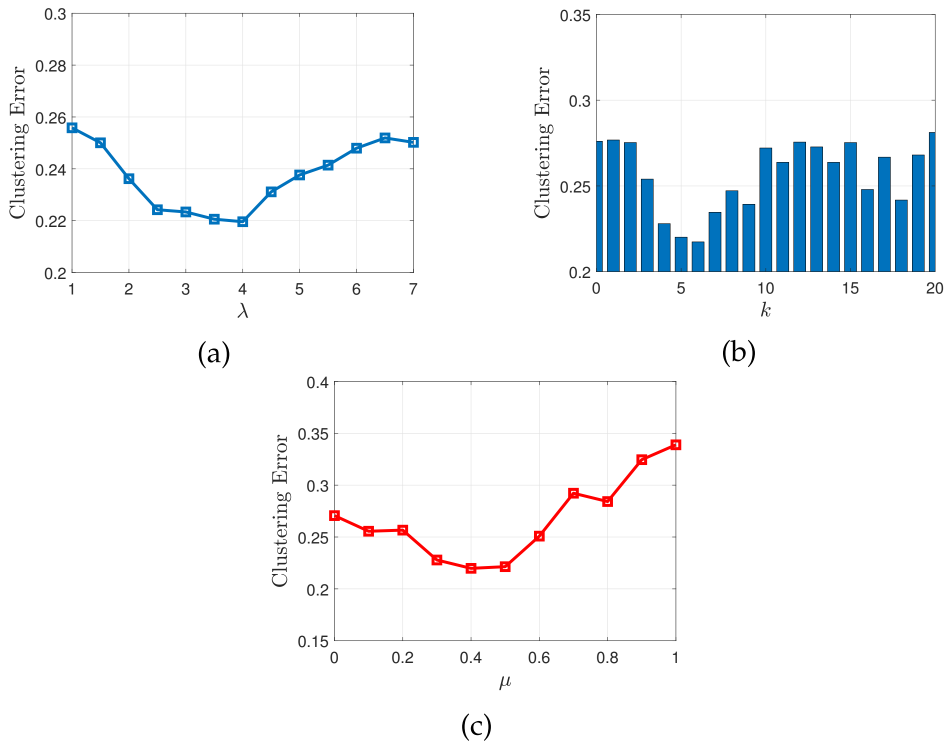


Figure 4.9: Sensitivity analysis of NLKSSC to parameter selection (a)λ, (b)μ, and (c)k for the UTKinect dataset.

Studying the sensitivity curve of k shows that its starting point has a similar CE to the start of μ sensitivity curve, as in both cases effect of $\mathcal{E}_{lsp}$ becomes zero in the optimization. Figure 4.9-b shows that $k \in \{3, 4, 5\}$ is a proper choice. However, with $k \leq 3$ the objective term $\mathcal{E}_{lsp}$ is not effective enough. On the other hand, with $k \geq 10$ the CE curve does not follow any constant pattern but generally becomes worse because large values of k removes the local effect of $\mathcal{E}_{lsp}$ in Equation 4.35. It is important to note that even a small neighborhood radius (e.g., $k = 4$) can significantly impact the global representation if the local neighborhoods overlap. Generally, similar sensitivity behaviors are also observed for the other datasets.

4.6 CONCLUSION

This chapter proposed a novel framework for embedding mocap data into the vector space, which is sparse and semantically interpretable. More specifically, I presented a novel non-negative kernel-based sparse coding frameworks NNKSC , LC-NNKSC , CKSC , and NLKSSC for the sparse encoding of motion sequences. The NNKSC provides the basis dictionary-based model to obtain such interpretable encoding of motion sequences, while LC-NNKSC and CKSC methods extend it to the supervised setting. On the other hand, the proposed NLKSSC algorithm employs the concept of non-negative sparse encoding to obtain unsupervised enriched embedding of the mocap dataset.

Given a kernel-based representation of motions is available (e.g., via a pairwise distance matrix), the proposed NNKSC method constructs its dictionary atoms via linear combinations of motions samples in the features space. The non-negativity sparse model forces each dictionary atom to be constructed from other sequences of similar type. In addition, each motion sample is encoded in NNKSC by dictionary atoms that are semantically similar to the input motion sequence. Therefore, the NNKSC model is interpretable in terms of its dictionary atoms and the resulting sparse codes. This characteristic makes this non-negative framework suitable for obtaining encoding of motion sequence, which is interpretable through its meaningful (motion-based) building blocks.

In order to enrich the sparse encoding with supervised information such as data labeling, I extended NNKSC to two novel discriminative K-SRC frameworks LC-NNKSC and CKSC . Generally, these two sparse coding algorithms enforce the reconstruction of motion sequences in the feature space based on dictionary atoms that can be associated with specific motion classes. These models focus on obtaining an encoding that its building blocks are interpretable based on the sparsely incorporated data classes. The LC-NNKSC algorithm employs a linear discriminative objective in its optimization framework, while CKSC proposes a more robust discriminative framework. The novel framework of CKSC particularly aims for the class-based encoding of motion sequences, which leads to a better discriminative encoding.

In order to solve the respecting optimization problems of LC-NNKSC and CKSC , I proposed different optimization algorithms such as NN-KFISTA , NN-KOMP , and NQP. These methods differ in their optimization problem, the sparsity constraint they provide, and their computational complexity. In comparison, the NQP method can be applied to more general problems while being more scalable compared to NN-KOMP and NN-KFISTA . My empirical evaluations on real mocap datasets and other multivariate time-series showed that both LC-NNKSC and CKSC algorithms successfully obtain enriched sparse encodings. These algorithms outperform other relevant kernel-based sparse coding methods regarding the interpretability of their base elements and the discriminative quality of the obtained encodings. Their superior performance mainly relies on their non-negative interpretable basis framework (NNKSC) and their specific discriminative formulations.

As another contribution of this chapter, I proposed NLKSSC as a novel kernel-based subspace sparse clustering framework, which obtains self-representative encoding of a mocap dataset. My NLKSSC method encodes each motion sequence directly in terms of a sparse set of other semantically similar sequences in its local neighborhood. Such specific encoding can reveal the underlying subspaces in the data distribution when no supervised information is given. Additionally, the non-negative property of the encoding along with the proposed post-processing step enhance the interpretation of the resulted encodings. The NLKSSC framework employs a novel locality objective and low-rank, affine sparse embedding of motion sequences. Implementations on real motion benchmarks and comparison with other state-of-the-art K-SSC algorithms show that the encoded vectors resulted from NLKSSC are locally more separable in terms of the underlying motion clusters.

Furthermore, I proposed the novel link-restore post-processing algorithm to mitigate the issue of common redundancy in non-negative K-SSC encodings. This algorithm corrects the broken links between close data points in the encoding's representative graph. Empirical evaluations demonstrated that link-restore can act as an effective post-processing step for different types of K-SSC methods that use non-negative sparse coding models.

In the next chapter, I extend the proposed ideas of Chapters 3 and 4 to multiple-kernel analysis of motion data. I propose frameworks that benefit from the component-wise representation of motion data for feature selection, prototype-based representation, and encoding of unobserved motion classes.

5 MULTIPLE KERNEL LEARNING FOR MOTION ANALYSIS

Publications: This chapter is partially based on the following publications.

- Hosseini, Babak and Barbara Hammer (2019b). "Interpretable Multiple-Kernel Prototype Learning for Discriminative Representation and Feature Selection". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM.
- — (2019c). "Large-Margin Multiple Kernel Learning for Discriminative Features Selection and Representation Learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE.
- — (2019d). "Multiple-Kernel Dictionary Learning for Reconstruction and Clustering of Unseen Multivariate Time-series". In: *27th European Symposium on Artificial Neural Networks (ESANN)*.

Multiple kernel learning (MKL) algorithms utilize different data representations in the feature space (base kernels) to obtain an optimal representation upon their combination (F. R. Bach, G. R. Lanckriet, and Jordan 2004). A multiple-kernel (multiple-kernel) representation of the data can carry non-redundant pieces of information about essential properties of the data (F. R. Bach, G. R. Lanckriet, and Jordan 2004; Teng, Y.-R. Lin, and Wen 2017). We can generally formulate an MKL problem as the minimization of a loss term defined in the Reproducing Kernel Hilbert Space to fulfill a given task.

For instance, in a classification setting, this formulation aims to better separate data classes in the RKHS by finding an optimal multiple-kernel representation of data in the features space (Gönen and Alpaydın 2011). Unlike sparse coding, which optimizes the representation of data in terms of (few) basic constituents, multiple kernel learning optimizes the coordination of basic kernels such that an optimum classification result can be derived based thereon. Depending on the problem's definition, MKL can be seen as either finding the best parameter values for a specific type of kernel function (W. Jiang and Chung 2014; J. Ye, S. Ji, and J. Chen 2008; Niazmardi, Safari, and Homayouni 2017; Gönen and Alpaydın 2011) or learning a weighting vector associated to the pre-computed base kernels (Y.-Y. Lin, T.-L. Liu, and Fuh 2011; Dileep and Sekhar 2009; H. Xue, Yu Song, and H.-M. Xu 2017; P. Du et al. 2017; Z. Xu, Jin, J. Ye, et al. 2009).

MKL has shown its benefit in different data-driven applications. In image processing problems, it is a common practice to derive specific representations by utilizing different image descriptors. Therefore, an MKL algorithm can learn which descriptors provide more discriminative representations of the data classes (Y.-Y. Lin, T.-L. Liu, and Fuh 2011; Dileep and Sekhar 2009; C. Singh and J. Singh 2019; Mukundan, Tolias, and Chum 2017). In other applications such as hyper-spectral imaging, specially designed sensors can capture many narrow spectral channels of high resolution. Hence, an MKL algorithm can properly combine such sensory information for the purpose of higher classification or segmentation (Gu, Chanussot, et al. 2017; Tianzhu Liu et al. 2016; Pinar et al. 2015) As expected, another relevant domain of MKL methods is multi-variate time-series (MTS), where each dimension of the MTS data can be represented by one or several kernels.

There are different problems in this area to which MKL algorithms are applied, such as time-series predication (X. Wang and Han 2014), anomaly detection (Das et al. 2010), video processing (F. Yan et al. 2009), and pattern recognition (Sanchez-Martinez et al. 2017).

When considering the multi-component characteristic of motion data, it is convenient to construct its multiple-kernel representation. Accordingly, such basic kernels can be given by (i) kernels derived from comparisons of single joints or modalities of the motion capture, or (ii) kernels derived from a comparison to a specific observed motion. This representation is specifically intuitive when the movement of different parts of the body is considered separately. Note that, in the second case, a mathematical similarity to kernel sparse coding can be observed in particular in the case of sparse dictionaries. Therefore, several works benefited from MKL ideas to process human actions. A group of these methods relies on obtaining joint or trajectory-based kernels, where the MKL framework looks for an efficient combination of such skeleton-based information (Althloothi et al. 2014; J. Sun et al. 2009). Other methods design their input kernels based on specific video-based descriptors (Yan Song et al. 2011; Ikizler-Cinbis and Sclaroff 2010), which signify human actions from different views.

In an MKL framework, when computing each base kernel from one specific dimension (or feature) of the data, the weighting scheme of the kernels can be seen as a weighted feature selection. Such perspective becomes more notable when several entities in such a weighting vector become zero. Investigating such idea in a discriminative framework, MKL can function as a discriminative feature selection method by assigning larger weights to the most discriminative dimensions of the data (Dileep and Sekhar 2009; Z. Xu, Jin, J. Ye, et al. 2009; Varma and Babu 2009; H. Xue, Yu Song, and H.-M. Xu 2017). From a specific perspective, any MKL algorithm can be used as a multiple-kernel feature selection method, given that it takes pre-computed kernels as the inputs. In particular, MKL methods such as (Rakotomamonjy et al. 2008; Tianzhu Liu et al. 2016; Z. Xu, Jin, H. Yang, et al. 2010; Gu, G. Gao, et al. 2014) exaggerate this kernel selection by employing sparsity constraints or objectives in their optimization problem.

Although multiple-kernel learning algorithms has obtained considerable results in various applications, significant well-studied methods are designed and restricted only to binary-classification problems (S.-J. Kim, Magnani, and Boyd 2006; Aiolli and Donini 2015; H. Xue, Yu Song, and H.-M. Xu 2017; Z. Xu, Jin, H. Yang, et al. 2010; Rakotomamonjy et al. 2008; Dileep and Sekhar 2009). It is possible to apply these binary MKL methods to multi-class problems using their ensemble (Gu, G. Gao, et al. 2014; Dileep and Sekhar 2009; Jingjing Yang et al. 2012). However, such strategy results in several kernel combinations for a single given task, that are not interpretable in terms of finding a unanimous set of relevant base kernels.

On the other hand, a different group of multiple-kernel learning methods extends their framework to multi-class problems by focusing on separating data classes in the combined RKHS space (J. Ye, S. Ji, and J. Chen 2008; Y.-Y. Lin, T.-L. Liu, and Fuh 2011; W. Jiang and Chung 2014; Gu, Qingwang Wang, et al. 2015; Qingwang Wang, Gu, and Tuia 2016; Gu, C. Wang, et al. 2012). However, their mathematical formulations either aim for a linear separation of classes in the features space or try to make each data class's distribution globally condensed. Nevertheless, as a common observation in real data, some classes consist of sub-clusters located on different regions of the feature space (e.g., having an XOR distribution in the feature space). For such problems, the principal assumption of the above multiple-kernel learning frameworks about obtaining that ideal target RKHS is not plausible. These shortcomings are fundamentally problematic for classifiers that rely on linear separation of classes in the feature space, such as kernel-based SVM (K-SVM) (Cristianini, Shawe-Taylor, et al. 2000).

As illustrated in Chapter 3, motion sequences can be compared to each other based on the semantic similarity of their pairwise joints' movement. Such component-wise alignment of motion data can result in a multiple-kernel representation, in which each base kernel is associated with one individual body joint. Furthermore, by applying a post-processing regularization step on my DTW-LMNN , I showed that by even using a small set of significant body joints (features), a classifier such as DTW-LMNN can reach its optimal performance. This observation is due to the considerable redundancy that exists among the movement of different body joints. As another point, eliminating some motion dimensions from the preprocessing steps, such as computing DTW distance, can lead to a notable reduction in computation time when the application of alignment techniques, such as DTW, is required. Therefore, it is of great interest to investigate a multiple-kernel-based feature selection for motion data that signifies relevant features, especially for practitioners.

As we observed in Chapter 3, the application of metric learning algorithms such as LMNN (Kilian Q. Weinberger and Lawrence K. Saul 2009) on motion data improves the classes' local separation in small neighborhoods in the space. The enhancement of the data distribution on such space, which is spanned by component-wise distance vectors, is specifically beneficial to neighborhood-based classifiers such as *k*NN (Goldberger et al. 2005; Shalev-Shwartz, Singer, and A. Y. Ng 2004). Given such motivation, the follow-up to the research question **RQ3** is:

RQ3-a: Can we use the multiple-kernel representation of motion sequences in a metric learning framework such as LMNN to perform an efficient feature selection for mocap data?

From a different perspective, domain specialists and practitioners are notably in favor of prototype-based (PB) models in the area of machine learning and knowledge representation. Cognitive psychology claims that human categorizes different data classes in his mind by finding their most representative prototypes (examples) (Rosch 1975). A supervised prototype learning algorithm constructs representatives in the input space, and predicts the class label based on their distances to the given data point (Friedman, Hastie, and Tibshirani 2001). In a mocap database, such representatives could be exemplar sequences which are selected or constructed according to their representative or discriminative quality. Apart from the straightforward interpretation of PB models, their decisions are highly explainable (e.g., for a practitioner) by the direct inspection of the prototypes to which each test data is assigned (Hammer, Hofmann, et al. 2014). Accordingly, several kernel-based algorithms exist, which makes PB models applicable to structured data such as motions sequences. In particular, kernel K-means (Shawe-Taylor and Cristianini 2004; S. Wang, Gittens, and Mahoney 2019) and kernelized LVQ variants (Hofmann et al. 2014; Coelho and Barreto 2019) represent the well-known unsupervised and supervised prototype learning algorithms, respectively. By considering the earlier discussion about multiple-kernel representation of data, an interesting question is if we can benefit from such representation in PB models. In such a framework, the base kernels are combined with a weighting scheme that results in more efficient prototypes regarding representation and discrimination of data samples.

Generally speaking, there has been no significant multiple-kernel prototype learning framework proposed yet to combine various pre-computed base kernels effectively. Nevertheless, in a group of methods similar to (J. Wang J. Yang, Bensmail, and X. Gao 2014; X. Zhu et al. 2017; Gan et al. 2018), the multiple-kernel learning framework is joined with sparse coding. Such a combination aims to improve the learned dictionary's reconstruction and discrimination quality by optimizing it on an efficiently combined RKHS. From a different perspective, one can consider the dictionary atoms as a set of representative prototypes, representing input data through a sparse encoding and possibly revealing their supervised properties. Despite the discriminative performance of these methods and their intuitive structure, another significant concern is to learn interpretable prototypes, which represent condensed data neighborhoods without any inter-class overlap (Friedman, Hastie, and Tibshirani 2001). Usually, this concern induces a trade-off between the discriminative and interpretative quality of the prototypes (Bien, Tibshirani, et al. 2011), and more often, the model sacrifices one of them in favor of the other. Hence, the learned dictionary atoms of multiple-kernel sparse coding methods either suffer from the weak interpretation or cannot discriminatively represent data classes. Accordingly, a follow-up research question for **RQ3** is raised:

RQ3-b: How can we reformulate a prototype learning problem as a multiple-kernel sparse coding framework, which results in interpretable and discriminative motion prototypes to represent other sequences.

Observing new motion categories in the recall phase of a motion recognition task is a common real-world challenge, especially when the motion is captured from daily human movements in a public environment. Furthermore, there is a considerable diversity in human activity categories, which makes it difficult to learn/define all the possible classes (D. Lu, J. Guo, and X. Zhou 2016).

In areas of machine learning, such a problem is generally formulated as zero-shot learning (ZSL), which is the problem of recognizing novel categories of data when no prior information is available during the training phase (Alabdulmohsin, Cisse, and Xiangliang Zhang 2016; Lampert, Nickisch, and Harmeling 2009; Socher et al. 2013; Wei Wang et al. 2019). One practical approach to such transfer learning is the incorporation of semantic attributes as descriptive features to map the input data to an intermediate space, in which different unseen categories can be separated into distinct clusters (Lampert, Nickisch, and Harmeling 2009; Socher et al. 2013; Y. Long et al. 2018).

Due to observing considerable number of new classes in multivariate time-series (MTS) such as audio signals and human motions, several ZSL methods focused on MTS problems (H.-T. Cheng et al. 2013; D. Lu, J. Guo, and X. Zhou 2016; Al-Naser et al. 2018; Choi et al. 2019). Different from images and video, MTS do not possess any general spatial dependency between its dimensions. Therefore a majority of ZSL algorithms are not applicable to such structured data. Nevertheless, the ZSL works on MTS data usually try to find semantic attributes shared between different time-series classes. Despite the achievements in learning unseen MTS data, either the existing methods depend on having prior information about the novel classes (e.g., samples/labels) (D. Lu, J. Guo, and X. Zhou 2016), or their representation for unseen data is not interpretable in terms of their learned attributes.

As discussed in Chapter 4, sparse coding frameworks can capture the intrinsic characteristics of a given dataset. These characteristics can be considered semantic attributes that are encoded by the sparse code or the learned dictionary. Accordingly, some ZSL works have benefited from sparse coding methods in designing more effective attributes for dealing with unseen data classes (Qiu, Z. Jiang, and Chellappa 2011; Ziming Zhang and Saligrama 2015; Kolouri et al. 2017). However, these efforts are mainly limited to the image (spatial) and video (spatiotemporal) datasets. Considering the potentials of K-SRC models in providing sparse encoding of motion data (Chapter 4), it is highly expected to benefit from such models for ZSL of motion sequences or other structured data types.

As a relevant observation in motion data, specifically human motions, it is highly expected to find similarity between specific joint movements of different motion sequences (Hosseini and Hammer 2019d). As a familiar daily life example, people perform many actions with their upper body joints while their lower body joints are engaged in the *walking* movement. We can mention some typical actions, such as *reading*, *waving*, *calling*, and *drinking*, being performed while *walking*. Such partial similarity between different motion classes can be used as semantic attributes, which also provides an interpretable encoding of an unseen motion sequence without any prior knowledge about its class label. This partial encoding can be used to categorize unseen motion sequences into their distinct underlying subspaces, and gives us some meaningful information about each encoded motion. Moreover, another related concern in this area of research is the partial or complete encoding of unseen motion classes based on their relation to some learned attributes or motions from the training data (P. Peng et al. 2018; Qiu, Z. Jiang, and Chellappa 2011). With that perspective, one can achieve an interpretable representation of an unseen motion based on its piece-wise relations to other known motion categories. Analyzing and comparing mocap data based on their partial components in the above paradigms raise another follow-up research question for **RQ3**:

RQ3-c: How can we employ the multiple-kernel representation of motion sequences in a sparse coding framework to obtain descriptive semantic attributes for the interpretable encoding of unseen motion sequences?

Accordingly, the following follow-up research question for **RQ3** is raised:

In this chapter, I propose different supervised and unsupervised multiple-kernel frameworks to To address the above research questions. These approahces are summarized in Figure 5.1. These algorithms improve the analysis of motion data given its dimension-based kernel transfers to RKHSs are available. Each of these algorithms has individual goals, which explain the need for their specific formulations. As a result, their outcome regarding the specific intake of motion dimensions is different and in favor of their specifically defined purpose. In summary, I have the following contributions with respect to the relevant state-of-the-artalgorithms.

- I propose a large-margin multiple-kernel algorithm (LMMK), which sparsely combines the given base kernels (derived from motion dimensions) to improve the classes' local separation in a resulting RKHS. LMMK learns a scaling of the feature space which signifies the relevant motion dimensions to the *k*NN classifier.
- I extend the application of prototype-based learning to multiple-kernel data representation. My proposed interpretable multiple-kernel prototype learning (IMKPL) algorithm transfers data to a combined RKHS, in which the learned prototypes

are interpretable by class-specific local neighborhoods. IMKPL optimization particularly shapes prototypes to be representative and discriminative regarding their neighboring points in RKHS. RKHS.

- I design a novel multiple-kernel dictionary structure that its atoms are constructed by specific combinations of the computed base kernels (corresponding to motion dimension). These dictionary atoms are used as semantic attributes, upon which unseen classes of motions can be recognized and categorized in an unsupervised way. My MKD algorithm also provides a partial reconstruction of unseen motions in the feature space with an interpretable encoding.

In the next section, I provide the necessary background for multiple-kernel learning

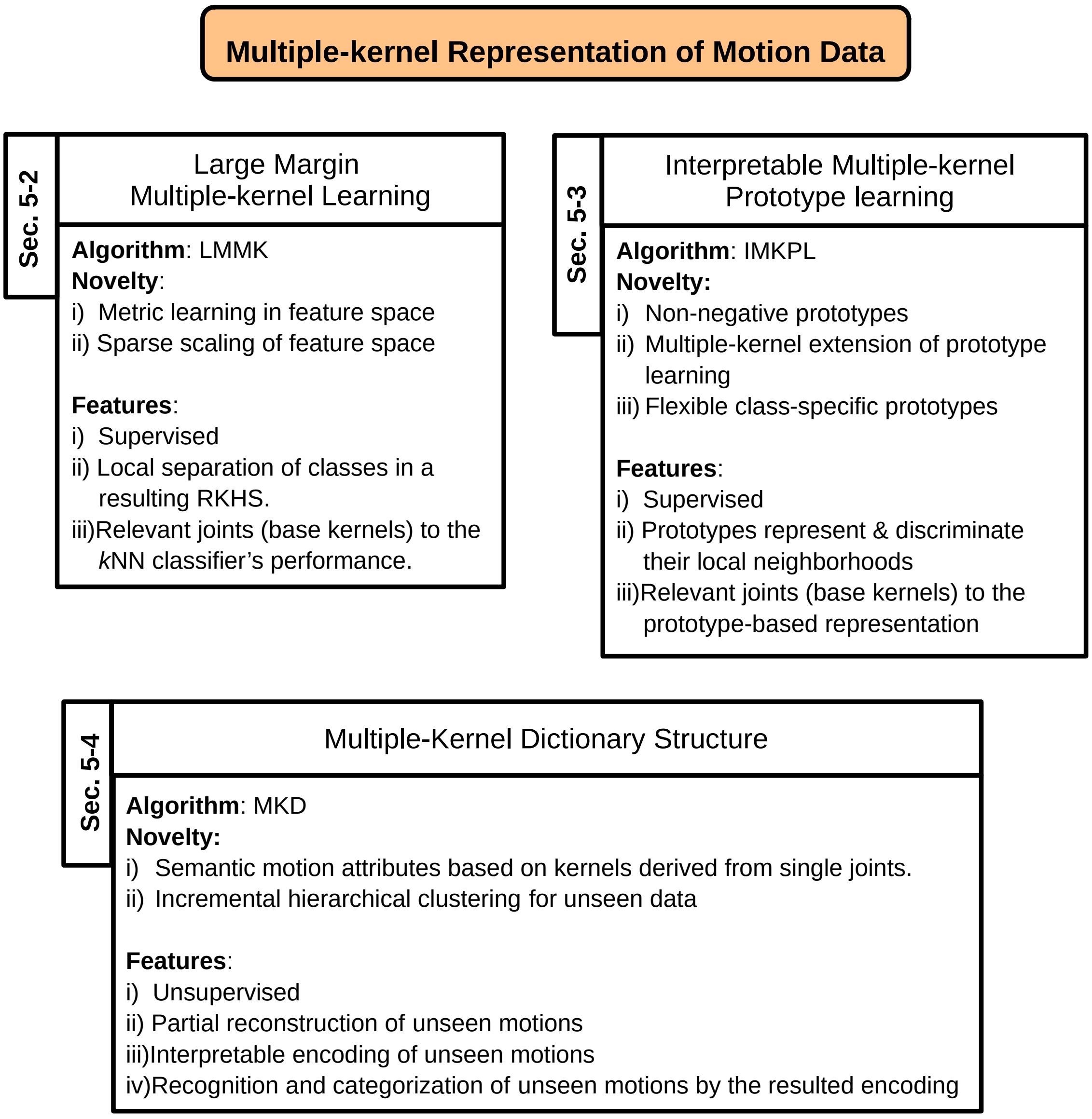

Multiple-kernel Representation of Motion Data
Sec. 5-2
Large Margin Multiple-kernel Learning
Algorithm: LMMK
Novelty:
i) Metric learning in feature space
ii) Sparse scaling of feature space
Features:
i) Supervised
ii) Local separation of classes in a resulting RKHS.
iii)Relevant joints (base kernels) to the *k*NN classifier's performance.
Sec. 5-3
Interpretable Multiple-kernel Prototype learning
Algorithm: IMKPL
Novelty:
i) Non-negative prototypes
ii) Multiple-kernel extension of prototype learning
iii) Flexible class-specific prototypes
Features:
i) Supervised
ii) Prototypes represent & discriminate their local neighborhoods
iii)Relevant joints (base kernels) to the prototype-based representation
Sec. 5-4
Multiple-Kernel Dictionary Structure
Algorithm: MKD
Novelty:
i) Semantic motion attributes based on kernels derived from single joints.
ii) Incremental hierarchical clustering for unseen data
Features:
i) Unsupervised
ii) Partial reconstruction of unseen motions
iii)Interpretable encoding of unseen motions
iv)Recognition and categorization of unseen motions by the resulted encoding

Figure 5.1: Summary of different proposed algorithms in Chapter 5 for interpretable representation of (motion) data based on multiple-kernel information. The methods are specifically distinguished according to the supervised / unsupervised, discriminative / representative, or the sparsity characteristics in their formulations.

and related multiple-kernel state-of-the-art. Then, the proposed large-margin multiple-kernel algorithm, interpretable multiple-kernel prototype learning framework, and multiple-kernel dictionary structure are introduced in the sequence of individual sections. Afterward, all proposed algorithms are empirically evaluated in the experiments Section followed by the chapter's conclusion.

5.1 STATE OF THE ART

In this chapter, I review the preliminaries for multiple-kernel learning and multiple-kernel dictionary learning. I also discuss the important works in those areas upon which I design my proposed LMMK , IMKPL , and MKD frameworks.

Multiple Kernel Learning

Similar to previous chapters, I consider a mocap training set as $\mathcal{X} = \{\mathbf{X}_i\}_{i=1}^N$ containing N motion sequences $\mathbf{X}_i \in (\mathbb{R}^d)^*$, which are d-dimensional time-series of different lengths. Hence, we can implicitly assume d non-linear mapping functions

$$\{\Phi_m : (\mathbb{R}^d)^* \to \mathbb{R}^{d_m}\}_{m=1}^d \tag{5.1}$$

exist which map $\mathcal{X}$ into d individual RKHSs (F. R. Bach, G. R. Lanckriet, and Jordan 2004; J. Wang J. Yang, Bensmail, and X. Gao 2014). Therefore, we can obtain a scaling of the feature space based on the following weighted concatenation:

$$\hat{\Phi}(\mathbf{X}) = [\sqrt{\beta_1}\Phi_1^\top(\mathbf{X}), \ldots, \sqrt{\beta_d}\Phi_d^\top(\mathbf{X})]^\top, \tag{5.2}$$

where vector $\hat{\Phi}(\mathbf{X})$ is the implicit mapping to the resulting RKHS, and $\vec{\beta}$ is the combination vector. Due to the finiteness of training samples $\mathbf{X}_i$, it can be assumed that the target of each implicit mapping Φ_m is a finite-dimensional Hilbert space which validates the concatenation of its corresponding embedding in Equation 5.2. By relating each $\Phi_m(\mathbf{X})$ to a kernel function $\mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_j) = \Phi_m^\top(\mathbf{X}_i)\Phi_m(\mathbf{X}_j)$, we can compute the weighted kernel function $\hat{\mathcal{K}}(\mathbf{X}_i, \mathbf{X}_j)$ corresponding to $\hat{\Phi}(\mathbf{X})$ as the additive combination (Dileep and Sekhar 2009)

$$\hat{\mathcal{K}}(\mathbf{X}_i, \mathbf{X}_j) = \sum_{m=1}^{d} \beta_m \mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_j) = \hat{\Phi}(\mathbf{X}_i)^\top \hat{\Phi}(\mathbf{X}_j). \tag{5.3}$$

Generally, one can formulate the MKL frameworks as variants of the following optimization problem:

$$\vec{\beta} = \underset{\vec{\beta}\in\mathcal{S}}{\arg\min} \quad loss(\{\mathcal{K}_m(\mathcal{X}, \mathcal{X})\}_{m=1}^d, \vec{\beta}, \vec{h}), \tag{5.4}$$

where $\mathcal{K}_m(\mathcal{X}, \mathcal{X})$ is the m-th kernel matrix for the training data $\mathcal{X}$. In Equation 5.4, the *loss* term is a cost function that its minimization reflects the given classification task and is also defined by considering the classifier's model. The set $\mathcal{S}$ defines the set of employed constraints on $\vec{\beta}$ based on the MKL algorithm.

If we apply each kernel function $\mathcal{K}_m$ only on the m-th dimension of the training data (resulting in d feature-kernels), we can assume each corresponding Φ_m in Equation 5.2 maps the m-th dimension of the data into one individual RKHS. In that case, each solution for Equation 5.4 represents a weighted feature selection obtained by the MKL algorithm based on the defined discriminative function *loss* and the constraints in $\mathcal{S}$. It is practical to apply a non-negativity constraint on each β_m to make the resulting kernel weights interpretable as the relative importance of each feature representation to the given discriminative task (Gönen and Alpaydın 2011).

Furthermore, by deriving each base kernel from a different source of information in the data, it is highly possible to observe substantial redundancy between these representations(P. Du et al. 2017). Therefore, it is desirable to reduce this redundancy in favor of the model's interpretation and its discrimination power. In the works similar to SimpleMKL (Rakotomamonjy et al. 2008) and class-specific MKL (Tianzhu Liu et al. 2016), they imposed sparsity on the weights of the base kernels by using a convex combination in the MKL problem. As an improvement, Group Lasso-MKL fused the MKL problem with the l_p-norm based on the group Lasso optimization (Tibshirani 1996) to enforce better the sparsity concern (Z. Xu, Jin, H. Yang, et al. 2010). In comparison, SparseRMKL (Gu, G. Gao, et al. 2014) benefits from an l_1-norm constraint in its optimization framework, which provides a better classification performance as well as an enhanced interpretation by specifying the most discriminative contributions among the set of the base kernels.

As a common characteristic among multi-class multiple-kernel algorithms, they try to learn the optimal kernel weights independently of the later on classifier's structure. Inspired by the Fisher Linear Discriminant Analysis (LDA) (Duda and Hart 1973), algorithms similar to DKL (J. Ye, S. Ji, and J. Chen 2008), MKL-DR (Y.-Y. Lin, T.-L. Liu, and Fuh 2011) and MKL-TR (W. Jiang and Chung 2014) are focused on reducing the intra-class covariances via using the scatter matrices of data in different RKHSs. In particular, the MKL-DR and MKL-TR methods employ low-dimensional projections, while the latter also applies the convex combination of the base kernels. As a different approach, the RMKL method (Gu, C. Wang, et al. 2012) performs singular value decomposition to find the base kernels, leading to maximum variation in the space spanned by them. It is claimed that this decomposition finds a more discriminative kernel combination than the original RKHS. Similarly, KNMF-MKL (Gu, Qingwang Wang, et al. 2015) was proposed by reformulating the RMKL approach using the non-negative matrix factorization framework (NMF) (D. D. Lee and Seung 2001).

To emphasize the considerable shortcomings of the existing MKL algorithms, I distinguish them into two general categories: First group of algorithms similar to (Dileep and Sekhar 2009; H. Xue, Yu Song, and H.-M. Xu 2017; Rakotomamonjy et al. 2008; Aiolli and Donini 2014; Aiolli and Donini 2015) focus on learning a multiple-kernel mapping to a target RKHS in which a classifier can linearly separate the different classes from each other. This objective coincides with the basic principle of the kernelized SVM's structure (Cristianini, Shawe-Taylor, et al. 2000), which is the linear separation of the classes in the feature space. Nevertheless, obtaining such an ideal representation is usually not affordable for real-world data, or it demands considerable domain knowledge for the specific design of such efficient kernels. This category generally includes binary MKL algorithms.

The other group of MKL methods includes algorithms such as (S.-J. Kim, Magnani, and Boyd 2006; J. Ye, S. Ji, and J. Chen 2008; Y.-Y. Lin, T.-L. Liu, and Fuh 2011; Qingwang Wang, Gu, and Tuia 2016) which follow methodologies analogous to the kernelized LDA's design scheme (Mika, Rätsch, and K.-R. Müller 2001). They focus on obtaining a condensed representation of data classes in the resulting RKHS, which is beneficial to multi-class problems. However, they don't perform well on real-data when data classes have distinct sub-clusters in the feature space. In such a case, a globally condense representation is difficult to achieve even with a multiple-kernel scheme, especially without doing any feature engineering (I. W. Tsang, Kocsor, and Kwok 2006).

In contrast, I propose a multiple-kernel learning framework in Section 5.2, which focuses on obtaining a local separation of motion classes in a combined RKHS. By using neighborhood-based decision-making, such a framework can mitigate the above limitation in the current multiple-kernel learning methods.

The goal of multiple-kernel dictionary learning (MKDL) is to find an optimal MK dictionary $\hat{\Phi}(\mathbf{D})$ on the combined RKHS to reconstruct the inputs as $\hat{\Phi}(\mathcal{X}) \approx \hat{\Phi}(\mathbf{D})\mathbf{\Gamma}$ in this space. A basic MKDL framework can be formulated as a variant of the following

$$\begin{aligned} \min_{\mathbf{\Gamma},\mathbf{U}} \quad & \|\hat{\Phi}(\mathcal{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 \\ \text{s.t.} \quad & \|\vec{\beta}\|_1 = 1,\ \beta_i \in \mathbb{R}_{\geq 0},\ \|\vec{\gamma}_i\|_0 \leq T, \end{aligned} \tag{5.5}$$

where the objective term $\mathcal{J}_{rec} = \|\hat{\Phi}(\mathcal{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$ measures the reconstruction quality of the data on the resulting RKHS. Similar to the dictionary model of Equation 4.6, the dictionary in Equation 5.5 is modeled as $\hat{\Phi}(\mathbf{D}) = \hat{\Phi}(\mathcal{X})\mathbf{U}$, where each column of $\mathbf{U}$ defines a linear combination of data points in the resulting combined RKHS. The constraint $\|\vec{\beta}\|_1 = 1$ applies an affine combination of the base kernels and also prevents the trivial solution $\vec{\beta} = 0$. The role of $\vec{\beta}$ in $\hat{\Phi}(\mathcal{X})$ is to enhance the discriminative power of the learned dictionary atoms $\{\hat{\Phi}(\mathcal{X})\vec{u}_i\}_{i=1}^k$ by increasing the dissimilarity between the different-label columns in $\hat{\Phi}(\mathcal{X})$.

Although $\mathcal{J}_{rec}$ is a common term in MKDL methods, it varies based on the multiple-kernel or dictionary-learning part's formulation. In (Thiagarajan, Ramamurthy, and Spanias 2014), the vector $\vec{\beta}$ was individually optimized to improve the linear separability of the classes on the RKHS. In contrast, (Shrivastava, Pillai, and Patel 2015) jointly optimized $\{\mathbf{U}, \vec{\beta}\}$ by pre-defining class-isolated sub-dictionaries in $\mathbf{U}$ and enforcing the orthogonality of each class to the dictionaries of other classes on the RKHS; and (X. Zhu et al. 2017) utilized an analysis-synthesis class-isolated dictionary model along with a low-rank constraint on $\mathbf{\Gamma}$.

Compared to these frameworks, my proposed multiple-kernel learning algorithm in Section 5.3 explicitly shapes the dictionary atoms as interpretable prototypes, which improve local representation and discrimination of the classes effectively. However, none of the major MKDL methods adequately provide such a PB model.

In this next section, I explain our proposed multiple-kernel frameworks with respect to their specific objectives, their formulations, and the individual optimization algorithms I use or proposed to solve them.

5.2 LARGE-MARGIN MULTIPLE KERNEL LEARNING FOR DISCRIMINATIVE FEATURE SELECTION

By focusing on discriminative tasks, multiple-kernel learning has been used successfully for feature selection and finding the data's significant modalities. In such applications, each base kernel represents one dimension of the data or is derived from one specific descriptor (e.g., in image processing). Therefore, multiple-kernel learning finds an optimal weighting scheme for the given kernels to increase classification accuracy. Nevertheless, the majority of the works in this area focus on only binary classification problems or aim for linear separation of the classes in the kernel space, which are not realistic assumptions for many real-world problems. In this section, I propose a novel multi-class multiple-kernel learning framework that improves state-of-the-art by enhancing the local separation of the classes in the feature space. Besides, by using a sparsity term, my large-margin multiple-kernel algorithm (LMMK) performs discriminative feature selection by aiming to employ a small subset of the base kernels. For motion datasets, the base kernels coincide with the different dimensions related to body joints. Therefore, the application of the LMMK algorithm on motion data results in a discriminative feature selection that determines a set of relevant motion dimensions to the given classification task.

I apply the metric learning concept to the data distribution in the feature space, such that it results in having dense neighborhoods of classes in which the different classes can be locally separated. Assuming that the dimensions of the feature space are related to individual RKHSs as in Equation 5.2, I employ metric learning to find the effective $\vec{\beta}$ that serves the above purpose. However, direct application of Equation 3.3 in the feature space has the following limitations:

First, via applying the Mahalanobis metric of Equation 3.1 to the feature space, the dimensions of the resulting $\tilde{\Phi}(\mathbf{X})$ lose their interpretability. Denoting $\Phi(\mathbf{X})$ as the non-weighted concatenation of the base kernels in Equation 5.2 (setting $\beta_m = 1\ \forall m$),

$$\tilde{\Phi}(\mathbf{X})(i) = \sum_j l_{ij}\Phi(\mathbf{X})(j) \quad \text{having } \tilde{\Phi}(\mathbf{X}) = \mathbf{L}\Phi(\mathbf{X}), \tag{5.6}$$

in which $\Phi(\mathbf{X})(i)$ and $\tilde{\Phi}(\mathbf{X})(i)$ denote the i-th entry of the vectors $\Phi(\mathbf{X})$ and $\tilde{\Phi}(\mathbf{X})$, respectively in the feature space. Consequently, each dimension of $\tilde{\Phi}(\mathbf{X})$ in the resulting RKHS loses its physical interpretation, as it is a weighted combination of the dimensions of the original RKHS.

Second, computing Equation 3.1 in the feature space (as in Equation 5.6) requires direct access to the dimensions of each $\Phi_m(\mathbf{X})$ in the feature space. This requirement cannot be directly fulfilled because it contradicts our assumption about the implicit definition of $\Phi_m(\mathbf{X})$.

To overcome the above issues, I propose the following optimization scheme with the same notations as used in Equation 3.3:

$$\begin{aligned}
\min_{\vec{\beta}} \quad & (1-\mu) \sum_{i,j\in\mathcal{N}_i^k} \mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i,\mathbf{X}_j) \\
& +\mu \sum_{i,j\in\mathcal{N}_i^k} \sum_{l\in\mathcal{I}_i^k} \xi_{ijl} + \lambda \textstyle\sum_m \beta_m \\
\text{s.t.} \quad & \mathcal{D}^{\phi}_{\vec{\beta}}(\vec{x}_i,\vec{x}_l) - \mathcal{D}^{\phi}_{\vec{\beta}}(\vec{x}_i,\vec{x}_j) \geq 1 - \xi_{ijl} \\
& \xi_{ijl} \geq 0, \quad \beta_m \geq 0.
\end{aligned} \tag{5.7}$$

In Equation 5.7, the distance metric $\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i,\mathbf{X}_j)$ is defined in the feature space as:

$$\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i,\mathbf{X}_j) = [\Phi(\mathbf{X}_i) - \Phi(\mathbf{X}_j)]^\top \mathbf{B} [\Phi(\mathbf{X}_i) - \Phi(\mathbf{X}_j)], \tag{5.8}$$

where $\mathbf{B}$ is a diagonal matrix formed based on the entries of $\vec{\beta}$. Equation 5.8 defines a Mahalanobis metric in the feature space with a diagonal covariance matrix $\mathbf{B}$. Therefore, I name $\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i,\mathbf{X}_j)$ a diagonal metric. Consequently, each learned β_m in Equation 5.7 acts as a selection weight for the m-th representation of the data in the original RKHS to locally discriminate the classes in the feature space (similar to Figure 3.1). Additionally, the last objective term in this optimization problem applies an l_1-regularization to enforce the selection of the most relevant feature-kernels $\Phi_m(\mathbf{X})$ to the defined discriminative objective. Therefore, my LMMK framework in Equation 5.7 is an MKL optimization problem designed for discriminative feature selection and representation learning.

Optimization

Based on Equation 5.3, the pairwise distance between each couple of $(\mathbf{X}_i, \mathbf{X}_j)$ in the feature space is computed as

$$\begin{aligned}&\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i, \mathbf{X}_j) = \\ &\textstyle\sum_{m=1}^{d} \beta_m [\mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_i) + \mathcal{K}_m(\mathbf{X}_j, \mathbf{X}_j) - 2\mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_j)].\end{aligned} \tag{5.9}$$

Hence, I can compute $\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{X}_i, \mathbf{X}_j)$ without performing any explicit calculation in the feature space in contrast to Equation 3.1. In addition, we have $\mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_i) = 1$ for all the input vectors and base kernels by normalizing the kernel matrices of the training set. Therefore, after eliminating the constant terms, the optimization problem of Equation 5.7 is simplified to

$$\begin{aligned}\min_{\vec{\beta}} \quad & (1-\mu)\Big(\sum_{i,j \in \mathcal{N}_i^k} [1 - \mathcal{K}_{(:)}(\mathbf{X}_i, \mathbf{X}_j)]\Big)\vec{\beta} \\ & +\mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l \in \mathcal{I}_i^k} \xi_{ijl} + \lambda \textstyle\sum_{m=1}^{d} \beta_m \\ \text{s.t.} \quad & 2[1 + \mathcal{K}_{(:)}(\mathbf{X}_i, \mathbf{X}_j) - \mathcal{K}_{(:)}(\mathbf{X}_i, \mathbf{X}_l)]\vec{\beta} \geq 1 - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \quad \beta_m \geq 0,\end{aligned} \tag{5.10}$$

where $\mathcal{K}_{(:)}(\mathbf{X}_i, \mathbf{X}_j) := [\mathcal{K}_1(\mathbf{X}_i, \mathbf{X}_j), \ldots, \mathcal{K}_d(\mathbf{X}_i, \mathbf{X}_j)] \in \mathbb{R}^d$. This optimization framework is a convex problem subject to the advance selection of the *targets* and *impostors* indexed by $\mathcal{N}_i^k$ and $\mathcal{I}_i^k$, respectively. Hence, it is an instance of the non-negative linear programming (LP), and we can efficiently optimize it via using solvers such as YALMIP (Lofberg n.d.) or CVX (Grant, Boyd, and Y. Ye 2008). Additionally, similar to a practical hint from (Kilian Q. Weinberger and Lawrence K. Saul 2009), I repeat the optimization loop for a few iterations while updating $\mathcal{N}_i^k$ and $\mathcal{I}_i^k$ at the end of each run. These few extra repetitions can lead to more optimal solutions.

Classification of Test Data

I perform the classification of each test motion sequence $\mathbf{Z}$ by using the kNN algorithm based on the distances in the resulting RKHS. To that aim, I compute $\mathcal{D}^{\phi}_{\vec{\beta}}(\mathbf{Z}, \mathbf{X}_i)$ as the distance between $\mathbf{Z}$ and each training sample using the learned diagonal matrix $\mathbf{B}$ in the feature space analogous to Equation 5.9.

Complexity and Convergence of LMMK

The optimization framework of Equation 5.10 is an LP problem, and consequently, it converges in limited t steps to an optimal solution. On the other hand, an LP solver optimizes $\vec{\beta}$ with the computational complexity of $\mathcal{O}(t(2d + 3N_l) + dN_j + 2dN_l)$, in which N_l and N_j are the total number of *targets* and the size of $\vec{\xi}$, respectively. Based on the definition of the *targets* and *impostors*, we have $N_l \approx \frac{N^2(C-1)}{C}$ and $N_j = kN$. Also, for common real-world datasets, we observe $N >> t$ in practice; hence, the total time complexity of the algorithm is approximately $\mathcal{O}(N^2)$. This complexity is almost comparable to computing the base kernel matrices for each dataset before running the algorithm.

Comparison to DTW-LMNN with Metric Regularization

As explained in Chapter 3, the proposed DTW-LMNN algorithm is applied to the vector of distances $\vec{D}^{ij}$, as pairwise distances between the components of each two given motion sequences in the dataset. Also, after learning the metric transform $\mathbf{L}$, the regularization method of Section 3.4 finds a small set of relevant dimensions in the mocap data. Such process in to some extent analogous to what I proposed in this section as the LMMK algorithm. Nevertheless, these two methods can be distinguished from the following aspects:

1. The DTW-LMNN does not have any feature selection objective while learning its metric coefficients, and it is more of a feature transformation algorithm than feature selection. On the other hand, the LMMK method is specifically formulated such that feature selection is one of its primary objectives. Therefore, even though we can select a small set of features from DTW-LMNN by regularizing its transform matrix $\mathbf{L}$, it is expected to achieve a sparser set of discriminative features from LMMK than DTW-LMNN .

2. The feature selection part in Chapter3 is distinct from the metric learning part. So, the main goal of DTW-LMNN is to learn a metric for better classification of the data. In contrast, the model obtained from LMMK is directly influenced by the sparseness target of $\vec{\beta}$, i.e., the feature selection scheme. Therefore, LMMK is used when feature selection has a significant role in the overarching task.

3. Although DTW-LMNN benefits from a complete metric transform (a full coefficient matrix $\mathbf{L}$) compared to LMMK (a sparse vector of coefficients $\vec{\beta}$), the discriminative performance of LMMK can be still comparable to of DTW-LMNN depending on the data distribution and the used kernel. For example, a Gaussian kernel naturally concentrates the data neighborhoods due to its radial basis function, which can improve the local separation of classes in RKHS compared to that in the distance space in DTW-LMNN . Additionally, the sparseness of the resulting diagonal metric in LMMK can better affect the accuracy compared to the full metric of DTW-LMNN depending on the amount and effect of the redundant information in the motion dimensions. Therefore, these methods have slightly different views on the discriminative problem, leading them to individual discriminative solutions.

The above aspects are empirically validated and discussed in the experiments of Section 5.5 related to the LMMK algorithm. In the next section, I propose my multiple-kernel dictionary learning method, which has a prototype-based view of the multiple-kernel problem while benefiting from the basic formulation of sparse coding frameworks.

That method is useful for learning discriminative and representative prototypes for motion (structured) data given a multiple-kernel representation of data is available,

5.3 interpretable multiple-kernel prototype learning

From a different perspective, prototype-based methods are of particular interest for domain specialists and practitioners because these models summarize a dataset by a small set of representatives. Explicitly talking about motion data, prototype-based learning can result in a set of motion prototypes base on which other motion samples can be represented and classified. Therefore, in a classification setting, the prototypes' interpretability is as significant as the prediction accuracy of the algorithm. Nevertheless, the state-of-the-art methods make an inefficient balance between these concerns by sacrificing one in favor of the other, especially if the given data has a kernel-based (or multiple-kernel) representation. In this section, I propose a novel interpretable multiple-kernel prototype learning (IMKPL), which benefits from the multiple-kernel representation of motion data to construct highly interpretable prototypes in the feature space. These prototypes are effective for the discriminative representation of the data. My method focuses on the local discrimination of the classes in the feature space and shapes the prototypes based on condensed class-homogeneous data neighborhoods. Besides, IMKPL learns a combined embedding in the feature space in which the above objectives are better fulfilled.

I want to learn an MK dictionary that its constituent prototypes (atoms) reconstruct the data while presenting discriminative characteristics interpretable in terms of the class labels. To be more specific, I aim for the following specific objectives:
Ob1: Assigning prototypes to the local neighborhoods in the classes to efficiently discriminate them on the RKHS regarding their class labels (Figure 5.3-d).
Ob2: Learning prototypes which can be interpreted by the condensed class-specific neighborhoods they represent (Figure 5.2-b)
Ob3: Obtaining an efficient MK representation of the data to assist the above objectives and improve the local separation of the classes in the resulting RKHS (Figure 5.3).

Definition 5.1. Each $\mathbf{X}$ is represented by a set of prototypes $\{\hat{\Phi}(\mathcal{X})\vec{u}_i\}_{i\in I}$ on the combined RKHS if $\|\hat{\Phi}(\mathbf{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\vec{\gamma}\|_2^2 < \epsilon$ for a small $\epsilon > 0$ and $\forall i \in I,\ \gamma_i \neq 0$.

Based on Definition 5.1, I call $\{\vec{u}_i\}_{i=1}^k$ the prototype vectors to represent the columns of $\hat{\Phi}(\mathcal{X})$, and I propose the interpretable multiple-kernel prototype learning algorithm to learn them while adequately addressing the above objectives. IMKPL has the novel optimization scheme of:

$$\begin{aligned} \min_{\vec{\beta},\mathbf{\Gamma},\mathbf{U}} \quad & \|\hat{\Phi}(\mathcal{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 + \lambda\mathcal{J}_{dis} + \mu\mathcal{J}_{ls} + \tau\mathcal{J}_{ip} \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T,\ \ \|\vec{\beta}\|_1 = 1,\ \ \|\hat{\Phi}(\mathcal{X})\vec{u}_i\|_2^2 = 1, \\ & \|\vec{u}_i\|_0 \leq T,\ \ u_{ji}, \beta_i, \gamma_{ji} \in \mathbb{R}_{\geq 0}, \end{aligned} \tag{5.11}$$

in which λ, τ, and μ are trade-off weights. The cardinality and non-negativity constraints on $\{\mathbf{U}, \mathbf{\Gamma}\}$ coincide with the dictionary structure $\hat{\Phi}(\mathcal{X})\mathbf{U}$ as discussed in Section 4.3. They motivate each prototype $\hat{\Phi}(\mathcal{X})\vec{u}_i$ to be formed by sparse contributions from similar training samples in $\hat{\Phi}(\mathcal{X})$ to increase their interpretability (Bien, Tibshirani, et al. 2011). Although each $\vec{u}_i$ is loosely shaped from the local neighborhoods in the RKHS, it cannot fulfill the objectives **Ob1** and **Ob2** on its own (Figure 5.2-a). Similar to K-SRC frameworks in Chapter 4, having $\|\hat{\Phi}(\mathcal{X})\vec{u}_i\|_2^2 = 1$ prevents the solution of $\vec{u}_i$ from being degenerated (Rubinstein, Zibulevsky, and Michael Elad 2008).

At first sight, the optimization problem of Equation 5.11 may look similar to the proposed CKSC framework in Chapter 4. In particular, when we neglect the role of $\vec{\beta}$, the reconstruction objective of Equation 5.11 (the 1st term) becomes identical to that of Equation 4.22. Even though $\hat{\Phi}(\mathcal{X})\vec{u}_i$ has the same mathematical formulation in both of the mentioned optimization frameworks, fulfilling the proposed objectives **Ob1** and **Ob2** will add specific characteristics to $\hat{\Phi}(\mathcal{X})\vec{u}_i$, which differentiate it from a typical dictionary atom as in Equation 4.22. Such properties let us treat each resulted $\hat{\Phi}(\mathcal{X})\vec{u}_i$ as a prototype for motion data, which locally represents and discriminates a condense neighborhood of motion sequences, and can be meaningfully (semantically) assigned to one specific motion class. However, we cannot generally assign such characteristics to the learned atoms in the CKSC framework. In the following subsections, I explain the novel terms $\{\mathcal{J}_{dis}, \mathcal{J}_{ls}, \mathcal{J}_{ip}\}$ and how they address the objectives **Ob1**-**Ob3**.

Discriminative Loss $\mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}, \vec{\beta})$

By rewriting $\hat{\Phi}(\mathcal{X})\mathbf{U}\vec{\gamma} = \hat{\Phi}(\mathcal{X})\vec{v}$, the vector $\vec{v} \in \mathbb{R}^N$ reconstructs a vector $\hat{\Phi}(\mathbf{X})$ based on other samples in matrix $\hat{\Phi}(\mathcal{X})$. Hence, by aiming for **Ob1**, we learn the prototype vectors $\{\vec{u}_i\}_{i=1}^k$ such that they represent each $\hat{\Phi}(\mathbf{X})$ with a corresponding vector $\vec{v}$ using mostly the local same-class neighbors of $\hat{\Phi}(\mathbf{X})$. Accordingly, I define the loss term $\mathcal{J}_{dis}$ as:

$$\begin{aligned}&\mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}, \vec{\beta}) = \\ &\tfrac{1}{2}\sum_{i=1}^{N}[\sum_{s=1}^{N} \vec{u}^s\vec{\gamma}_i(\vec{h}_i^\top \vec{h}_s \|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \|\vec{h}_i - \vec{h}_s\|_2^2)].\end{aligned} \tag{5.12}$$

Proposition 5.1. *The objective $\mathcal{J}_{dis}$ in Equation 5.12 has its minimum if $\forall \mathbf{X}_i$, $\hat{\Phi}(\mathbf{X}_i) \approx \hat{\Phi}(\mathcal{X})\mathbf{U}\vec{\gamma}_i$ s.t. $\forall t : \gamma_{ti} \neq 0, \forall s : u_{st} \neq 0, \vec{h}_i = \vec{h}_s$ and $\|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 \approx 0$.*

Proof. Refer to Appendix A.11. □

Although Proposition 5.1 describes the ideal situations, in practice, it is common to observe $\|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 < \epsilon$ for a small, non-negative ϵ when $\mathbf{X}_s$ is among the neighboring points of $\mathbf{X}_i$. This condition results in small non-zero minima for $\mathcal{J}_{dis}$. Besides, for a given $\mathbf{X}_i$, if its cross-class neighbors lie closer to its same-class neighbors, Ω_{si} obtains higher values by choosing $\mathbf{X}_s$ s.t. $\vec{h}_s \neq \vec{h}_i$ in favor of better minimizing $\mathcal{J}_{rec}$ (e.g., the squares in Figure 5.2-b which is a part of $\vec{u}_1$).

Based on Proposition 5.1, minimizing $\mathcal{J}_{dis}$ enforces the framework in Equation 5.11 to learn $\mathbf{U}$ such that each prototype $\hat{\Phi}(\mathcal{X})\vec{u}_i$ is shaped by a concentrated neighborhood in RKHS, providing a discriminative representation for its nearby samples. However, $\mathcal{J}_{dis}$ is still flexible in tolerating small cross-class contributions in the representation of each $\mathbf{X}_i$ in case of having overlapping classes in the data. For example, although a square sample in Figure 5.2-b has contributed to the reconstruction of $\mathbf{X}$ via $\vec{u}_1$ (due to their small distance), $\mathbf{X}$ is still represented mostly by samples of its own class (*circles*).

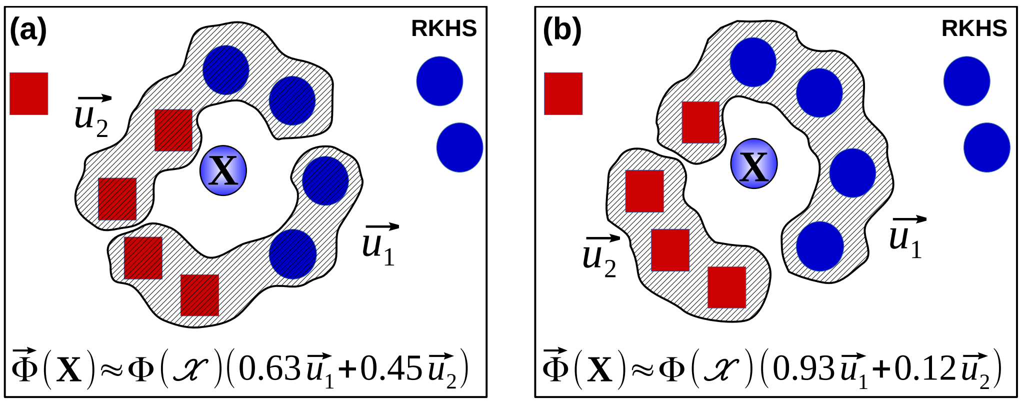

(a)
RKHS
$\vec{u}_2$
X
$\vec{u}_1$
$\vec{\Phi}(\mathbf{X})\approx\Phi(\mathcal{X})(0.63\vec{u}_1+0.45\vec{u}_2)$
(b)
RKHS
$\vec{u}_2$
X
$\vec{u}_1$
$\vec{\Phi}(\mathbf{X})\approx\Phi(\mathcal{X})(0.93\vec{u}_1+0.12\vec{u}_2)$

Figure 5.2: The effect of $\mathcal{J}_{dis}$ in Equation 5.11. (a): When $\lambda = 0$, prototypes $(\vec{u}_1, \vec{u}_2)$ (hatched selections) are shaped and reconstruct $\hat{\Phi}(\mathbf{X})$ by its neighboring samples from both classes (circles and squares). (b): When $\lambda \neq 0$, these prototypes are formed s.t. $\hat{\Phi}(\mathbf{X})$ is approximately represented by $\vec{u}_1$, which is mostly shaped by its local, same-class neighbors (circles).

Interpretability Loss $\mathcal{J}_{ip}(\mathbf{U})$

Definition 5.2. Prototype $\hat{\Phi}(\mathcal{X})\vec{u}_i$ is interpretable as a local representative of the class q if the set $\{\mathbf{X}_t | u_{ti} \neq 0\}$ forms a concentrated neighborhood in the RKHS, and $\frac{\vec{h}^q \vec{u}_i}{\|\mathbf{H}\vec{u}_i\|_1} \approx 1$.

When the class-overlapping is subtle, minimizing $\mathcal{J}_{dis}$ can result in interpretable prototypes (e.g., in Figure 5.2-b, $\vec{u}_1$ can still be interpreted as a local representative for the *circle* class). However, a relatively large overlap of the classes results in having more than one large entries in each $\vec{s} = \mathbf{H}\vec{u}_i$ (similar to $\vec{u}_1$ in Figure 5.2-a). Therefore, to better satisfy objective **Ob2**, I define $\mathcal{J}_{ip}(\mathbf{U}) = \|\mathbf{HU}\|_1$, such that its minimization reduces $\|\vec{s}\|_1$ for each prototype vector. This term (together with) $\mathcal{J}_{dis}$ results in a significantly sparse $\mathbf{H}\vec{u}_i$, such that $\vec{h}^q \vec{u}_i / \|\mathbf{H}\vec{u}_i\|_1$ obtains a value close to 1. Such a situation improves the interpretability of each $\hat{\Phi}(\mathcal{X})\vec{u}_i$ according to Definition 5.2.

Local-Separation Loss $\mathcal{J}_{ls}(\vec{\beta})$

According to Eqs. (5.2 and (5.11, the weighting vector $\vec{\beta}$ is already incorporated into $\mathcal{J}_{rec}$ and $\mathcal{J}_{dis}$ via its role in $\hat{\Phi}(\mathcal{X})$. Hence, minimizing them w.r.t. to $\vec{\beta}$ optimizes the combined embedding in the features spaces to fulfill the objectives **Ob1** and **Ob2** better. Besides, as a practical complement, I optimize $\vec{\beta}$ to separate the classes locally in k-size neighborhoods. I propose $\mathcal{J}_{ls}$ as the following novel, convex loss:

$$\mathcal{J}_{ls}(\vec{\beta}) = \sum_{i=1}^{N} \Big[\sum_{s \in \mathcal{N}_i^k} \|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \sum_{s \in \overline{\mathcal{N}_i^k}} \hat{\Phi}(\mathbf{X}_i)^\top \hat{\Phi}(\mathbf{X}_s) \Big], \tag{5.13}$$

where $\mathcal{N}_i^k$ specifies the same-label k-nearest neighbors of $\mathbf{X}_i$ on the RKHS, and $\overline{\mathcal{N}_i^k}$ is its corresponding k-size set for the different-label neighbors of $\mathbf{X}_i$. Equation 5.13 reaches its minima when for each $\mathbf{X}_i$, we have both of:

1. The summation of its distances to the nearby same-label points is minimized.

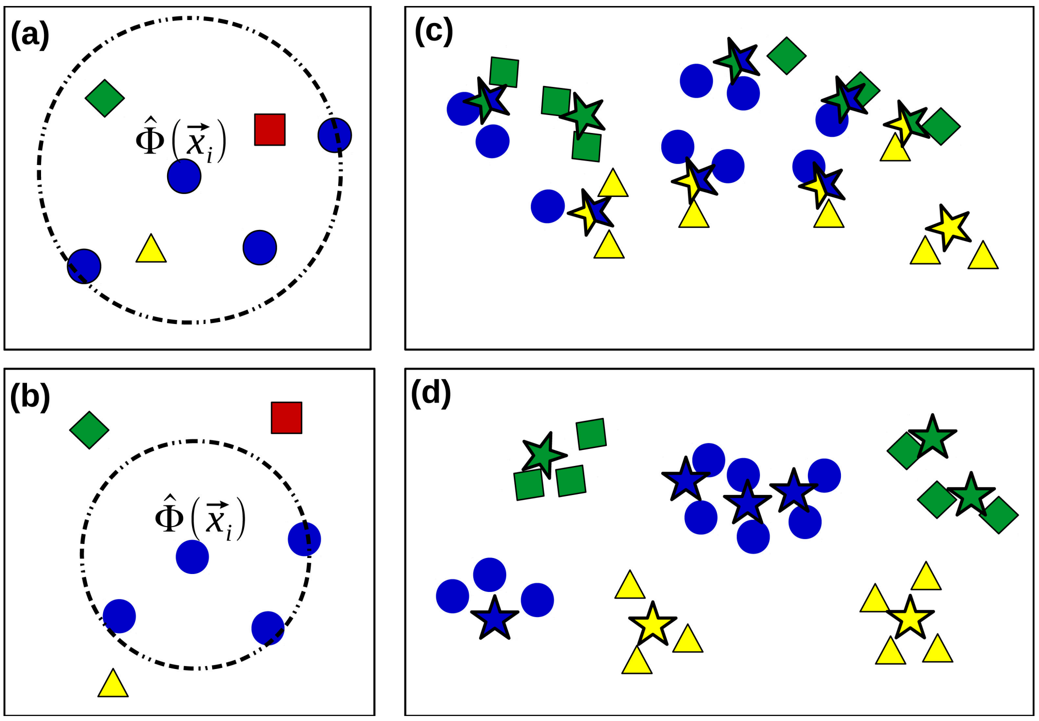


Figure 5.3: Effect of $\mathcal{J}_{ls}$ on the local separation of each $\hat{\Phi}(\mathbf{X}_i)$ from its different-label neighbors in RKHS when $k = 4$ (**b** compared to **a**), which concentrates the classes locally (**d** compared to **c**) and improves the interpretation of the prototypes $\{\hat{\Phi}(\mathcal{X})\vec{u}_i\}_{i=1}^{k}$ (the stars) by the class-neighborhood to which they are assigned (their colors).

2. It is dissimilar from the nearby data of other classes (Figure 5.3-b).

Therefore, having $\mathcal{J}_{ls}$ in conjunction with other terms in Equation 5.11 makes the classes locally condensed and distinct from each other, facilitating learning better interpretable, discriminative prototypes (Figure 5.3-d). In the next section, I explain how to solve the optimization problem of Equation 5.11 efficiently.

Optimization Scheme of IMKPL

After rewriting the optimization problem of Equation 5.11 using the given definitions for $\{\mathcal{J}_{dis}, \mathcal{J}_{ls}, \mathcal{J}_{ip}\}$, I optimize its parameters $\{\mathbf{U}, \mathbf{\Gamma}, \vec{\beta}\}$ by adopting the alternating optimization scheme.

Proposition 5.2. *Denoting* $\mathbf{U} \in \mathbb{R}^{N\times k}$, $\mathbf{\Gamma} \in \mathbb{R}^{k\times N}$, $\vec{\beta} \in \mathbb{R}^{d}$, *and*

$$\begin{aligned}\mathcal{G}(\mathbf{U},\mathbf{\Gamma},\vec{\beta}) = \;& \|\hat{\Phi}(\mathcal{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 \\ & +\lambda\tfrac{1}{2}\sum_{i=1}^{N}\Big[\sum_{s=1}^{N}\vec{u}^s\vec{\gamma}_i(\vec{h}_i^\top\vec{h}_s\|\hat{\Phi}(\mathbf{X}_i)-\hat{\Phi}(\mathbf{X}_s)\|_2^2 + \|\vec{h}_i-\vec{h}_s\|_2^2)\Big] \\ & +\mu\sum_{i=1}^{N}\Big[\sum_{s\in\mathcal{N}_i^k}\|\hat{\Phi}(\mathbf{X}_i)-\hat{\Phi}(\mathbf{X}_s)\|_2^2 + \sum_{s\in\overline{\mathcal{N}_i^k}}\hat{\Phi}(\mathbf{X}_i)^\top\hat{\Phi}(\mathbf{X}_s)\Big] + \tau\|\mathbf{H}\mathbf{U}\|_1,\end{aligned}$$

the objective function $\mathcal{G}(\mathbf{U},\mathbf{\Gamma},\vec{\beta})$ *is multi-convex in terms of* $\{\mathbf{\Gamma},\mathbf{U},\vec{\beta}\}$.

Proof. Refer to Appendix A.12. □

Benefiting from Proposition 5.2, at each of the following alternating steps, I update only one of the parameters while fixing the others (Algorithm 5.1). The derivation of the following sub-problems is provided in the supplementary material.

Updating the Matrix of Sparse Codes $\mathbf{\Gamma}$

By fixing $\{\mathbf{U}, \vec{\beta}\}$, using Equation 5.3, and removing the constant terms, I reformulate Equation 5.11 w.r.t. each $\vec{\gamma}_i$ as:

$$\begin{aligned} \min_{\vec{\gamma}_i} \quad & \vec{\gamma}_i^\top (\mathbf{U}^\top \hat{\mathcal{K}} \mathbf{U}) \vec{\gamma}_i + [\lambda \tilde{\mathcal{K}}(i,:) - 2\hat{\mathcal{K}}(i,:)] \mathbf{U} \vec{\gamma}_i \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T, \ \ \gamma_{ji} \in \mathbb{R}_{\geq 0}, \end{aligned} \tag{5.14}$$

where $\tilde{\mathcal{K}} = \mathbf{1} - (\mathbf{H}^\top \mathbf{H}) \odot \hat{\mathcal{K}}$, and "$\odot$" denotes the Hadamard product operator. This optimization problem is a non-negative quadratic programming problem with a cardinality constraint on $\vec{\gamma}_i$. The matrix $\mathbf{U}^\top \hat{\mathcal{K}} \mathbf{U}$ is positive semidefinite (PSD) because $\hat{\mathcal{K}}$ is PSD and $\mathbf{U}$ is non-negative. Hence, Equation 5.14 is a convex problem, and I efficiently solve it by using the proposed NQP algorithm of Section 4.3 (Algorithm 4.5). Hence, I update the columns of $\mathbf{\Gamma}$ individually.

Updating Prototype Matrix U

Similar to the approximation of $\mathbf{\Gamma}$, the prototype vectors $\vec{u}_i$ are updated sequentially. I rewrite the reconstruction objective $\mathcal{J}_{rec}$ in Equation 5.11 as

$$\|\hat{\Phi}(\mathbf{X})\mathbf{E}_i - \hat{\Phi}(\mathbf{X})\vec{u}_i \vec{\gamma}^i\|_F^2, \quad \mathbf{E}_i = (\mathbf{I} - \textstyle\sum_{j \neq i} \vec{u}_j \vec{\gamma}^j), \tag{5.15}$$

where $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix. By using Equation 5.15 and writing $\mathcal{J}_{dis}$ in terms of $\vec{u}_i$, I reformulate Equation 5.11 as

$$\begin{aligned} \min_{\vec{u}_i} \quad & \vec{u}_i^\top (\vec{\gamma}^i \vec{\gamma}^{i^\top} \hat{\mathcal{K}}) \vec{u}_i + [\vec{\gamma}^i (-2\mathbf{E}_i^\top \hat{\mathcal{K}} + \lambda \tilde{\mathcal{K}}) + \tau \vec{1} \vec{1}^\top \mathbf{H}] \vec{u}_i \\ \text{s.t.} \quad & \|\vec{u}_i\|_0 < T, \ \ \|\hat{\Phi}(\mathcal{X})\vec{u}_i\|_2^2 = 1, \ \ u_{ji} \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{5.16}$$

Analogous to Equation 5.14, this is a convex non-negative quadratic problem in terms of $\vec{u}_i$ with a hard limit on $\|\vec{u}_i\|_0$. Hence, I update the prototype vectors $\{\vec{u}_i\}_{i=1}^k$ by solving Equation 5.16 using the NQP algorithm. For updating each $\vec{u}_i$, I update its corresponding $\mathbf{E}_i$ and normalize vector $\vec{u}_i$ afterward, similar to the update steps of the CKSC method in Section 4.3.

Updating Kernel Weights $\vec{\beta}$

By normalizing each base kernel $\mathcal{K}_m$ in advance, I can simplify Equation 5.11 to the following linear programming (LP) problem

$$\begin{aligned} \min_{\vec{\beta}} \quad & (\vec{\mathcal{E}}_{rec} + \lambda \vec{\mathcal{E}}_{dis} + \mu \vec{\mathcal{E}}_{ls})^\top \vec{\beta} \\ \text{s.t.} \quad & \textstyle\sum_{m=1}^d \beta_m = 1, \ \ \beta_m \in \mathbb{R}_{\geq 0}, \end{aligned} \tag{5.17}$$

where I derive the entries of $\vec{\mathcal{E}}_{rec}$, $\vec{\mathcal{E}}_{dis}$, and $\vec{\mathcal{E}}_{ls}$ by incorporating Equation 5.3 into the terms $\mathcal{J}_{rec}$, $\mathcal{J}_{dis}$, and $\mathcal{J}_{ls}$, respectively. I compute their m-th entries ($m = 1, \ldots, d$) as

$$\begin{aligned} \mathcal{E}_{rec}(m) &= \text{tr}[\mathcal{K}_m(\mathbf{I} - 2\mathbf{U}\mathbf{\Gamma}) + \mathbf{\Gamma}^\top \mathbf{U}^\top \mathcal{K}_m \mathbf{U}\mathbf{\Gamma}], \\ \mathcal{E}_{dis}(m) &= \text{tr}(\tilde{\mathcal{K}}_m \mathbf{U}\mathbf{\Gamma}), \\ \mathcal{E}_{ls}(m) &= \sum_{i=1}^N \sum_{s \in \mathcal{N}_i^k} [2 - 2\mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_s)] + \sum_{s \in \overline{\mathcal{N}_i^k}} \mathcal{K}_m(\mathbf{X}_i, \mathbf{X}_s), \end{aligned} \tag{5.18}$$

Algorithm 5.1 Interpretable Multiple-Kernel Prototype Learning algorithm: learns a set of prototypes $\{\vec{u}_i\}_{i=1}^k$, a weighting scheme of the given kernels, and the matrix of sparse codes $\mathbf{\Gamma}$ as the approximate solution to Equation 5.11.

```
1:  Parameters: Weights $\{\lambda, \mu, \tau\}$, sparsity $T$, neighborhood size $k$, and stopping threshold $\delta$.
2:  Input: Label matrix $\mathbf{H}$, kernel functions $\{\mathcal{K}_m(\mathcal{X},\mathcal{X})\}_{m=1}^d$.
3:  Output: Prototype vectors $\{\vec{u}_i\}_{i=1}^k$, kernel weights $\vec{\beta}$, encoding matrix $\mathbf{\Gamma}$.
4:  Initialization: Computing $\{\tilde{\mathcal{K}}, \{\tilde{\mathcal{K}}_m\}_{i=1}^d, \vec{\mathcal{E}}_{ls}\}$, $\vec{\beta} = \vec{\mathbf{1}}$.
5:  while [whole objective of Equation 5.11] $> \delta$ do
6:      Computing $\hat{\mathcal{K}}(\mathcal{X},\mathcal{X}) = \sum_{m=1}^d \beta_m \mathcal{K}_m(\mathcal{X},\mathcal{X})$.
7:      Updating $\mathbf{\Gamma}$ based on Equation 5.14 using NQP.
8:      Updating $\mathbf{U}$ based on Equation 5.16 using NQP.
9:      Updating $\vec{\beta}$ based on Equation 5.17 using an LP solver.
10: end while
```

where $\tilde{\mathcal{K}}_l$ is derived by computing $\tilde{\mathcal{K}}$ while replacing $\mathcal{K}$ with $\mathcal{K}_m$. Therefore, I can efficiently solve the LP in Equation 5.17 using conventional linear solvers (Strayer 2012). Algorithm 5.1 provides an overview of all the optimization steps for my IMKPL framework.

Representation of the Test Data

To represent (reconstruct) a test data $\mathbf{Z}$ by the trained $\mathbf{U}$ and $\vec{\beta}$, I compute the sparse code $\vec{\gamma}_{test}$ using Equation 5.14 while setting $\lambda = 0$. The relational values of the entries in $\vec{\gamma}_{test}$ show the main prototypes that are used to represent $\mathbf{Z}$.

Complexity and Convergence of IMKPL

In order to calculate the computational complexity of IMKPL per iteration, I analyze the update of each $\{\mathbf{\Gamma}, \mathbf{U}, \vec{\beta}\}$ individually. In each iteration, the update of $\mathbf{\Gamma}$ and $\mathbf{U}$ are done using the NQP algorithm, which has the time complexity of $\mathcal{O}(nT)$, where n is the number of dimensions in the quadratic problem. I also set $k = CT$ as an effective choice in my model, while in practice, the maximum number of non-zero elements of $\vec{\gamma}^i$ in Equation 5.16 is smaller than $\frac{N}{C}$.

Therefore, optimizing $\mathbf{\Gamma}$ and $\mathbf{U}$ leads to $\mathcal{O}(CNT^2 + CTN^2)$ and $\mathcal{O}(CNT^2 + TN^2 + CN)$ computational costs, respectively, and optimizing $\vec{\beta}$ with an LP solver has the computational complexity of $\mathcal{O}(2td + dN^2 + dkN)$, where t is the convergence iteration of the LP-solver. The time-consuming matrix multiplications of Equation 5.18 are already carried out while solving Equation 5.14 and 5.16.

As in the implementations, we observe/choose $C, T, k << N$ (eps. for large-scale datasets), the computational complexity of IMKPL in each iteration is approximately $\mathcal{O}(dN^2 + N^2)$. Therefore, IMKPL is more scalable than its alternative MK algorithms (X. Zhu et al. 2017; Thiagarajan, Ramamurthy, and Spanias 2014; Shrivastava, Pillai, and Patel 2015) which have complexity close to $\mathcal{O}(dN^3)$.

In the experiments of Section 5.5, I present the convergence curve of the IMKPL algorithm, which stops in less than 20 iterations for all the selected real-datasets. Despite that, the following theorem guarantees the convergence of Algorithm 5.1:

Theorem 5.1. *The iterative updating procedure in Algorithm 5.1 converges to a locally optimal point in a limited number of iterations.*

Proof. Refer to Appendix A.13. □

Comparison to the CKSC Algorithm from Chapter 4

We can convert the problem of Equation 5.11 into a single-kernel formulation by setting all entries of β equal to 1. In that case, we obtain a kernel-based prototype learning algorithm as the single-kernel variant of the IMKPL algorithm (ISKPL). At first sight, the formulation of ISKPL may look similar and comparable to the proposed CKSC framework of Chapter 4. Although IMKPL is constructed upon the NNKSC 's non-negative framework, it principally differs from the CKSC algorithm from the following specific points:

- The ISKPL formulation focuses explicitly on local same-class construction of dictionary atoms, while CKSC has more freedom in that regard. Therefore, in comparison, it is expected from ISKPL to learn a dictionary $\Phi(\mathcal{X})\mathbf{U}$ with more interpretable entries $\Phi(\mathcal{X})\vec{u}_i$.
- On the other hand, these methods also differ regarding their discriminative terms, which encode the supervised information into the sparse vectors. While ISKPL focuses on representing data points in the feature space by their neighboring prototypes for better interpretation, CKSC represents a more consistent test/train framework which shows more robustness regarding the discriminative encoding of motion sequences.

Therefore, while ISKPL is more practical for the prototype-based representation of motion data, CKSC is more effective for discriminative encoding of such data. Regardless of the above differences, I empirically demonstrate in the experiments of Section 5.5 that both methods perform better than other state-of-the-art alternative methods w.r.t. model interpretation and the discriminative quality of their encoding.

5.4 multiple-kernel dictionary structure

Although there are many annotated benchmark motion datasets, a typical observation for real-world motion analysis tasks is encountering unseen motion classes. This is a significant problem for uncontrolled environments such as CCTV camera recordings or social robots in public environments. In zero-shot learning, as a specific branch of machine learning, there exist many approaches for description and recognition of unseen classes in datasets. Nevertheless, it becomes a challenging problem when dealing with multivariate time-series (MTS) (e.g., motion data), where we cannot directly apply such vectorial algorithms to the temporal inputs.

On the other hand, in previous sections of this chapter, we observed that component-wise analysis of motion sequences might reveal more semantic characteristics about the underlying movement. More specifically, some joint (dimensions) movements can represent particularities, while others can reveal commonalities between different classes of motion.

Based on the above perspective, I propose a novel multiple-kernel dictionary (MKD) learning in this section, which learns semantic attributes based on specific combinations of MTS dimensions in the feature space. Hence, the MKD can fully or partially reconstruct the unseen classes by means of interpretable connections to the observed training samples (seen classes). Furthermore, the sparse encodings of unseen classes based on the attributes of the learned MKD are used in a proposed incremental clustering algorithm to categorize the unseen MTS classes in an unsupervised way.

I consider the training set of mocap sequences $\mathcal{X} = \{\mathbf{X}_i\}_{i=1}^N$ belongs to C distinct data classes with the corresponding label set $\mathcal{H} = \{1, \cdots, C\}$. Accordingly, the set of unseen sequences $\mathcal{Z}$ belongs to the label set $\mathcal{Q}$, such that $\mathcal{Q} \cap \mathcal{H} = \emptyset$. Based on the above description, we are interested in:

1. Obtaining semantic attributes that create interpretable relations between sequences $\mathbf{Z}_i \in \mathcal{Z}$ and the seen classes in $\mathcal{X}$ (Figure 5.4).
2. Using the learned attributes for efficient clustering of the unseen set $\mathcal{Z}$.

Similar to Figure 5.4, it is common to observe for real-world MTS data (e.g., human motions) to find partial similarities between different data classes when considering a subset of their dimensions. Therefore, these similarities can lead to an interpretable description for a novel data sample (from $\mathcal{Z}$) via its relation to the seen classes (from $\mathcal{X}$). Furthermore, such a description leads to a better clustering of novel data points $\mathbf{Z}_i$ without having any prior information on their class labels. To achieve the above, I propose an MKD model trained based on $\mathcal{X}$ and learns semantic attributes similar to Figure 5.4-left. To be more specific, MKD combines dimensions of similar MTS samples in the feature space under non-negativity constraints. These attributes can encode each unseen $\mathbf{Z}_i \in \mathcal{Z}$ as an interpretable description of its dimensions and better separate it from previous (unknown) classes in $\mathcal{Z}$ (Figure 5.4-right).

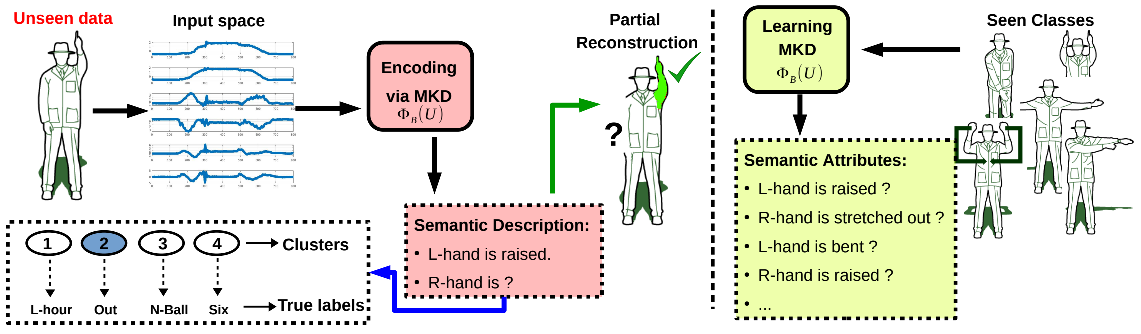


Figure 5.4: General overview of the MKD framework. The dictionary learns the semantic attributes based on the seen classes. These attributes are used for the interpretable description of unseen class data, which leads to categorizing and partial reconstruction of the data.

For a d-dimensional motion sequence, I assume there exist d non-linear implicit kernel functions $\{\Phi_m(\mathbf{X})\}_{m=1}^{d}$ to map each dimension of $\mathbf{X}$ into an individual Reproducing Kernel Hilbert Spaces as $\Phi_m : (\mathbb{R}^d)^* \to \mathbb{R}^{d_m}$. Hence, defining $\Phi(\mathbf{X}, \vec{\beta})$ as

$$\Phi(\mathbf{X}, \vec{\beta}) = [\sqrt{\beta_1}\Phi_1^\top(\mathbf{X}), \dots, \sqrt{\beta_d}\Phi_d^\top(\mathbf{X})]^\top \tag{5.19}$$

describes a weighted combination of these kernels with the non-negative coefficient vector $\vec{\beta} \in \mathbb{R}^d$, which induces an embedding of the data into the feature space. I can apply this embedding to the whole training data via

$$\Phi(\mathcal{X}, \vec{\beta}) := [\Phi(\mathbf{X}_1, \vec{\beta}) \cdots \Phi(\mathbf{X}_N, \vec{\beta})]. \tag{5.20}$$

On the other hand, it is rational to assume structures of different class-specific subspaces in the feature space correspond to k different weighting schemes of the individual kernels as $\{\vec{\beta}_i\}_{i=1}^{k}$. Hence, based on a weighting matrix

$$\mathbf{B} = [\vec{\beta}_1 \cdots \vec{\beta}_k] \in \mathbb{R}^{d \times k}, \tag{5.21}$$

I define my novel multiple-kernel dictionary (MKD) matrix $\Phi_{\mathbf{B}}(\mathbf{U})$ as

$$\Phi_{\mathbf{B}}(\mathbf{U}) := [\Phi(\mathcal{X}, \vec{\beta}_1)\vec{u}_1 \cdots \Phi(\mathcal{X}, \vec{\beta}_k)\vec{u}_k] \qquad \text{where } \mathbf{U} = [\vec{u}_1 \dots \vec{u}_k] \in \mathbb{R}^{N \times k}. \tag{5.22}$$

Each dictionary column $\Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i$ in Equation 5.22 is a weighted combination of selected dimensions and selected samples from $\mathcal{X}$ based on the value of $\vec{\beta}_i$ and $\vec{u}_i$, respectively. Due to the relation of $\Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i$ to different dimensions of $\mathcal{X}$, its columns can learn semantic attributes similar to those of Figure 5.4.

To make a comparison between the structure of $\Phi_{\mathbf{B}}(\mathbf{U})$ and a more conventional multiple-kernel dictionary structure as in IMKPL algorithm ($\hat{\Phi}(\mathcal{X})\mathbf{U}$), one must consider each individual atom thereof. More specifically, each all the dictionary atoms of $\hat{\Phi}(\mathcal{X})\mathbf{U}$, as in Equation 5.11, are formed in the same features space under the scaling vector $\vec{\beta}$. In contrast, each column of $\Phi_{\mathbf{B}}(\mathbf{U})$ from Equation 5.22 is shaped in an individual RKHS scaled by one column of $\mathbf{B}$ (Equation 5.21). Applying that to motion data, each atom of the proposed MKD structure can be related to the movement of a specific set of body joints, while all atoms of $\hat{\Phi}(\mathcal{X})\mathbf{U}$ are connected to a global set of body joints.

To fit $(\mathbf{U}, \mathbf{B})$ to the data efficiently, I aim for the reconstruction of training samples in the feature space as $\Phi(\mathcal{X}) \approx \Phi_{\mathbf{B}}(\mathbf{U})\mathbf{\Gamma}$ with a sparse encoding matrix $\mathbf{\Gamma}$. Hence, I propose the following MKD sparse coding framework (MKD-SC) for training the dictionary parameters $(\mathbf{B}, \mathbf{U})$ and sparse codes $\mathbf{\Gamma}$:

$$\begin{aligned} \min_{\mathbf{B},\mathbf{\Gamma},\mathbf{U}} \quad & \|\Phi(\mathcal{X}) - \Phi_{\mathbf{B}}(\mathbf{U})\mathbf{\Gamma}\|_F^2 \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T, \;\; \|\vec{u}_i\|_0 < T, \\ & \|\Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i\|_2^2 = 1, \;\; u_{ij}, \beta_{ij}, \gamma_{ij} \in \mathbb{R}_{\geq 0}, \;\; \forall ij, \end{aligned} \tag{5.23}$$

The loss term in Equation 5.23 measures the encoding's reconstruction error given the multiple-kernel dictionary $\Phi_{\mathbf{B}}(\mathbf{U})$ and the sparse codes $\mathbf{\Gamma}$. Similar to other sparse coding frameworks in this chapter and Chapter 4, the l_2-norm constraint on $\Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i$ prevents the optimization solutions from becoming degenerated (Rubinstein, Zibulevsky, and Michael Elad 2008).

The dictionary $\Phi_{\mathbf{B}}(\mathbf{U})$ in Equation 5.23 contains attributes (columns) that are weighted combinations of different exemplars and dimensions from $\mathcal{X}$. The non-negativity constraints on $(\mathbf{B}, \mathbf{\Gamma}, \mathbf{U})$ leads to a combination of similar semantically similar sequences in such formulation. In addition, applying the sparsity limit T on the cardinality of each $\vec{\gamma}_i$ and $\vec{u}_i$ motivates the encoding model to learn significant, non-redundant information in the data. As a result, the model proposed in Equation 5.23 learns attributes as columns of $\Phi_{\mathbf{B}}(\mathbf{U})$, which lead to interpretable encoding vectors $\vec{\gamma}_i$. Each entry in an encoded $\vec{\gamma}_i$ can be interpreted according to the specific dimensions in the particular motion sequences to which it is connected.

In the following sections, I show how one can benefit from this proposed MKD model to categorize and partially reconstruct unseen motion sequences (or, in general, MTS data).

Partial Reconstruction of Unseen Motions

In real-world MTS datasets such as human motions, it is expected to observe partial similarities between different motion classes' dimensions. Therefore, some body movements in one motion can be described by the same body movements in another motion. For example, one can make a similarity mapping between the leg movements in two generally different actions performed while the subject is walking. In such a scenario, given the first motion (**X**) is from the training set and the latter is an unseen sequence (**Z**), we can reconstruct (or represent) the leg movement of **Z** by addressing the leg dimension of **X**. In other words, we can partially encode **Z**.

To obtain an encoding vector $\vec{\gamma}$ for an unseen **Z**, we can minimize the problem of Equation 5.23 only regarding $\vec{\gamma}$ as follows

$$\begin{aligned} \vec{\gamma} = \quad & \underset{\vec{\gamma}}{\arg\min} \quad \|\Phi(\mathbf{Z}) - \Phi_{\mathbf{B}}(\mathbf{U})\vec{\gamma}\|_F^2 \\ & \text{s.t.} \qquad \|\vec{\gamma}\|_0 < T, \gamma_i \in \mathbb{R}_{\geq 0} \end{aligned} \tag{5.24}$$

To find the dimensions in **Z** which can be partially encoded according to the resulting $\vec{\gamma}$, I define the following error measure:

$$\mathcal{J}_{rec}^{\mathcal{S}}(\mathbf{Z}, \mathbf{B}, \mathbf{U}) = \|\mathbf{I}_{\mathcal{S}}\Phi(\mathbf{Z}) - \Phi_{\mathbf{B}_{\mathcal{S}}}(\mathbf{U})\vec{\gamma}\|_2^2 / \|\mathbf{I}_{\mathcal{S}}\Phi(\mathbf{Z})\|_2^2, \tag{5.25}$$

in which $\mathcal{S}$ denotes the index set of selected dimensions from **Z**. Notations $\mathbf{B}_{\mathcal{S}}$ and $\mathbf{I}_{\mathcal{S}}$ are **B** and an identity matrix, respectively, where all their entries are zero except the rows in **B** and diagonal elements of **I** corresponding to the index set $\mathcal{S}$.

Consequently, the learned dictionary $\Phi_{\mathbf{B}}(\mathbf{U})$ can partially reconstruct the unseen time-series **Z** for the subset $\mathcal{S}$ of its dimensions, if for a chosen relatively small ϵ:

$$\begin{aligned} \mathcal{S} = \quad & \underset{\mathcal{S}}{\arg\max} \quad |\mathcal{S}| \\ & \text{s.t.} \qquad \mathcal{J}_{rec}^{\mathcal{S}}(\mathbf{Z}, \mathbf{B}, \mathbf{U}) < \epsilon \end{aligned} \tag{5.26}$$

The parameter ϵ in Equation 5.26 makes a trade-off between the quality of the encoding and the number of dimensions from **Z** that are reconstructed in RKHS.

As the primary step to compute Equation 5.25, based on the dot-product rule in Equation 5.3 for each individual kernel $\mathcal{K}_m(\mathcal{X}, \mathcal{X})$ and the definition of MKD in Equations 5.22 and 5.20, I denote the following

$$\mathcal{K}_{\mathbf{B}_{\mathcal{S}}}^{ij}(\mathbf{X}_\eta, \mathbf{X}_\xi) := \Phi(\mathbf{X}_\eta, \vec{\beta}_i)^\top \Phi(\mathbf{X}_\xi, \vec{\beta}_j) = \sum_{m \in \mathcal{S}} \beta_{mi}\beta_{mj}\mathcal{K}_m(\mathbf{X}_\eta, \mathbf{X}_\xi). \tag{5.27}$$

Algorithm 5.2 Incremental Clustering algorithm: incrementally categorizes the encoded unseen motion sequence **Z** based on its dimension-wise reconstruction. The clustering algorithm constructs the dendrogram $\mathcal{H}$ for those sequences in an online fashion.

```
1: **Input:** The encoding matrix $\mathbf{R}$ from Equation 5.29, the current tree $\mathcal{H}$.
2: **Output:** Place of $\mathbf{Z}$ in the hierarchy $\mathcal{H}$.
3: **if** $\exists C_n$ such that $d(\mathbf{Z}, C_n) \leq \bar{d}(C_n)$ **then**
4:     **if** $C_n$ is a leaf node **then**
5:         add $\mathbf{Z}$ to $C_n$.
6:         **if** $(\bar{d}(C_{n1}) + \bar{d}(C_{n2}))/2\bar{d}(C_n) \leq k_{clust}$ **then**.
7:             split $C_n$ into $C_{n1}$ and $C_{n2}$ using $k$-means.
8:             **if** $(\bar{d}(C_{n1}) + \bar{d}(C_{n2}))/2\bar{d}(C_n) \leq k_{rmv}$ **then**
9:                 Replace $C_n$ with $C_{n1}$ and $C_{n2}$.
10:            **else**
11:                add $\{C_{n1}, C_{n2}\}$ as the children of $C_n$.
12:            **end if**
13:        **end if**
14:    **else**
15:        Create a new child for $C_n$ as $C_{n_t}$ and add $z$ to it.
16:    **end if**
17: **else**
18:    Create a new leaf at the top level containing $\mathbf{Z}$.
19: **end if**
```

In Equation 5.27, the matrix $\mathcal{K}_m(\mathbf{X}_\eta, \mathbf{X}_\xi)$ is the kernel function associated with the m-th implicit mapping $\Phi_m(\mathbf{X})$ and is computed based on the m-th feature of the motion sequences. Using Equation 5.27, we can rewrite Equation 5.23 in terms of the parameters $(\mathbf{U}, \mathbf{B}, \mathbf{\Gamma})$ and update each parameter individually. Hence, we can be compute

$$\mathcal{J}^{\mathcal{S}}_{rec}(\mathbf{Z}, \mathbf{B}, \mathbf{U}) = [\vec{\gamma}^\top \mathbf{M} \vec{\gamma} + \vec{v}^\top \vec{\gamma} + \sum_{m \in \mathcal{S}} \mathcal{K}_m(\mathbf{Z}, \mathbf{Z})] / \sum_{m \in \mathcal{S}} \mathcal{K}_m(\mathbf{Z}, \mathbf{Z}), \tag{5.28}$$

where $m_{ij} = \vec{u}_i^\top \mathcal{K}^{ij}_{\mathbf{B}_\mathcal{S}}(\mathcal{X}, \mathcal{X}) \vec{u}_j$ and $v_j = -2\mathcal{K}^{\mathbf{1}j}_{\mathbf{B}_\mathcal{S}}(\mathbf{Z}, \mathcal{X}) \vec{u}_j$ are the entries of $\mathbf{M}$ and $\vec{v}$, respectively. The term $\mathcal{K}^{\mathbf{1}j}_{\mathbf{B}_\mathcal{S}}$ denotes using a vector of ones instead of β_i in Equation 5.27.

In the next section, I propose a clustering method based on the above interpretable encoding which describes a partial similarity between the unseen sequence **Z** and the members of the training set $\mathcal{X}$ in RKHS.

Incremental Clustering of Unseen Motions

I propose Algorithm 5.2 that relies on the partial similarity of different motion classes and the descriptive quality of the learned attributes of MKD. This algorithm incrementally clusters the unseen sequences of $\mathcal{Z}$ into a dendrogram $\mathcal{H}$ in an online fashion and also finds their potential sub-clusters. To that aim, for each unknown motion sequence **Z**, I prepare an encoding matrix $\mathbf{R} \in \mathcal{R}^{N \times d}$, i-th column of which represents the weights of contribution from $\mathcal{X}$ in the reconstruction of the i-th dimension of **Z**. Therefore, matrix **R** is constructed as

$$r_{ji} = \sum_{t=1}^{k} \beta_{it} u_{jt} \gamma_t \tag{5.29}$$

where r_{ji} denotes the j-th entry of the i-th column of $\mathbf{R}$. This matrix is considered as a rich encoded descriptor for dimensions of $\mathbf{Z}$ based on $\mathcal{X}$ and is used in Algorithm 5.2 to compare $\mathbf{Z}$ to the previously categorized unseen data in $\mathcal{H}$ to find the best place for $\mathbf{Z}$ in the dendrogram. Line 3 of the algorithm finds C_n as the most similar node to $\mathbf{Z}$ based on the distance term $d(\mathbf{Z}, C_n) = \|\mathbf{R_Z} - \overline{\mathbf{R}}_{C_n}\|_F^2$, and the intra-cluster distance for each node C_n as $\bar{d}(C_n) = E_{\mathbf{Z}_i \in C_n}[d(\mathbf{R}_{\mathbf{Z}_i}, \overline{\mathbf{R}}_{C_n})]$, where $\overline{\mathbf{R}}_{C_n} = E_{\mathbf{Z}_i \in C_n}[\mathbf{R}_{\mathbf{Z}_i}]$. Regarding line 8, I choose $k_{rmv} = 0.3$ in our experiments, which results in an acceptable clustering outcome.

Optimization Scheme

I optimize the parameters $\mathbf{U}$, $\boldsymbol{\Gamma}$, and $\mathbf{B}$ in alternating steps, such that at each update step, I optimize Equation 5.23 with respect to one parameter while fixing the others. In the following update steps, the notation $\mathcal{K}_{\mathbf{B}}^{ij}(\mathbf{X}_\eta, \mathbf{X}_\xi)$ refers to $\mathcal{K}_{\mathbf{B}_\mathcal{S}}^{ij}(\mathbf{X}_\eta, \mathbf{X}_\xi)$ while the full matrix $\mathbf{B}$ is used instead of $\mathbf{B}_\mathcal{S}$. Consequently, the entries of matrix $\mathcal{K}_{\mathbf{B}}^{ij}(\mathcal{X}, \mathcal{X})$ are filled with the corresponding values of $\mathcal{K}_{\mathbf{B}_\mathcal{S}}^{ij}(\mathbf{X}_\eta, \mathbf{X}_\xi)$ $\forall \xi, \eta = 1, \ldots, N$. Using this summarized notation, we can rewrite Equation 5.23 in terms of each parameter $(\mathbf{U}, \mathbf{B}, \boldsymbol{\Gamma})$, and update that parameter individually.

Updating Sparse Codes $\boldsymbol{\Gamma}$

By fixing $\mathbf{U}$ and $\mathbf{B}$ and removing the constant terms w.r.t. $\boldsymbol{\Gamma}$, the optimization problem of Equation 5.23 is reduced to the following framework, which optimizes each individual sparse code $\vec{\gamma}$ corresponding to each single input $\mathbf{X}$

$$\begin{array}{ll} \min\limits_{\vec{\gamma}} & \frac{1}{2}\vec{\gamma}^\top \mathbf{M} \vec{\gamma} + \vec{v}^\top \vec{\gamma} \\ \text{s.t.} & \|\vec{\gamma}\|_0 < T, \quad \gamma_i \in \mathbb{R}_{\geq 0}, \quad \forall i, \end{array} \tag{5.30}$$

in which $m_{ij} = 2\vec{u}_i^\top \mathcal{K}_{\mathbf{B}}^{ij}(\mathcal{X}, \mathcal{X})\vec{u}_j$ and $v_j = -2\mathcal{K}_{\mathbf{B}}^{\mathbf{1}j}(\mathbf{X}, \mathcal{X})\vec{u}_j$ are the entries of $\mathbf{M}$ and $\vec{v}$, respectively. The term $\mathcal{K}_{\mathbf{B}}^{\mathbf{1}j}$ denotes using a vector of ones instead of β_i in Equation 5.3. The optimization problem in Equation 5.30 is a non-negative quadratic programming problem with an l_0-norm constraint on $\vec{\gamma}$. Therefore, it can be optimized via the NQP algorithm from Section 4.3 (Algorithm 4.5).

Updating the Dictionary Matrices $\mathbf{U}, \mathbf{B}$

I update the vectors associated with the dictionary atoms individually. To do so, for each pair of $\{\vec{u}_i, \vec{\beta}_i\}$, the loss terms of Equation 5.23 can be reformulated as

$$\mathcal{J}_{rec}(\mathcal{X}, \boldsymbol{\Gamma}, \mathbf{U}, \mathbf{B}) = \|\Phi(\mathcal{X}) - \sum_{i \neq j} \Phi(\mathcal{X}, \vec{\beta}_j)\vec{u}_j \vec{\gamma}^j - \Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i \vec{\gamma}^i\|_F^2 \tag{5.31}$$

Via further simplifying $\mathcal{J}_{dic}$ loss, Equation 5.23 can be re-formulated in terms of $\vec{u}_i$ as

$$\begin{array}{ll} \min\limits_{\vec{u}_i} & \vec{u}_i^\top [(\vec{\gamma}^i \vec{\gamma}^{i\top} \mathcal{K}_{\mathbf{B}}^{ii}(\mathcal{X}, \mathcal{X})) \\ & -2\vec{\gamma}^i [\mathcal{K}_{\mathbf{B}}^{\mathbf{1}i}(\mathcal{X}, \mathcal{X}) - \sum\limits_{i \neq j} \mathcal{K}_{\mathbf{B}}^{ij}(\mathcal{X}, \mathcal{X})\vec{u}^j \vec{\gamma}^j]^\top \vec{u}_i \\ \text{s.t.} & \|\vec{u}_i\|_0 < T, \quad \|\Phi(\mathcal{X}, \vec{\beta}_i)\vec{u}_i\|_2^2 = 1, \quad u_{ji} \in \mathbb{R}_{\geq 0} \quad \forall j, \end{array} \tag{5.32}$$

in which $\vec{\mathbf{1}}$ is a vector of ones, and the $diag(.)$ operator creates a vector based on the diagonal elements of its matrix argument. Similarly, via using Equation 5.31, the optimization problem for updating $\vec{\beta}_i$ is simplified as

$$\begin{aligned} \min_{\vec{\beta}_i} \quad & \tfrac{1}{2}\vec{\beta}_i^\top \mathbf{H}\vec{\beta}_i + \vec{c}^\top \vec{\beta}_i \\ \text{s.t.} \quad & \|\Phi(\mathcal{X},\vec{\beta}_i)\vec{u}_i\|_2^2 = 1, \;\; \beta_{ji} \in \mathbb{R}_{\geq 0} \quad \forall j \end{aligned} \tag{5.33}$$

where, $\mathbf{M}$ elements of $\mathbf{M}$ and $\vec{v}$ are computed as:

$$\begin{aligned} m_{jj} &= 2[\vec{\gamma}^i\vec{\gamma}^{i\top}\vec{u}_i^\top \mathcal{K}_j(\mathcal{X},\mathcal{X})\vec{u}_i + \lambda diag(\mathcal{K}_j(\mathcal{X},\mathcal{X}))^\top - \mathcal{K}_j(\mathcal{X},\mathcal{X})] \\ v_j &= 2[\sum_j \beta_{lj}(\vec{u}_i^\top \mathcal{K}_j(\mathcal{X},\mathcal{X})\vec{u}_j\vec{\gamma}^i\vec{\gamma}^{j\top}) - \vec{\gamma}^i\mathcal{K}_j(\mathcal{X},\mathcal{X})\vec{u}_i]. \end{aligned} \tag{5.34}$$

Based on Equation 5.34, the off-diagonal elements of $\mathbf{M}$ are all zero.

The optimization problem in Equation 5.32 is an instance of non-negative quadratic programming with an l_0-norm constraint on $\vec{u}_i$. Therefore, it can be optimized via the NQP algorithm from Section 4.3 (Algorithm 4.5). However, Equation 5.34 is an unconstrained non-negative quadratic problem, which can be solved by a non-negative QP method such as (Brand and D. Chen 2011; X. Xiao and D. Chen 2014). Furthermore, after updating each $\vec{u}_i$ or $\vec{\beta}_i$, they are adjusted as follows to normalize the dictionary atom $\Phi(\mathcal{X},\vec{\beta}_i)\vec{u}_i$.

$$\begin{aligned} \vec{u}_i &\leftarrow \frac{\vec{u}_i}{\|\Phi(\mathcal{X},\vec{\beta}_i)\vec{u}_i\|_2} = \frac{\vec{u}_i}{\sqrt{\vec{u}_i^\top \mathcal{K}_{\mathbf{B}}^{ii}\vec{u}_i}}, \\ \vec{\beta}_i &\leftarrow \frac{\vec{\beta}_i}{\|\Phi(\mathcal{X},\vec{\beta}_i)\vec{u}_i\|_2} = \frac{\vec{\beta}_i}{\sqrt{\vec{u}_i^\top \mathcal{K}_{\mathbf{B}}^{ii}\vec{u}_i}} \end{aligned} \tag{5.35}$$

Considering the above updates steps, the following represents the training loop for optimizing Equation 5.23:

1. Updating $\vec{\gamma}_i \;\; \forall i = 1,\dots,N$
2. Updating $\vec{u}_i$ and $\vec{\beta}_i$ in a subsequent order $\forall i = 1,\dots,k$

In the next section, I implement the proposed multiple-kernel methods of this chapter on real-world mocap benchmarks and evaluate their performance based on their specific purposes.

5.5 EXPERIMENTS

This section evaluates the performance of my proposed multiple-Kernel algorithms LMMK , IMKPL , and MKD-SC on real-world data. Since the proposed algorithms have different goals, I implement them with different setups and evaluate them via using their specific metrics. Particularly, for evaluation of LMMK and IMKPL , i use supervised settings, while the performance of MKD-SC is determined by unsupervised (clustering) measures.

Evaluating Large-Margin Multiple Kernel Learning

In this section, I implement my proposed LMMK algorithm on different motion datasets and evaluate its performance by carrying out empirical comparisons to other MKL alternative algorithms. For these experiments, I chose the following datasets: Schunk , , UTKinect , HDM05 , and CMU-9, CLL_SUB_111 , and `TOX_171` , which are introduced in Section 2.4. Except for the last two datasets, which are high-dimension vectorial data, other selected datasets are human motion capture benchmarks. The CLL_SUB_111 and `TOX_171` are specifically chosen to evaluate the performance of LMMK against high-dimensional, non-temporal data types. Additionally, more experiments on image benchmarks are also available in (Hosseini and Hammer 2019c), which analyzes the representation learning performance of my LMMK algorithm.

For all datasets, the base kernels $\{\mathcal{K}_m(\mathcal{X},\mathcal{X})\}_{m=1}^{d}$ are computed using the global alignment kernel (GAK) (Cuturi et al. 2007). Each $\mathcal{K}_m(\mathcal{X},\mathcal{X})$ is computed based on the pairwise DTW distances between motion sequences while considering only the m-th dimension. Hence, each base kernel represents one specific dimension of the motion sequence. Exceptionally for UTKinect , prior to GAK's application, I use the preprocessing from (Vemulapalli, Arrate, and Chellappa 2014) to obtain the Lie Group representation.

Parameters Tuning

The LMMK algorithm's hyper-parameters (k,μ,λ) are tuned throughout the cross-validation (CV) on the training set. However, based on practical evidence (Section 5.5), having $0.4 \leq \mu \leq 0.6$ and choosing the neighborhood radius as $1 \leq k \leq 5$ can lead to satisfactory performance. Furthermore, I advise the reader to tune (μ,k) first and find the optimal sparsity weight (λ) afterward. The above strategy can significantly reduce the parameter search space. Likewise, I tune the hyper-parameters of the baseline algorithms based on performing CV on the training set.

Alternative Methods

To have a proper evaluation, I make my comparisons between LMMK and the following major MKL algorithms: MKL-TR (W. Jiang and Chung 2014), MKL-DR (Y.-Y. Lin, T.-L. Liu, and Fuh 2011), DMKL (Qingwang Wang, Gu, and Tuia 2016), KNMF-MKL (Gu, Qingwang Wang, et al. 2015), and RMKL (Gu, C. Wang, et al. 2012). These algorithms are designed for multi-class MKL problems; hence, we can inspect their results from discriminative feature selection. For comparison, I also include the implementation result of my distance-based metric learning method DTW-LMNN from Chapter 3. As the baseline classifiers, I also implement multi-class SVM (C.-C. Chang and C.-J. Lin 2011) and kNN using the average of the base kernels resulting in SVM-ave and kNN -ave, respectively.

It is important to emphasize that the purpose of my sparse coding frameworks is to perform discriminative feature selection of motion data given multiple-kernel representation of data is available. Hence, although there exist various deep learning classifiers or object detection methods specially designed for image or video datasets, they do not fit the multiple-kernel scope of my comparisons. Furthermore, there exist state-of-the-art algorithms specifically designed for the classification of temporal data. They generally perform temporal segmentation or frame-based analysis of each data sequence. Therefore, these algorithms do not belong to the intended multiple-kernel

Table 5.1: Comparison of accuracies (*Acc*) and $||\vec{\beta}||_0$ on the MTS datasets.

Method	UTKinect		CMU-9		Schunk	
	Acc	$\|\|\vec{\beta}\|\|_0$	*Acc*	$\|\|\vec{\beta}\|\|_0$	*Acc*	$\|\|\vec{\beta}\|\|_0$
*k*NN -ave	85.70	60	85.34	62	82.32	64
SVM-ave	87.17	60	88.32	62	84.24	64
DLK	87.71	41	88.95	34	87.32	44
RMKL	90.09	55	89.57	50	88.47	56
KNMF-MKL	90.48	48	90.37	57	87.63	53
MKL-DR	90.84	31	91.73	40	88.91	37
DMKL	92.31	24	93.31	34	91.81	27
MKL-TR	93.20	20	93.66	21	92.73	**11**
DTW-LMNN	**98.92**	17	95.94	15	**96.82**	24
LMMK(proposed)	98.55	**14**	**96.72**	**12**	96.25	12
	HDM05		CLL_SUB_111		TOX_171	
	Acc	$\|\|\vec{\beta}\|\|_0$	*Acc*	$\|\|\vec{\beta}\|\|_0$	*Acc*	$\|\|\vec{\beta}\|\|_0$
*k*NN -ave	83.66	93	71.49	11340	78.26	5748
SVM-ave	85.74	93	74.52	11340	82.45	5748
RMKL	89.85	88	77.61	7563	84.57	3780
KNMF-MKL	87.81	89	76.23	6390	85.38	3579
MKL-DR	90.54	57	77.56	410	86.47	479
DMKL	92.71	36	79.34	**87**	89.73	89
MKL-TR	94.17	**20**	83.89	224	92.41	132
DTW-LMNN	97.06	23	**85.32**	383	97.43	105
LMMK(proposed)	97.07	23	84.63	118	**98.04**	**53**

The best result (**bold**) is according to a two-sample t-test at a 5% significance level.

scope of my experiments. Nevertheless, as a suggested extended experimental setting, one can use such methods as preprocessing techniques to obtain more discriminative base kernels for the multiple-kernel learning methods.

I evaluate the performance of the selected MKL algorithms based on classification accuracy $Acc = 100 \times [\texttt{\#correct predictions}]/N$. In order to evaluate the feature selection performance of the selected baselines, besides the classification accuracy (*Acc*), I also measure the number of selected features of the data (base kernels) via $||\vec{\beta}||_0$. For DTW-LMNN , $||\vec{\beta}||_0$ is obtained by regularizing the relevance profile of its learned metric the way described in Section 3.5. Consequently, a large *Acc* along with a small $||\vec{\beta}||_0$ describes an ideal discriminative feature selection, in which the classes could be distinguished with high accuracy while using a few selected features. All the experiments are done using 10-fold CV averaged over 10 repetitions. To preserve the possibility of comparing results against the experiments of Chapter 4, I use the exact CV indexes from that chapter.

Discriminative Feature Selection Results

Table 5.1 contains the implementation results of LMMK and other MKL algorithms on the selected MTS benchmarks. The LMMK algorithm outperforms other MKL baselines regarding classification accuracy. While LMMK has 7.63% and 4.7 higher accuracy compared to the best multiple-kernel learning baseline for TOX_171 and UTKinect datasets, this advance is 0.74% for the CLL_SUB_111 dataset. This observation shows that the local class-separation strategy's effectiveness varies among different datasets and depends on their class-distributions. Comparing the accuracy of LMMK to *k*NN , my proposed algorithm significantly increases the nearest neighbor classifier's performance. Specifically for the TOX_171 dataset, - in which *k*NN -ave has a relatively low accuracy due to its large number of features (11340) - LMMK optimization leads to a 19.78% increase in the performance of *k*NN . Considering other baselines, DMKL and MKL-TR alternatively take the second position in classification accuracy, which shows that the discriminative effect of the low-rank model in MKL-TR may vary depending on the given dataset. It is interesting to see that DTW-LMNN has a slightly better accuracy than LMMK for UTKinect , Schunk , and CLL_SUB_111 datasets, while LMMK outperform it for CMU Mocap and TOX_171 datasets. Hence, the sparse metric and centralized kernel representation used in LMMK can result in a more discriminative representation of the data compared to DTW-LMNN if its model suits the given class distribution. Generally, the comparison between discriminative quality of these two methods depends on the class distribution of the given task.

Regarding the feature selection performance, the value of $||\vec{\beta}||_0$ has ranked LMMK among the small-feature group of methods (DMKL, MKL-TR, LMMK), which is due to the direct application of an l_1-norm sparsity term in the optimization scheme of Equation 5.7. In comparison, MKL-TR obtained smaller values for $||\vec{\beta}||_0$ in CLL_SUB_111 and HDM05 datasets, while DMKL has the smallest feature set for Schunk . Nevertheless, these two methods showed lower classification accuracy in return. Therefore, I can claim that LMMK achieves more discriminative feature-selections even for these cases. To explain other baselines' feature selection results, DMKL and MKL-TR use a convex combination constraint on $\vec{\beta}$, which directly enforces sparsity, while MKL-DR and DLK have quadratic constraints on the kernel weights, which applies a weaker restriction on the number of non-zero kernel weights. On the other hand, KNMF-MKL and RMKL do not have any constraint in their optimization framework related to the sparseness of the selected features, which leads to a relatively poor feature selection result.

Compared to the DTW-LMNN method from Chapter 3, LMMK shows a slightly better feature selection. Although both methods have a similar optimization framework, the diagonal metric and the active sparsity objective of LMMK leads to selecting a tighter set of base kernels to represent motion sequences in a combined RKHS. However, as mentioned before, this small set of selected features may not always lead to better accuracy compared to DTW-LMNN . Therefore, we cannot make a unanimous vote regarding comparing the discriminative feature selection performance of these two methods.

Effect of the Parameter Setting

In this section, I study the effect of the parameters (λ, k, μ) on the performance of LMMK. As described in Figure 5.5, I perform three experiments on the CMU dataset, for each of which I study the algorithm's performance by changing one of the above parameters while fixing the two others.

At first, I change λ in the range $[0 \;\; 14]$ as in Figure 5.5-a. Based on the observations, I conclude that increasing the value of λ leads to a stronger sparsity force in Equation 5.7 and consequently results in a smaller set of selected features for both datasets. Figure 5.5-b shows that limited increases in λ can improve the classification accuracy, but large values

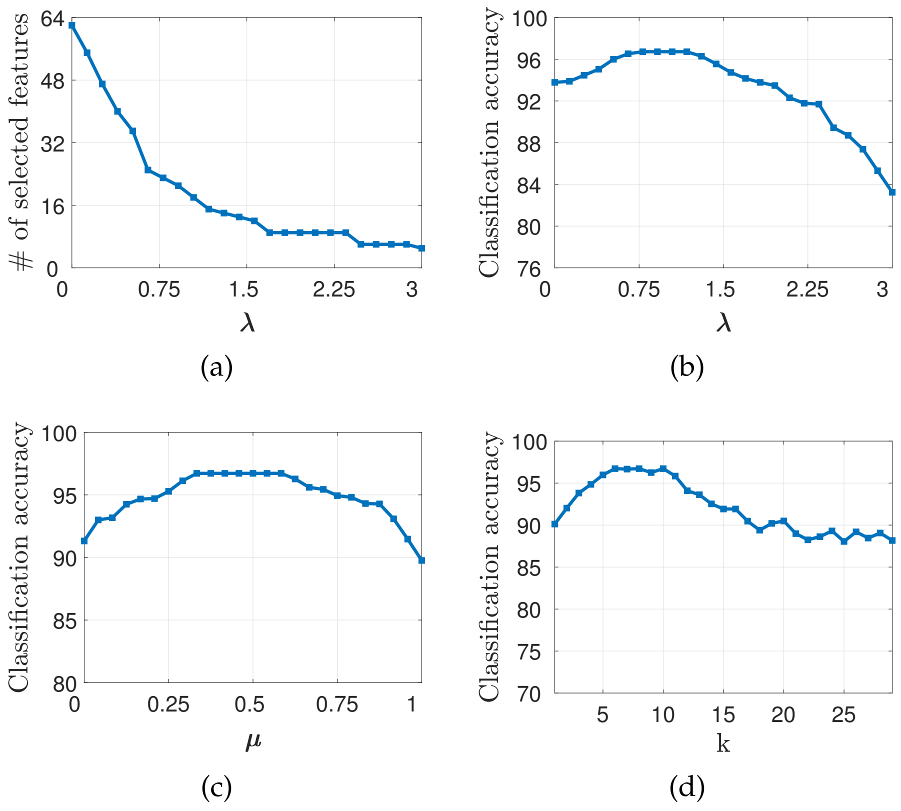


Figure 5.5: Effects of parameter changes on LMMK's performance for the CMU-9.

of λ would damage the discriminative property of the resulting RKHS. It is essential to indicate that the points $\lambda = 0$ in Figure 5.5-a and Figure 5.5-b are related to the performance of LMMK$_{\lambda=0}$, which is the LMMK's algorithm without having the sparsity term in Equation 5.7. Based on the figures, LMMK$_{\lambda=0}$ has an accuracy of 93.78% for the CMU dataset, which is comparable to the performances of DMKL and MKL-TR (as the best baselines in Table 5.1). This evidence proves my claim regarding the effectiveness of focusing on the classes' local discrimination in the feature space, even without the sparsity objective. Additionally, making a comparison between LMMK$_{\lambda=0}$ and sparse LMMK reveals the notable benefit of the l_1-norm sparsity term to both feature selection and classification accuracy.

Figure 5.5-c demonstrates the effect of the trade-off between the first two objective terms in Equation 5.7. For the Pascal dataset, balancing the pulling and pushing terms (with $0.35 \leq \mu \leq 0.6$) leads to the highest accuracy. Based on the experimental observations like the above, tuning μ around 0.5 generally results in a good performance.

According to the classification accuracy curves of Figure 5.5-d, the best choice for the value of k depends on the distribution of the classes; nevertheless, selecting large values for this parameter (e.g., $10 \leq k$) is expected to reduce the Acc dramatically. As an explanation, by increasing the size of neighborhoods (k), LMMK can no longer preserve its local property.

Evaluating Interpretable Multiple-Kernel Prototype Learning

As the second set of experiments, I implement the proposed IMKPL algorithm on the same selected datasets from the previous section. Therefore, the base kernels are computed as described in that section. Nevertheless, I evaluate its performance by making empirical comparisons to different baseline methods and also by employing additional performance measures to those from the previous section.

Parameters Tuning

I perform 5-fold cross-validation on the training set to tune the hyper-parameters $\{\lambda, \mu, T, \tau\}$ in Equation 5.11. I carry out a similar procedure regarding the parameter tuning of other baselines. For IMKPL , I determine the number of prototypes as $k = CT$ and the neighborhood radius $k = T$. As the rationale, the constraint $\|\vec{u}_i\|_0 \leq T$ and the term $\mathcal{J}_{dis}$ in Equation 5.11 make each $\vec{u}_i$ effective mostly on its T-radius neighborhood. In practice, choosing $\lambda = \mu = \tau \in [0.2\ 0.4]$ is a good working setting for IMKPL to initiate the parameter tuning (e.g., Figure 5.9).

Alternative Methods

I compare my proposed method to the following state-of-the-art prototype-based learning or multiple-kernel dictionary learning methods: KRSLVQ (Hofmann et al. 2014), PS (Bien, Tibshirani, et al. 2011), MKLDPL (X. Zhu et al. 2017), DKMLD (Thiagarajan, Ramamurthy, and Spanias 2014), and MIDL (Shrivastava, Pillai, and Patel 2015). The KRSLVQ algorithm is the sparse variant of the kernelized-robust LVQ (Hammer, Hofmann, et al. 2014), and for the PS algorithm, I use its distance-based implementation. These two algorithms are implemented on the average-kernel inputs ($\vec{\beta} = \vec{1}$). I also implement ISKPL as the single-kernel variant of IMKPL on that input representation. We can compare ISKPL to its multiple-kernel version to investigate the individual effect of its multiple-kernel part. Additionally, one can make a comparison between ISKPL and the sparse coding framework CKSC , which was proposed in Chapter 4. This comparison is of particular interest due to the general similarity of these two methods' structures.

It is important to emphasize that I exclusively select the baselines that can be evaluated according to my specific research objectives (**Ob1-Ob3**) in Section 5.3. For each method, I evaluate the quality of the learned prototypes on the resulting RKHS (based on $\{\mathbf{U}, \vec{\beta}\}$) by utilizing the following measures, which coincide with the objectives **Ob1-Ob3** in Section 5.3. Furthermore, all the experiments are done using the same CV scheme from the previous experiment section (for LMMK).

Interpretability of the Prototypes (IP) As discussed in Section 5.3, I have two main preferences regarding the interpretability of each prototype $\hat{\Phi}(\mathcal{X})\vec{u}_i$:

1. Its formation based on class-homogeneous data samples.
2. Its connection to local neighborhoods in the feature space.

Therefore I use the following IP term to evaluate the above criteria based on the values of the prototype vectors $\{\vec{u}_i\}_{i=1}^{k}$:

$$IP = 100 \times \frac{1}{k} \sum_{i=1}^{k} \frac{\vec{h}^q \vec{u}_i}{\|\mathbf{H}\vec{u}_i\|_1} exp(-\sum_{s,t} u_{si} u_{ti} \|\hat{\Phi}(\mathbf{X}_s) - \hat{\Phi}(\mathbf{X}_t)\|_2^2), \tag{5.36}$$

Table 5.2: Comparison of baselines regarding $IP(\%)$ and $DR(\%)$.

Methods	`UTKinect`		CMU-9		Schunk		HDM05		`CLL_SUB`		`TOX_171`	
	IP	DR	IP	DR	IP	DR	IP	DR	IP	DR	IP	DR
IMKPL	96	91	94	87	96	90	98	92	91	75	95	89
ISKPL	92	82	92	83	91	84	96	84	88	70	93	79
MKLDPL	78	60	71	64	77	70	76	72	75	57	82	66
DKMLD	74	52	66	61	74	64	70	64	67	51	71	60
MIDL	69	50	58	58	70	61	61	59	66	50	69	60
KRSLVQ	77	–	70	–	80	–	71	–	69	–	76	–
PS	79	–	74	–	83	–	82	–	78	–	80	–

in which $q = \arg\max_q \vec{h}^q \vec{u}_i$ is the class to which the i-th prototype is assigned. The first part of this equation obtains the maximum value of 1 if each $\vec{u}_i$ has its non-zero entries related to only one class of data, while the exponential term becomes 1 (maximum) if those entries correspond to a condensed neighborhood of points in RKHS. Hence, IP becomes close to 100% if both of the above concerns are sufficiently fulfilled. For the PS algorithm, I measure IP based on the samples inside the ϵ-radius of each prototype (Bien, Tibshirani, et al. 2011).

Discriminative Representation (*DR*) In order to properly evaluate how discriminative each prototype $\hat{\Phi}(\mathcal{X})\vec{u}_i$ is I define the discriminative representation term as

$$DR = 100 \times \frac{1}{k} \sum_{i=1}^{k} \frac{\sum_{s:\vec{h}_s=q} \gamma_{is}}{\|\vec{\gamma}^i\|_1}, \tag{5.37}$$

where q is the same as in IP measure, and $\mathbf{\Gamma}$ is computed based on the test set. Hence, DR becomes 100% (maximum) if each prototype i which is assigned to class q only represents (reconstructs) data from that class; i.e., the prototypes provide exclusive representation of their corresponding classes. Vector $\vec{\gamma}^i$ is the i-th row of $\mathbf{\Gamma}$, which shows the role of $\vec{u}_i$ in the encoding of all data samples. Based on the given definition in Equation 5.37, DR is also dependent on the quality of class-based interpretation of each dictionary atom. The DR measure does not fit the models of KRSLVQ and PS algorithms.

Classification Accuracy of Test Data (*Acc*) For each test data $\mathbf{X}_{test}$, I predict its class as $q = \arg\max_q \vec{h}^q \mathbf{U} \vec{\gamma}_{test}$, meaning that the q-th class provides the most contributions in the reconstruction of $\mathbf{X}_{test}$. The accuracy value Acc is defined similar to the evaluation of LMMK in the previous section.

Results: Efficiency of the Prototypes

In Table 5.2, I compare the baselines regarding the interpretability and discriminative qualities of their trained prototypes. Considering the IP values, IMKPL significantly outperforms both the MKDL and prototype-based learning algorithms. As the best result, for the Schunk dataset, my method has a margin of 19% compared to the best baseline algorithm (MKLDPL). Also, the ISKPL algorithm obtains higher interpretability performances than the single-kernel and multiple-kernel baselines, which shows the effectiveness of the prototype leaning parts of the design ($\mathcal{J}_{dis}$ and $\mathcal{J}_{ip}$). Besides, the difference between the IP values of ISKPL and IMKPL signifies the role of the $\mathcal{J}_{ls}$ objective in enhancing the interpretation of IMKPL 's prototypes by learning a suitable MK representation. Comparing the IP value of both IMKPL and ISKPL to CKSC from Chapter 4 (Figure 4.5) shows that ISKPL and its multiple-kernel version have more interpretable models in terms of their base elements. Specifically, the ISKPL framework focuses on the local representation of data and the formation of its prototypes by exemplars from condensed neighborhoods. Hence, its prototypes have better interpretation w.r.t. the class labels. Other algorithms show weak results in learning class-specific and locally concentrated prototypes.

We observe similar behaviors by comparing the algorithms based on the discriminative DR measure. Table 5.2 shows that the prototypes learned by IMKPL are more efficient regarding the exclusive representation of the classes on a combined RKHS. For instance, IMKPL outperforms MKLDPL (best baseline) with the DR margin of 31% on the UTKinect dataset. Furthermore, the ISKPL has a higher DR than other multiple-kernel methods (except IMKPL), which shows its prototypes are both interpretable and discriminative to a considerable extent.

Results: Accuracy and Feature Selection

Each base kernel $\mathcal{K}_i$ is derived from one dimension of the data. Therefore, I evaluate the feature selection performance of the algorithms by comparing $\|\vec{\beta}\|_0$ and Acc among them. As presented in Table 5.3, IMKPL has the best prediction accuracy for all datasets. It outperforms other baselines with relatively significant Acc-margins (e.g., 4.50% compared to MKLDPL on UTKinect). Particularity, comparing the Acc value of IMKPL to

Table 5.3: Comparison of IMKPL to selected baselines regarding Acc (%) and $\|\vec{\beta}\|_0$.

Methods	UTKinect		CMU-9		Schunk	
	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$
IMKPL	**98.82**	**14**	**94.58**	**22**	**95.21**	22
ISKPL	90.32	–	90.07	–	91.23	–
MKLDPL	94.32	32	93.87	34	93.46	32
DKMLD	91.64	30	92.11	27	92.58	**16**
MIDL	91.01	47	90.46	51	91.36	40
KRSLVQ	88.75	–	88.54	–	88.47	–
PS	85.89	–	86.38	–	84.52	–
	HDM05		CLL_SUB		TOX_171	
	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$
IMKPL	**96.37**	30	**81.73**	204	**97.21**	**72**
ISKPL	91.03	–	77.95	–	88.07	–
MKLDPL	94.95	40	79.63	310	94.72	347
DKMLD	92.74	**18**	78.25	**101**	90.49	130
MIDL	91.41	50	77.24	452	87.63	571
KRSLVQ	88.90	–	74.66	–	86.21	–
PS	84.64	–	74.03	–	82.47	–

The best result (**bold**) is according to a two-valued t-test at a 5% significance level.

ISKPL shows the effectiveness of the multiple-kernel formulation of IMKPL (role of $\vec{\beta}$ in Equation 5.11) in locally separating data classes in RKHS. For `TOX_171` , IMKPL has 9.16% classification accuracy than the ISKPL algorithm.

On the other hand, comparing the prediction accuracy of ISKPL to KRSLVQ and PS (as the major prototype-based learning methods) demonstrates the significant discriminative performance of my prototype-based algorithm even for single-kernel input. Even though ISKPL obtained lower *Acc* values than MKLDPL and DKMLD (as it does not optimize $\vec{\beta}$), its higher *DR* values show its design's effectiveness ($\mathcal{J}_{dis}$ and $\mathcal{J}_{ip}$) regarding our expectations from an interpretable prototype-based representation. The reason for the higher *DR* value of ISKPL is partially connected to the high *IP* value of its learned prototypes. Comparing the prediction accuracy of ISKPL to the proposed CKSC algorithm from Chapter 4 (Table 4.1) illustrates that CKSC obtains higher classification accuracy than ISKPL . This observation reveals that the CKSC model performs better with respect to encoding the supervised information related to motion classes. Although ISKPL has a more interpretable model, CKSC presents a more robust discriminative encoding, specifically relying on its consistent test and train model.

In addition, comparing the accuracy of IMKPL to LMMK (Table 5.1) shows that LMMK has a higher classification accuracy than IMKPL . This observation is due to the fact that the formulation of LMMK only aims for a better separation of data classes in the resulting RKHS. However, besides discriminative encoding of motion sequences, LMMK has other objectives in its model, which aims for the prototype-based encoding of motion sequences and their interpretability. Therefore, its specifically combined RKHS would sacrifice its discriminative performance in favor of these additional objectives.

Studying $\|\vec{\beta}\|_0$ in Table 5.3 demonstrates that IMKPL obtains the smallest set of selected features on three of the datasets (CMU Mocap , UTKinect , and `TOX_171`) compared to other multiple-kernel prototype-based baselines. It particularly shows a significant feature selection performance on `TOX_171` by obtaining 97.21% accuracy

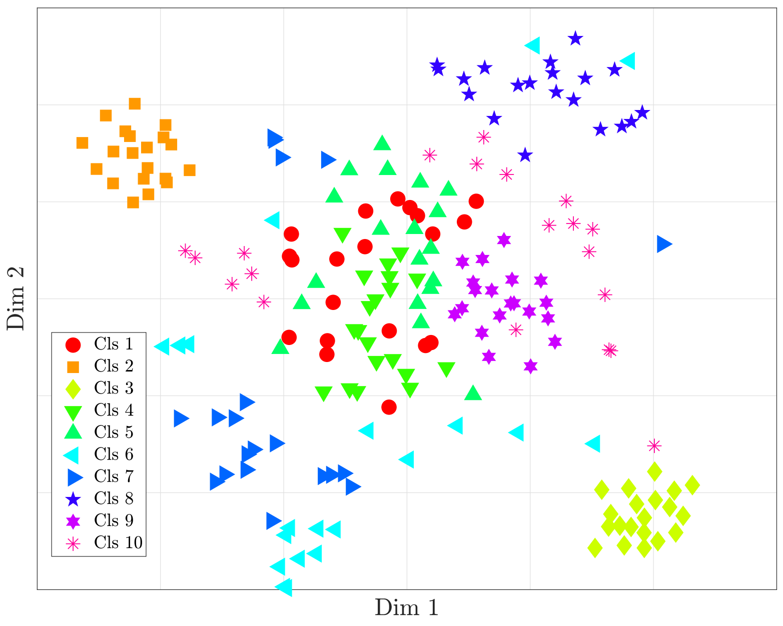


Figure 5.6: 2-dimensional embedding of the `UTKinect` dataset (based on the average-kernel) which visualizes the relative overlap of the classes (colored figure).

Table 5.4: Number of prototypes assigned to each class of the `UTKinect` dataset.

Classes	1	2	3	4	5	
Names	walk	sit down	stand up	pick up	carry	
# Prototypes	7	2	2	8	8	
Classes	6	7	8	9	10	All
Names	throw	push	pull	wave	clap	
# Prototypes	6	5	4	3	5	50

while selecting 72 features out of the total 5748 dimensions. Regarding other datasets (CLL_SUB_111 , HDM05 , and Schunk), also considering the *Acc* value next to $\|\vec{\beta}\|_0$ reveals that the multiple-kernel optimization of IMKPL (role of $\vec{\beta}$ in Equation 5.11) finds an efficient set of features that leads to a discriminative PB model with a high performance (but not necessarily the smallest feature set).

Comparing IMKPL to LMMK , the metric learning approach yields a more compact feature selection (smaller $\|\vec{\beta}\|_0$) on almost all datasets (except UTKinect). While the main objective of LMMK is to find a sparse set of features that increase the separation of data classes, the IMKPL model also aims for the reconstruction of data as well as the interpretable formation of the prototypes. Therefore, in order to fulfill these extra objectives, IMKPL needs to use more resources from the data (as input dimensions).

Detail Analysis of Prototypes

It is a pre-requisite feature for many prototype-based methods to fix the number of prototypes for each class of data through the training phase (e.g., MKLDPL, DKMLD, and KRSLVQ). However, as a common observation in real-world datasets, data classes

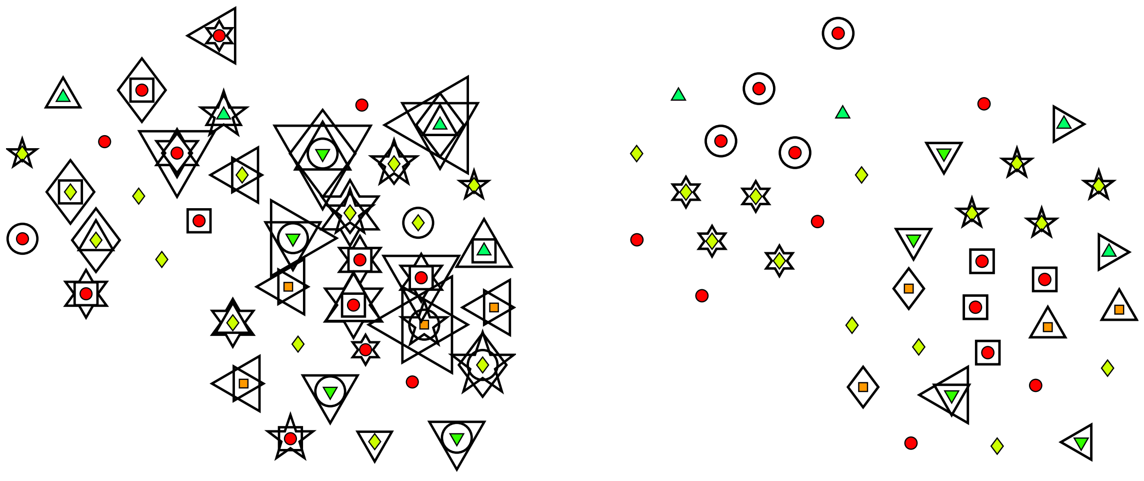

(a) The MKLDPL model. (b) The IMKPL model.

Figure 5.7: The contribution of training samples in the formation of dictionary atoms for (a) MKLPLD, (b) IMKPL on a dense neighborhood in the HDM05 dataset. Each small shape is a training sample related to one class of data. Each big shape type represents one dictionary atom $\Phi(\mathcal{X})\vec{u}_i$ and indicates the training samples on which it is built corresponding to the non-zero entries of $\vec{u}_i$.

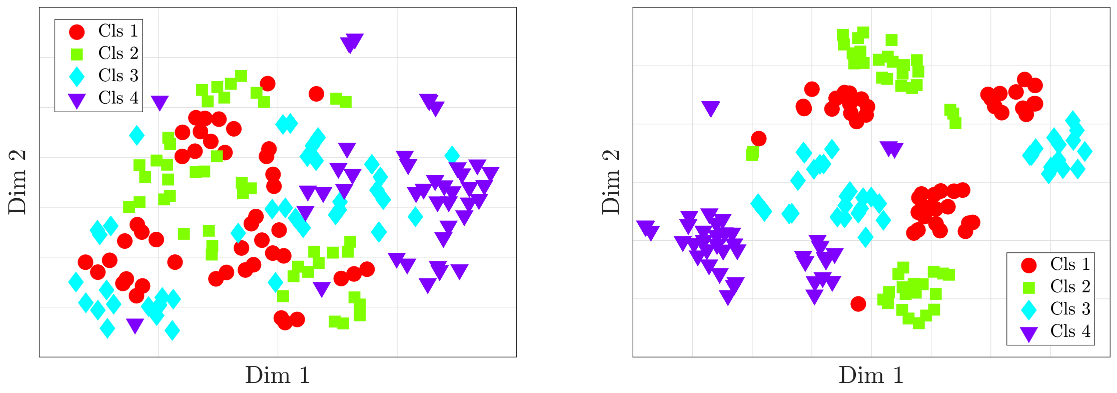


Figure 5.8: Visualization of overlapping classes for the `TOX_171` dataset based on the average-kernel combination (left) and the optimized $\vec{\beta}$-combined embedding (right). Clearly, the kernel weighting scheme has reduced the overlap between the classes.

are not distributed homogeneously. Even having the same number of data per class, their local distributions can be significantly diverse.

In our IMKPL model, although we decide in advance about the total number of prototypes to learn for each dataset as $k = CT$, IMKPL automatically assigns the proper number of prototypes to each class of data to fulfill the defined objectives **Ob1-Ob3** better. Table 5.4 represents the number of prototypes learned per motion class for the `UTKinect` dataset, which shows a notable variation among them. Also, by considering the 2D embedding of the `UTKinect` dataset in Figure 5.6 (using the t-SNE algorithm (Maaten and G. Hinton 2008)), it is clear that IMKPL assigns more prototypes to classes that suffer from significant overlap (e.g., *pick up* and *carry*) and fewer representatives to the more condensed classes (e.g., *sit down* and *stand up*).

Accordingly, Figure 5.7 visualizes the formation of dictionary atoms based on non-zero entries of columns of **U** for the HDM05 dataset. The relatively small entries of each $\vec{u}_i$ are zeroed, such that the remaining coefficients point toward significant training samples for the given dictionary atom. As we can observe for the MKLDPL model (Figure 5.7-a), its dictionary vectors have considerable overlap regarding their contributing classes. The cross-class contributions for the formation of each $\vec{u}_i$ is much higher compared to the IMKPL model (Figure 5.7-b), resulting in the IP value of 78. On average, each $\vec{u}_i$ from MKLDPL is connected to data samples from 3 to 4 different classes, and the majority of the sequences are used to construct the prototype vectors in **U**. In contrast, the prototypes of the IMKPL model are mostly constructed from one class of data, resulting in an IP value of 98%. Comparing IMKPL to CKSC ($IP = 94\%$) from Chapter 4 (Figure 4.6-c), IMKPL results in fewer overlapping prototypes in terms of the data sequence they use. This formation is dues to the dense data neighborhoods from which prototypes are constructed.

Visualization of the Learned Kernel

To visualize the effect of the learned kernel weights ($\vec{\beta}$) on the distribution of classes, we visualized the 2-dimensional embeddings of the `TOX_171` dataset in Figure 5.8 (using the t-SNE method). Clearly, the optimized $\vec{\beta}$ has lead to better local separation of the classes in the resulting RKHS (Figure 5.8-left) compared to the average-kernel representation of

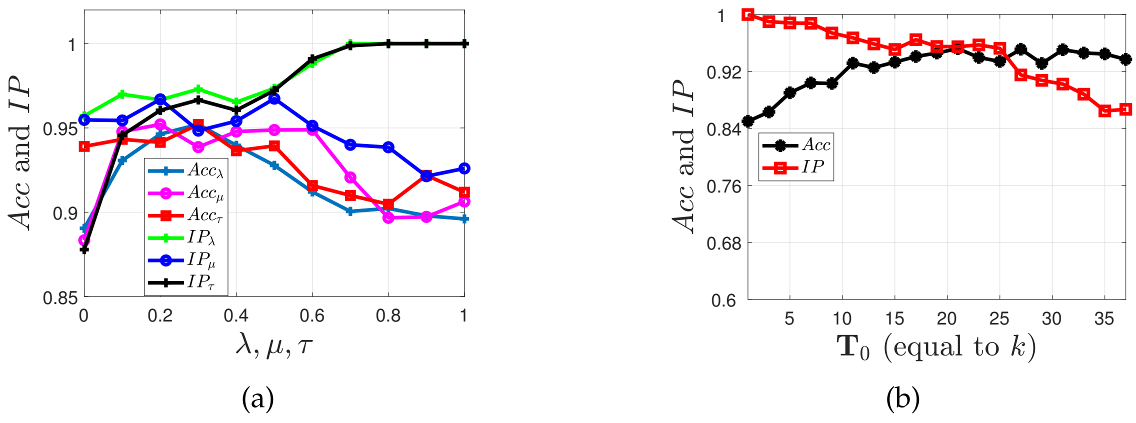


Figure 5.9: The isolated effect of changing the parameters $\{\lambda, \mu, \tau\}$ (a) and T (b) on the performance measures Acc and IP for the Schunk dataset.

the data ($\vec{\beta} = \vec{1}/d$) in Figure 5.8-right. This observation complies with the role of $\mathcal{J}_{ls}$ in Equation 5.11.

Effect of Parameter Settings

We study the effect of parameters $\{\lambda, \mu, \tau, T\}$ on the Acc and Ip performance of IMKPL by conducting four individual experiments on the `Isolet` dataset. Each time, we change one parameter while fixing others by their values related to results in Table 5.3.

As illustrated by Figure 5.9-(left), the performance is acceptable when $\lambda, \mu, \tau \in [0.1\ 0.5]$, but Acc and IP may decrease outside of this range. Specifically, τ has a slight effect on Acc, but it increases the value of IP almost monotonically. In comparison, μ and λ influence Acc more significantly. Nevertheless, they have small effects on IP when they are small (in $[0\ 0.6]$), but λ has a productive and μ a slight destructive effect for their larger values. When the data classes have large overlap in the RKHS, focusing only on $\mathcal{J}_{ls}$ (large μ) does not necessarily provide the best prototype-based solution.

Figure 5.9-(right) shows that increasing T generally improves Acc up to an upper limit. Since $k = CT$, large values of T leads to learning redundant prototypes. Besides, increasing T generally degrades the IP value, but it almost reaches a lower bound value for large T ($\approx 87\%$ for Schunk) because of the minimum interpretability induced by the non-negativity constraint $u_{ji} \in \mathbb{R}_{\geq 0}$ in Equation 5.11.

Running Time and Convergence Curve

To evaluate the computational complexity of IMKPL , we compare the training running time of the selected methods on `CLL_SUB`, UTKinect , and CMU datasets. As reported in Table 5.5, IMKPL has a smaller computational time than other MK algorithms (MKLDPL,

Table 5.5: Training run-time of baseline algorithms (seconds).

Dataset	**IMKPL** **(proposed)**	MKLDPL	DKMLD	MIDL	KRSLVQ	PS
`CLL_SUB`	2.58e2	2.85e4	4.08e4	8.76e4	1.24e2	2.32e0
UTKinect	8.09e0	8.83e2	1.67e3	3.57e3	1.47e1	7.36e-2
CMU-9	1.59e0	8.31e1	1.32e2	3.25e2	1.34e0	1.49e-2

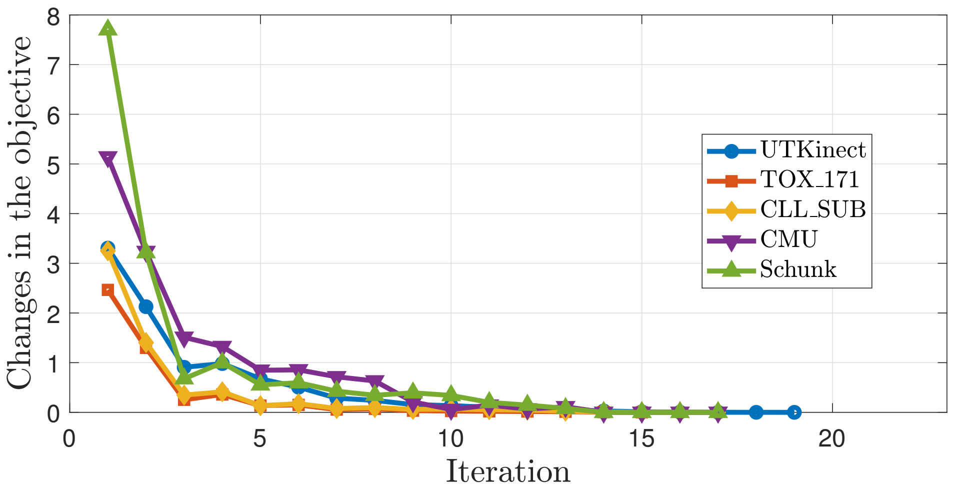


Figure 5.10: The convergence curves of IMKPL on the selected datasets.

Table 5.6: Average of DRA measure (%) for the reconstruction of the unseen classes.

	Cricket	CMU-9	Words	Squat
DRA (%)	76.4	84.5	80.2	62.6

DKMLD, and MIDL) and is even faster than or comparable to KRSLVQ (as a single-kernel method) when the number of features d is small in relation to N (UTKinect and CMU). Although the PS algorithm has a shorter running time than IMKPL , it is not applicable to the multiple-kernel data.

In Figure 5.10, I plot the changes in the value of the whole objective function of Equation 5.11 during the training iterations. Based on this figure, Algorithm 5.1 is considered converged when the above value becomes relatively small, which occurs rapidly on all the selected datasets in the experiments (less than 20 iterations).

Evaluating Multiple-Kernel Dictionary Structure

To evaluate the performance of my MKD-SC framework for representation and discrimination of unseen data, I choose the MTS datasets CMU Mocap , Cricket , Words , and Squat with the descriptions provided in Section 2.4. For all the datasets, the dimension-specific kernels are computed as in Section 5.5.

For tuning T and the dictionary size in Equation 5.23, I use 5-fold cross-validation.

Partial Reconstruction Results

In order to evaluate the reconstruction quality for each unseen data $\mathbf{Z}$, I define the dimension-reconstruction accuracy measure as

$$DRA := \frac{|\mathcal{S}|}{d} \qquad \mathcal{S} \text{ from Equation 5.26 for } \epsilon = 0.1.$$

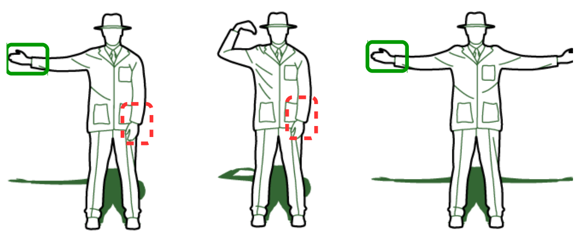
Furthermore, each reconstructed dimension of $\mathbf{Z}$ that satisfies the above threshold is interpreted via the class of data with the most contribution as in Section 5.4. Table 5.6

reports the DRA values for the selected MTS datasets, where the CMU and Words datasets have higher DRA values due to their diverse set of training classes, which increases the dimension-level similarity between seen and unseen classes. As an example, I illustrate the dimension-level reconstruction of two unseen categories from the Cricket dataset in Figure 5.11, In that experiment, the *No ball* class is fully reconstructed via its relation to the movement of the left hand in the *Short* class and to that of the right hand in the *Wide* class.

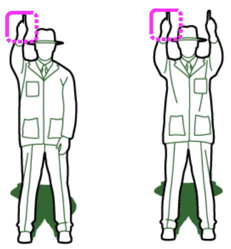
Incremental Clustering Results

To evaluate the incremental clustering of Section 5.4, I use the average clustering error (CE) and normalized mutual information (NMI) (Wencheng Zhu, J. Lu, and J. Zhou 2018). To that aim, I cut each dendrogram from where it has an equal number of clusters to the ground truth. Therefore, the average CE is calculated over 10 clustering repetitions for each algorithm. The NMI measures the amount of information shared between the clustering and the ground-truth, which lies in the range of $[0, 1]$, while the ideal score of 1 means totally independent clusters. As the most relevant baseline, I choose the self-learning algorithm (D. Lu, J. Guo, and X. Zhou 2016) without its novelty detection part. Besides, I implement the spectral clustering algorithm (SC) on the original kernel matrix $\mathcal{K}(\mathbf{Z}, \mathcal{X})$ to compare my framework to the regular clustering of $\mathcal{Z}$. As another baseline, I also use the NNKSC algorithm (Section 4.2) as the single-kernel predecessor of MKD-SC, for which the $\mathbf{R}$ matrix becomes an N-dimensional vector.

According to the clustering results in Table 5.7, the proposed MKD-SC method provides encodings that lead to better clustering of the unseen data compared to the baselines. The superiority of the spectral-clustering over NNKSC and self-learning methods (e.g.,

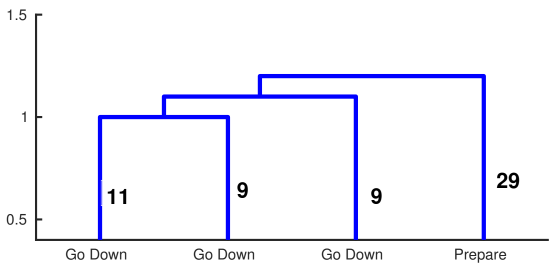


(a) *no ball* $\rightarrow \{$*short* $+$ *wide*$\}$

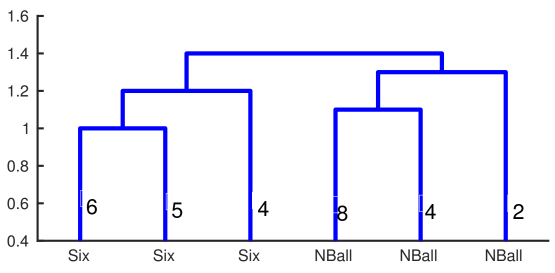


(b) *out* $\rightarrow \{$*six*$\}$

Figure 5.11: Dimension-level interpretation of *no-ball* and *out* (Cricket) based on the training classes. Related dimensions are specified using same-color rectangles.



(a) Squat dataset



(b) Cricket dataset

Figure 5.12: Incremental clustering dendrograms for unseen classes of Squat **(a)** and Cricket **(b)**.

Table 5.7: Clustering error (CE) (%) and NMI for the unseen categories.

Methods	Words		Squat		CMU-9		Cricket	
	CE	NMI	CE	NMI	CE	NMI	CE	NMI
MKD-SC(Proposed)	12.31	0.89	0	1	9.28	0.92	0	1
Self-learning	18.75	0.84	0	1	14.25	0.87	16.63	0.85
NNKSC	21.61	0.78	15.74	0.88	18.88	0.85	12.45	0.87
SC	27.51	0.76	13.04	0.90	23.45	0.76	8.04	0.89

for Cricket dataset) depends on the discriminative quality of the original kernels. The self-learning method can have a better performance than NNKSC and spectral-clustering when its descriptor-based features can better discriminate between the different categories of the unseen classes.

Based on clustering dendrograms in Fig. 5.12, the unseen Squat and Cricket classes are well categorized, which shows the effectiveness of the learned attributes for the distinct representation of the unknown classes. The incremental clustering also categorized these unseen classes into a few sub-clusters. For Squat (Fig. 5.12-a), the 3 sub-clusters for the *Go-down* class are related to different performance styles of the dataset's 3 participants regarding this specific phase of the squat. Similarly, for each unseen category of the Cricket dataset (Fig. 5.12-b), there are sub-clusters recognized for each of the distinct main clusters, which reveal the existing structured variation within each of these classes.

5.6 conclusion

In this chapter, I proposed three multiple-kernel learning frameworks which focus on transferring data to a combined RKHS in favor of their specific supervised or unsupervised objectives. The new RKHS is formed as a linear combination of individual kernels that correspond to the input motion sequence's individual dimensions. From another perspective, each framework provides a specific feature selection according to its defined goal.

My proposed LMMK algorithm performs discriminative multiple-kernel learning for multi-class classification problems. This algorithm focuses on increasing the local separation of the classes in the feature space, which improves the *k*NN classifier's performance classifier for the motion sequences. To that aim, I applied metric learning to the feature space by defining a diagonal multiple-kernel metric in the RKHS. Furthermore, I employed an l_1-norm sparsity term in the formulation of LMMK to find a sparse weighted combination of the base kernels. This sparse set of selected kernels can be interpreted as semantically relevant dimensions of the input motion sequence to the given supervised task. I implemented my algorithm on real-world mocap benchmarks (as instances of multi-class multidimensional time-series), which shows that LMMK outperforms other multiple-kernel learning algorithms in terms of discriminative feature selection. Based on my empirical evaluations, the LMMK method is comparable to the DTW-LMNN algorithm of Chapter 3 regarding classification accuracy, but it outperforms DTW-LMNN with a more effective feature selection.

As another multiple-kernel learning framework, my IMKPL algorithm focuses on the interpretable prototype-based representation of motion data in the feature space. This framework is constructed upon a multiple-kernel dictionary learning formulation. This algorithm learns semantically interpretable prototypes as the local representatives of motion classes in RKHS (e.g., a subset of similar *walking* sequences) while effectively discriminating the classes from each other in that space. To that aim, the IMKPL method performs an efficient feature selection for motion data, which is beneficial to the defined prototype-based representation. Empirical evaluations on both vectorial and motion domains validate the superiority of IMKPL over other prototype-based baselines regarding the interpretability and discriminative power of its specific model. The implementations showed that IMKPL cannot outperform the LMMK method in terms of discriminative feature selection. Nevertheless, its highly interpretable prototype-based model is particularly beneficial to practitioners and domain experts.

As the last proposed algorithm in this chapter, I proposed an unsupervised multiple-kernel framework, which provides an interpretable analysis of unseen classes in a motion dataset. My MKD-SC algorithm is constructed based on a novel multiple-kernel dictionary structure, which uses the multiple-kernel representations of motion dimensions to learn semantic attributes. Based on these attributes, my unsupervised MKD-SC framework reconstructs the unseen classes (partially or entirely) in the feature space according to the relation of their dimensions to those of the seen categories. Such particular encoding provides an interpretable description for the observed novel motion types. Benefiting from the obtained sparse encoding, I proposed an online clustering, which incrementally categorizes novel motions into distinct clusters upon their observation. Experiments on real mocap benchmarks show the effectiveness of my MKD-SC framework in obtaining interpretable descriptions for unseen MTS classes. Additionally, the designed incremental clustering algorithm outperforms other baselines in terms of clustering accuracy, when the baselines are directly applied to the input kernels.

In the chapters up to here, I proposed motion data analysis models for which the input data consists of already segmented motion sequences. In addition, those models treat each sequence as a pack without focusing on individual regions of the input time-series in the temporal axis. Even by referring to DTW as a method that analyze the temporal content of the input, its analysis is not actively affected by the next-level supervised or unsupervised task. Regarding this concern, I design a novel deep learning framework in the next chapter, which directly analyzes the temporal content of motion sequences according to the given supervised task. Specifically, it finds interpretable discriminative patterns in long sequences of motions. Such temporal patterns are beneficial to both activity recognition and temporal segmentation problems in motion data and improve the interpretability of the network.

6 INTERPRETABLE MOTION ANALYSIS WITH CONVOLUTIONAL NEURAL NETWORK

Publications: This chapter is partially based on the following publications.

- Hosseini, Babak, Romain Montagne, and Barbara Hammer (2019). "Deep-Aligned Convolutional Neural Network for Skeleton-based Action Recognition and Segmentation". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE.
- — (2020). "Deep-Aligned Convolutional Neural Network for Skeleton-Based Action Recognition and Segmentation (extended article)". In: *Data Science and Engineering*. ISSN: 2364-1541. URL: https://doi.org/10.1007/s41019-020-00123-3.

Skeleton-based action recognition is a specific domain of motion analysis in which the mocap data describes the movements of skeleton-joint, and its purpose is to classify the actions represented by the movement sequences (Jake K Aggarwal and Ryoo 2011). In recent years, skeleton-based action recognition has become an interesting problem for many deep learning algorithms such as convolutional neural networks (CNNs) (Y. Du, Fu, and Liang Wang 2015; Ke et al. 2017; Sijie Yan, Xiong, and D. Lin 2018) and recurrent neural networks (RNN) (Y. Du, Wei Wang, and Liang Wang 2015; S. Song et al. 2017; J. Liu, Shahroudy, et al. 2018). RNN methods can learn the temporal dynamics of the sequential data; nevertheless, they have practical shortcomings in training their stacked structures (M. Liu, Hong Liu, and Chen Chen 2017; H. Wang and Liang Wang 2017). Compared to RNN architectures, CNN-based methods provide more effective solutions by extracting local features from their input and finding discriminative patterns in the data (Gehring et al. 2017; Ke et al. 2017).

Regardless of CNN's promising feature extraction capability, its specific convolutional structure is designed originally for image-based input data and primarily relies on spatial dependencies between the neighboring points. In contrast, such a direct relationship does not generally exist in skeleton-based action datasets. Although some works tried to solve this problem by using 1-dimensional filters (only for the temporal dimension), it is still not an efficient solution to this specific shortcoming of CNN-based frameworks (Y. Zheng et al. 2014). Therefore, as a common strategy, CNN models are combined with other methods such as long short term memory (LSTM), reinforcement learning (RF), and graph-based model as the preprocessing or post-processing step of the deep architecture (Núñez et al. 2018; Y. Tang et al. 2018; Sijie Yan, Xiong, and D. Lin 2018).

Despite the notable performance of deep neural networks in classification tasks, their model interpretation is always of particular interest for practitioners and domain experts (Patterson and Gibson 2017). Accordingly, several methods are proposed that particularly focus on the interpretation of CNN models. Some techniques modify the network architecture to add such characteristics to it (Kuo et al. 2019; Quanshi Zhang, Nian Wu, and S.-C. Zhu 2018; Bolei Zhou et al. 2016; S. Shen et al. 2019), while other methods focus on interpreting the decision-making process of the network (A. Nguyen, Yosinski, and Clune 2016; Montavon, Lapuschkin, et al. 2017; Montavon, Samek, and K.-R. Müller 2018; Kuo 2016; Fong and Vedaldi 2017). Regardless of the improvements in this area, the majority of these techniques are only applicable to the standard form of a CNN model (LeCun et al. 1989), which generally have a weak classification accuracy on motion data. Furthermore, the mentioned high-performance combination of CNN with other action recognition methods prevents the whole architecture from becoming interpretable and makes it unsuitable for applying the mentioned interpretation techniques.

Regarding the classification of temporal data, it is shown that via comparing each data sequence to some predefined or learned sequences, we can classify the data samples with high accuracy (Anagnostopoulos et al. 2006; Rakthanmanon and E. J. Keogh 2013). Such methods rely on the semantic similarity and temporal alignment of time-series (such as DTW) (Petitjean, Forestier, Geoffrey I Webb, et al. 2016). In algorithms similar to (L. Ye and E. Keogh 2009; C. Ji et al. 2019), finding a small distinct subsequence in the input data (called shapelet) can reveal its classification label. These subsequences are constructed of short time-series which present semantic similarity to specific parts of longer sequences. In the context of motion analysis, one can consider these short sequences as temporal prototypes, which carry meaningful information about the given data or the defined task (Yeh 2018). For instance, to distinguish the *walking* class from other types of motions, a few leg movement frames might be enough to declare a motion sequence as *walking*. Therefore, to benefit from the highly interpretable property of such temporal prototypes, I like to address the follow-up research question of **RQ4**:

RQ4-a: How can we incorporate the above concept of temporal prototype-aliment into a CNN architecture to make it more interpretable for motion data classification?

Working on motion data analysis, a crucial step before applying many algorithms on such sequential data is the temporal segmentation of the motions. In that initial step, we need to split the long stream of recorded data into meaningful non-overlapping actions in the time axis (F. Zhou, De la Torre, and Hodgins 2013). The semantic notion of the action segments depends on the application and the defined overarching task. However, the manual segmentation of such data is considerably time-consuming, especially due to the ever-increasing growth in the size of such datasets. Accordingly, several unsupervised algorithms are proposed for temporal segmentation of motion data, which does not use any prior knowledge about its constituent actions (B. Krüger et al. 2017; F. Zhou, De la Torre, and Hodgins 2013; S. Li, K. Li, and Fu 2015; Tierney, J. Gao, and Yi Guo 2014). These methods mostly rely on the self-similarity or temporal clustering of time-frames in the given motion stream. However, due to their unsupervised nature, they are prone to over-segmentation of actions into smaller sub-sequences that do not coincide with the actions' semantic priors.

On the other hand, no major supervised segmentation method has been proposed yet for general skeleton-based data. In that context, (Lichen Wang, Z. Ding, and Fu 2018) and (T. Zhou et al. 2020) proposed supervised frameworks, which extend the temporal clustering of motion frames to a transfer learning problem, which benefit from supervised information. However, such methods still need to solve their optimization problem per test sequence and required prior knowledge about the test data. In some cases, supervised domain-specific segmentation methods are proposed, which benefit from deep architecture (Escalera et al. 2014; J. Y. Chang 2014; Neverova et al. 2016). Nevertheless, these networks have skeleton-specific architectures according to a particular segmentation problem and are not applicable to general skeleton-based mocap data.

Extending the segmentation problem to the temporal classification of motion sequences aims to segment and predict the action to which each time-frame belongs.

Temporal classification is a popular concept in other sequential data domains such as speech recognition and text analysis, where it is known as the sequence labeling of the input data stream (Gehring et al. 2017). Several deep learning models are proposed for such application based on CNN, RNN, or LSTM networks (Lample et al. 2016; X. Ma and Hovy 2016; Z. Yang, Salakhutdinov, and Cohen 2016; Alzaidy, Caragea, and Giles 2019; Tsai et al. 2019). The majority of these techniques rely on employing a Conditional Random Field (CRF) module (Lafferty, McCallum, and Pereira 2001) in their architecture, which considers dependencies between the time-frame predictions. Although these methods are effective regarding classification accuracy and computational complexity, their implementation on skeleton-based motion data requires domain-specific data preprocessing. Moreover, those methods that rely on specific world embeddings require substantial changes in the network's structure to make them applicable to motion sequences. According to the notable performance of deep neural networks in segmentation and classification of temporal data, my next follow-up research question of **RQ4** is:

RQ4-b: How can we design a deep neural network which can effectively perform temporal classification specifically for motion data?

Regarding the above research questions, I propose the deep-aligned convolutional neural network (DACNN) as a novel deep neural architecture for skeleton-based action recognition and segmentation (SBARS). This network has a CNN model in its core design while introducing a new type of interpretable filter in its primary layer inspired by the time-series alignment concept. Compared to the state-of-the-art deep neural architecture for SBARS problems, DACNN has a more interpretable structure and is also flexible in terms of the input size and the complexity of the given problem. To be more specific, I have the following contributions with respect to the state-of-the-art in the temporal classification of motion data:

- I introduce the alignment kernels (Al-filters) in the context of CNN, which are more efficient than the convolutional filters regarding the temporal feature extraction and classification of skeleton-based action data.
- DACNN learns temporal sub-sequences in the data as essential local patterns, making the network's decision-making process more interpretable and leading to more accurate predictions.
- As another crucial contribution to the state-of-the-art, my DACNN architecture can incrementally extend its depth (number of middle layers) based on the quality and length of the learned Al-filters during the learning process and without disrupting the training phase.

In the next section, I summarize the most relevant work in segmentation and temporal classification literature. Then, I introduce the alignment filters based on which I propose the novel architecture of the DACNN model. The proposed architecture is empirically evaluated on mocap benchmarks, and the chapter is concluded afterward.

6.1 state of the art

Generally, it is possible to split the skeleton-based action recognition methods into two general categories:

The first group includes methods with a preprocessing step to extract features (usually hand-coded) that best represent the skeleton information. For instance, in (Jiang Wang et al. 2012), the local occupancy pattern was proposed based on the joints' depth appearance and an ensemble action recognition model. In (Hussein et al. 2013), they proposed a discriminator based on the covariance matrix of joints locations, while in (Vemulapalli, Arrate, and Chellappa 2014), the algorithm was designed based on the 3D geometric relationships between different regions of the body. Methods similar to (Si et al. 2018) are constructed upon the spatial processing of individual groups of motion dimensions related to particular human parts (such as hands, legs, and shoulders.)

The second category benefits from the general strength of deep neural networks in performing enriched feature extraction. These methods are generally designed based on CNN and RNN models with architectural modifications or in combination with other techniques. Among RNN frameworks, a regularized LSTM architecture is proposed in (Wentao Zhu et al. 2016) for co-occurrence feature extraction. A spatiotemporal attention-based model is utilized in (S. Song et al. 2017) to assign different weights to different frames, and a trust-gate technique was proposed in (J. Liu, Shahroudy, et al. 2018) to deal with the noise in skeleton-based data.

Regarding the approaches based on CNN models, Tang *et al.* (Y. Tang et al. 2018) combined CNN with a reinforcement learning module to learn the most efficient video frames. In (Ke et al. 2017), cylindrical coordinates were utilized to present a new skeleton representation. The skeleton data was transformed into images in (M. Liu, Hong Liu, and Chen Chen 2017) to be more appropriate for CNN architecture, while in (H. Wang and Liang Wang 2017) two CNN models were trained individually based on the joint position and velocity information to perform skeleton-based action recognition.

A standard convolutional neural network has the same structure as a feedforward neural network that replaces its matrix multiplications with convolutional operators (LeCun et al. 1989). A CNN architecture consists of several convolutional layers (conv. layers), one or few fully connected layers (FC layers), and a final output layer. The network takes in the raw input and gives an output vector. Similar to feed-forward networks, the output vector has the same size as the number of classes in the data distribution. Each element of this vector shows the likelihood for one class of data in the given classification task. The original CNN architecture takes 2D input sizes. However, several works such as (Y. Zheng et al. 2014; Lea et al. 2016; L. Sun et al. 2015) showed that CNN models with employed 1D architectures could lead to more efficient processing of multivariate time-series (such as motion sequences).

A 1D architecture works upon the 1D convolution operator between an input $\vec{x} \in \mathbb{R}^{1\times n}$ and a filter $\vec{w} \in \mathbb{R}^{1\times k}$, resulting in an output vector $\vec{o}$ such as :

$$s(t) = (\vec{w} * \vec{x})(t) = \sum_{i=-\frac{k}{2}}^{\frac{k}{2}} w(i) \cdot x(t+i), \tag{6.1}$$

where the $*$ denotes the convolution operator. Based on the above formulation, the lth conv. layer of the network has the filter with parameter tensor $\mathcal{W}_l \in \mathbb{R}^{d_{l-1}\times d_l \times k}$, which means the l-th layer has d_l set of $d_{(l-1)}$-channel filters of length k. Assuming the input to later l is a feature map $\mathbf{O}_{(l-1)} \in \mathbb{R}^{d_{l-1}\times T_{(l-1)}}$, the output of layer l is computed as

$\mathbf{O}_{(l)} = \sigma_l(\mathbf{S}_l)$, where σ is the activation function for layer l, and matrix $\mathbf{S}_l$ is computed as

$$s_l(j,t) = \sum_{i=1}^{d} [w_l(i,j,:) * o_{(l-1)}(i,:)](t) + b_l(j). \tag{6.2}$$

In Equation 6.2, $\vec{b}_l$ is the bias vector for layer l, similar to the bias concept in a multilayer perceptron (MLP) layer (Rosenblatt 1957).

The typical activation operator for conv. layers is the ReLU operator, which only passes the positive values to its output, seeking relevant patterns in each feature map. Some works suggested improved replacements to ReLU such as leaky ReLU (B. Xu et al. 2015), parametric ReLU (K. He et al. 2015), and ELU (Clevert, Unterthiner, and Hochreiter 2015), which try to mitigate the dying ReLU issue (always having negative inputs) (Connie et al. 2017).

As a typical abstraction operation in CNN models, the time axis of $\mathbf{O}_{(l)}$ is scanned by a max-pooling operator similar to the convolution in Equation 6.1, except that for each time-frame t, its output is the maximum of $x((t-1)m+i+1)$ for $i = -\frac{p}{2}, \ldots, \frac{p}{2}$. Such operation results in $\mathbf{O}_{(l)} \in \mathbb{R}^{d_l \times T_{(l)}}$ such that $T_{(l)} = \frac{T_{(l-1)}}{m}$, where (m, p) are the stride and kernel size of the pooling operator, respectively. As illustrated in Figure 6.1, the input sequence $\mathbf{X} \in \mathbb{R}^{d \times T}$ is scanned by several conv. layers in sequential order, through which the time-length of $\mathbf{X}$ is divided by the pooling operators while its number of channels (depth) is increased according to the filter's channels. Therefore, after passing the signal through q consecutive conv. layers, the resulting feature map is $\mathbf{O}_{(q)} \in \mathbb{R}^{d_q \times \frac{T}{p^q}}$. This process extracts the relevant information from $\mathbf{X}$.

After concatenating the elements of $\mathbf{O}_{(q)}$ (flattening), they are fed to a fully connected (FC) layer (or a sequence of them). For each FC layer with d_{out} neurons and input sized of d_{in}, resulting in the weight matrix $\mathbf{W} \in \mathbb{R}^{d_{out} \times d_{in}}$, its output is computed as

$$\mathbf{O}_{(out)} = \sigma(\mathbf{W} \times \mathbf{O}_{(in)} + \vec{b}), \tag{6.3}$$

where $\vec{b}$ is the bias vector, and $\sigma(x)$ is the activation function, typically the sigmoid function $\frac{1}{1+e^{-x}}$ or the ReLU operator. The last layer of the network results in a score vector $\vec{y} \in \mathbb{R}^C$, each entry of which shows the likelihood of the corresponding class for

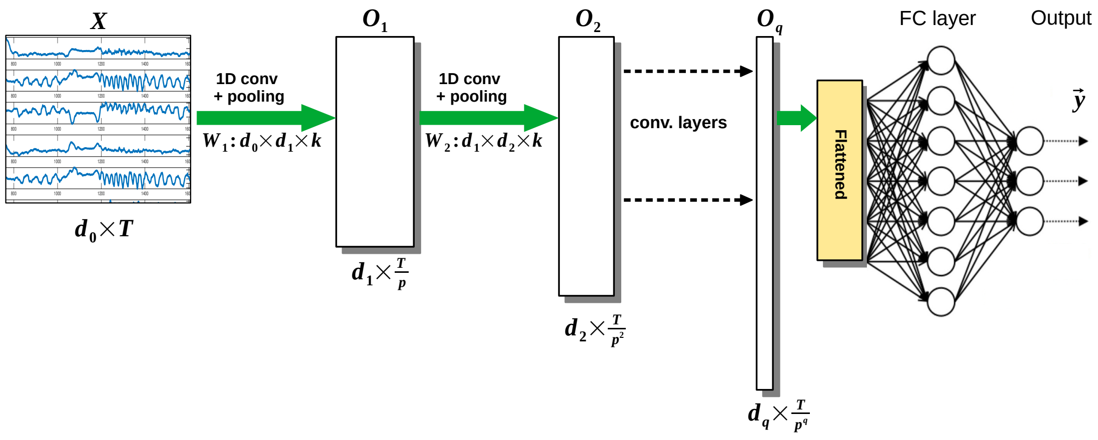


Figure 6.1: The typical structure of a 1D convolutional neural network with multivariate sequential input $\mathbf{X}$ and a likelihood output vector $\vec{y}$.

the given input sequence $\mathbf{X}$. Given the label of $\mathbf{X}$ as a one-hot vector $\vec{h}$, the network loss can be defined as a cross-entropy cost function or its variant:

$$LOSS(\mathbf{X}) = -\sum_{i=1}^{C} h_i log(y_i) \tag{6.4}$$

The training of a CNN network follows the same backpropagation principle of feedforward neural networks (Rumelhart, G. E. Hinton, and Williams 1986), in which the loss term of Equation 6.4 is minimized for all data points by finding optimal network parameters. The general optimization algorithm for neural networks is the stochastic gradient descent (SGD) (Robbins and Monro 1951), or its variants, which updates each network parameter w as

$$w_{i+1} = w_i - \lambda_i \nabla LOSS(w_i), \tag{6.5}$$

where i and λ_i denote the optimization iteration and its update step, respectively. The gradient $\nabla LOSS(w_i)$ is calculated based on the partial derivative of the loss term with respect to w and is averaged over a batch of data samples (Krizhevsky, Sutskever, and G. E. Hinton 2017). Several enhancements are already proposed for more efficient training of a CNN architecture, such as batch normalization of each layer's input (Ioffe and Szegedy 2015) to reduce its overfitting and speed up the training or using dropouts for FC layers to improve the generalization capability of the convolutional network (N. Srivastava et al. 2014).

Despite the high-performance of well-known deep CNN models such as VGG, ResNet, Inception, and Xception (Simonyan and Zisserman 2014; K. He et al. 2016; Szegedy et al. 2015; Chollet 2017) in extracting enriched features from input data, their architectures take fixed-size inputs and result in vectorial outputs. In some applications, input data is collected in a laboratory setting; and hence, all images can easily be re-scaled into a fixed size. Face recognition (Georghiades, Belhumeur, and Kriegman 2001; Gross et al. 2010) or gesture classification (Ren, J. Yuan, and Zhengyou Zhang 2011; Ohn-Bar and Trivedi 2014) datasets are the typical examples of such data. Nevertheless, such a workaround is not generally effective against real-world images or segmentation problems where multiple classes may exist in a single image (J. Long, Shelhamer, and Darrell 2015). A commonly used solution is to crop the image into several small sub-images and train or recall the network based on them (Girshick 2015).

On the other hand, the concept of using flexible input sizes in convolutional networks was first introduced in (Matan et al. 1992; Wolf and Platt 1994) for image classification. These methods replace the final fully connected layer of the CNN with another conv layer, which produces a score output that is a down-scaled feature map of the input image. Hence, the output's spatial size would change proportional to the input's size. Several works like (F. Ning et al. 2005; Pinheiro and Collobert 2014; Sermanet et al. 2013) used fully convolutional inference in their network for image segmentation and restoration problems. Specifically, the proposed fully convolutional network (FCN) in (J. Long, Shelhamer, and Darrell 2015; Tompson et al. 2014) is trained end-to-end. The FCN in (Tompson et al. 2014) is designed for a binary pose estimation problem, while (J. Long, Shelhamer, and Darrell 2015) proposes a more general multiclass FCN architecture for image segmentation as in Figure 6.2. Its network produces an output likelihood map with the same spatial size as the input image and the depth channels for each segment class.

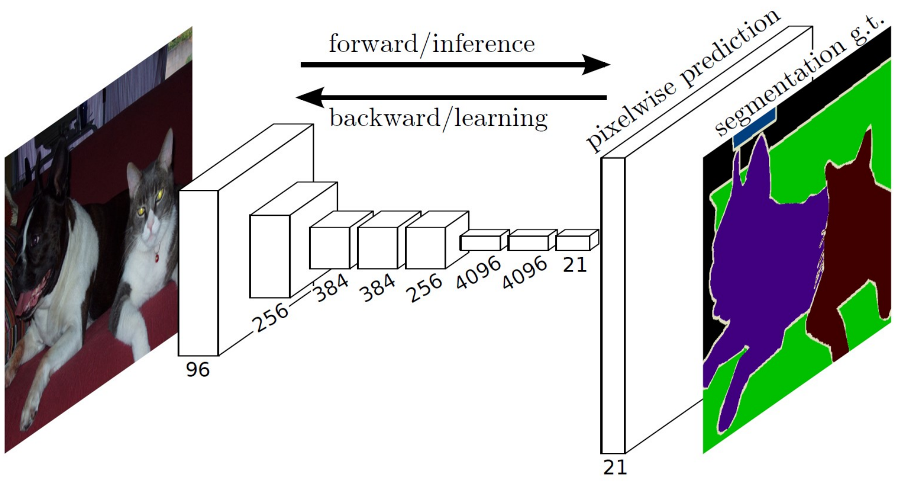


Figure 6.2: The fully convolutional neural network replaces the final FC layer with another conv. layer and proper up-sampling, which results in an output likelihood map with the same spatial size of the input. The image is taken from (J. Long, Shelhamer, and Darrell 2015)

Temporal segmentation of motion data aims to split the time-frames of an input motion sequence into several non-overlapping subsequences. These subsequences (segments) could be temporally connected, or several continuous time-frames separate them from each other. Those time-frames are generally named the gap segments (F. Zhou, De la Torre, and Hodgins 2013). For ease of reading in the rest of this document, I use the segmentation term to replace the temporal segmentation. Unsupervised motion segmentation methods do not use annotations or labeled data. Therefore, the majority of them benefit from clustering methods to assign particular parts of the motion into separate clusters in the time domain. However, their main difference lies in the way they pre-process the time-frames before the clustering step.

In (F. Zhou, De la Torre, and Hodgins 2008) and its successor version (F. Zhou, De la Torre, and Hodgins 2013), the segmentation problem is performed by the combination of DTW and kernel k-means clustering. Their optimization scheme minimizes the clustering cost by finding segments that are optimally aligned to their repetitions in the long motion stream. In methods similar to (Yi Guo, J. Gao, and F. Li 2013; Tierney, J. Gao, and Yi Guo 2014; S. Li, K. Li, and Fu 2015), the time-frames are directly clustered using the temporal clustering method as the temporal extension of sparse subspace clustering (Elhamifar and Rene Vidal 2013). In such clustering frameworks, the general aim is to enforce the encoding of consecutive time-frames similar to cluster them into the same segment. However, such temporal regularizers commonly lead to overlapping segment borders in the unsupervised setting. In a different group of unsupervised motion segmentation works such as (Stollenwerk et al. 2016; B. Krüger et al. 2017), the motion's self-similarity matrix is constructed by calculating the pairwise distance of the time-frames. In (Stollenwerk et al. 2016), a clustering technique is directly applied to such representation for segmentation of articulated hand motions, while (B. Krüger et al. 2017) performs further analysis such as graph-based relation of neighboring time-frames before applying the final clustering stage.

Even though the unsupervised segmentation methods show considerable progress over time, their performance is generally dependent on some assumptions about the underlying segments. Usually, it is required to know the approximate number of unique segments, observe segment repetitions in the stream, or fine-tune the parameter which controls the segmentation resolution.

On the other hand, supervised segmentation tries to benefit from annotated training data (already segmented motions) to remove the above domain-specific limitations and improve segmentation performance. Specifically for motion segmentation, (Lichen Wang, Z. Ding, and Fu 2018; T. Zhou et al. 2020) formulate the problem as a transfer learning framework. They find a linear mapping between time-frames of a test motion and that of a train motion followed by a temporal clustering application. However, such frameworks require pre-assumptions such as knowing the exact number of segment clusters or having a temporal coincidence between the segments of test and train motion (T. Zhou et al. 2020). There exist several supervised methods similar to (J. Y. Chang 2014; Neverova et al. 2016), which are proposed for specific motion segmentation problems (Escalera et al. 2014). They usually rely on specific spatial or geometrical processing of the problem or benefit from the available additional modalities. Regardless of their high performance for the particular problem, their application on other segmentation tasks is limited.

In the area of speech and text analysis, the combination of segmentation and classification problems for sequential data is addressed as the sequence labeling problem (N. Nguyen and Yunsong Guo 2007). The most effective sequence labeling methods are deep neural architectures constructed upon CNN, RNN, or LSTM networks. A majority of these proposed deep learning algorithms take advantage of CRF for the temporal segmentation of their input stream. The CRF is a probabilistic modeling technique, which considers dependencies between the neighboring time-frames in a graphical model (Lafferty, McCallum, and Pereira 2001). In works similar to (Zhiheng Huang, W. Xu, and Yu 2015; Lample et al. 2016; X. Ma and Hovy 2016), they add a CRF layer to the last layer of an LSTM network to perform sequence labeling. As a different method, the CRF layer is combined with a deep gated RNN in (Z. Yang, Salakhutdinov, and Cohen 2016) for word labeling. In (Gehring et al. 2017), they propose a temporal labeling architecture entirely based on CNNs and combined with gated units and attention modules. Their method is efficient regarding the accuracy and computational complexity. Despite the high performance of these methods for sequence labeling of text and speech input, their application on skeleton-based motion data may require finding a proper preprocessing stage or applying substantial changes in the networks' architecture.

As a difference to the state-of-the-art mentioned above, my proposed algorithm performs the segmentation and classification of the skeleton-based actions in an end-to-end architecture, which takes flexible input sizes and results in an interpretable model for sequential input data. In the next section, I introduce the idea of alignment kernels for CNN models, upon which I propose my novel convolutional network.

6.2 ALIGNMENT KERNELS FOR CNN

A typical CNN architecture designed for image processing tasks has 2D convolution filters in its conv. layers (LeCun et al. 1989). It is easy to observe that filters in the first conv. layer of the network mathematically behave as degree-1 polynomial kernels. In particular, applying a convolution filter **W** (conv-filter) with an $n \times n$ receptive field to

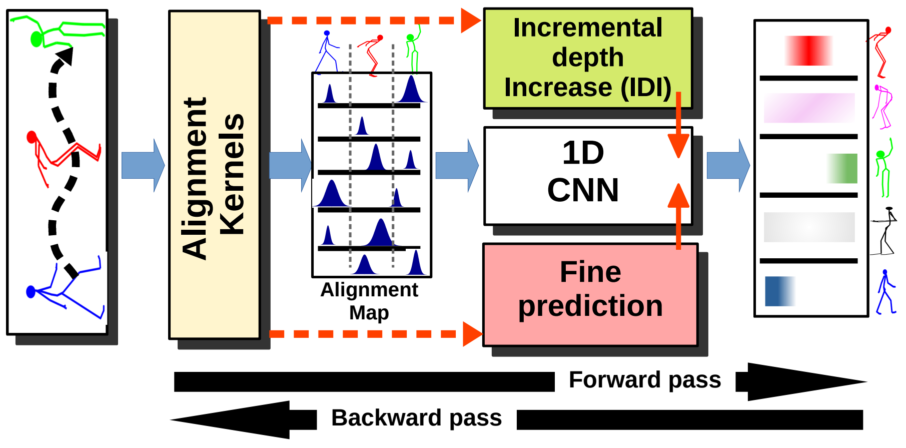


Figure 6.3: General overview of the DACNN framework. Alignment kernels preprocess the input streams. 1D-CNN performs temporal prediction based on the derived alignment map. The IDI-module automatically increases the depth of 1D-CNN. The fine-prediction unit improves the resolution of output prediction.

an image patch $\mathbf{X}$ of the same size computes the feature value o as

$$o = \sum_{i,j=1}^{n} \mathbf{x}_{ij}\mathbf{w}_{ij} + b = (\vec{x}^{\top}\vec{w} + b)^1 = \mathcal{K}_{pol1}(\vec{x}, \vec{w}), \tag{6.6}$$

where b is the filter bias, and $(\vec{x}, \vec{w})$ are the vectorized forms of $(\mathbf{X}, \mathbf{W})$, respectively. In Equation 6.6, $\mathcal{K}_{pol1}(\vec{x}, \vec{w})$ denotes the polynomial kernel of degree-1 between $\vec{x}$ and $\vec{w}$, which measures the similarity between the input patch $\mathbf{X}$ and the filter $\mathbf{W}$. Extending Equation 6.6 to all the filters in the first layer of CNN and all parts of the input data, the first layer of a CNN can be interpreted as a multiple-kernel function (Gönen and Alpaydın 2011), which measures the similarity of the given input to some exemplars/filters. However, in kernel-based classification problems, it is known that employing the squared exponential kernel (A.K.A Gaussian kernel) can better discriminate the input space compared to polynomial kernels (Camps-Valls and Bruzzone 2005). This superiority results from the flexibility and large input domain of Gaussian kernels. Although such a perspective gives us a motivation to design squared exponential filters, in the following, I discuss that employing such filters can also result in a more interpretable architecture.

Alignment Filters

In a C-class temporal classification task, we can define a frame-based labeling matrix $\mathbf{H} \in \mathbb{R}^{C\times T}$ corresponding to each skeleton-based motion sequence $\mathbf{X} \in \mathbb{R}^{d\times T}$. In $\mathbf{H}$, each entry $h_{ct} = 1$ if the t-th frame of $\mathbf{X}$ belongs to class c (when having more than one action in $\mathbf{X}$). Hence, the matrix $\mathbf{H}$ is zero elsewhere and $c \in \{1, \ldots, C\}$ in a C-class setting. Hence, we are interested in predicting the true value of $\mathbf{H}$ for each input $\mathbf{X}$ in an SBARS task. Therefore, the input layer of a CNN consists of d separate channels $\{\vec{x}_j\}_{j=1}^{d} \in \mathbb{R}^{1\times T}$, each of which contains the temporal skeleton data related to one dimension of $\mathbf{X}$. Based on the discussed rationale in the previous section, I propose the following distance-based non-linear kernel as the fundamental feature extraction unit of my DACNN architecture (Alignment layer in Figure 6.3):

$$g(\vec{x}_j|_{t_0}^{t}, \vec{f}_i^1) = e^{-\|\vec{f}_i^1 - \vec{x}_j|_{t_0}^{t}\|_2^2}, \quad \forall i = 1, \ldots, d_1, \tag{6.7}$$

where $\{\vec{f}_i^1\}_{i=1}^{d_1} \in \mathbb{R}^{1\times t}$ are the alignment filters (Al-filter) with the receptive field of t, and $\vec{x}_j|_{t_0}^{t}$ denotes a subsequence of length t starting from the t_0-th frame of channel $\vec{x}_j$. After scanning each channel of $\mathbf{X}$ by these filters with a stride s (Figure 6.4), we obtain a tensor $\mathbf{V} \in \mathbb{R}^{d_1\times T\times d}$ as the activation map. Each entry v_{jik} from $\mathbf{V}$ represents the similarity between the j-th window in channel $\vec{x}_i$ and the filter $\vec{f}_k$, and summing $\mathbf{V}$ over its second dimension results in the more summarized activation map $\mathbf{V}^1 \in \mathbb{R}^{d_1\times T}$ (Figure 6.4).

To make an analogy to a regular CNN structure, I can also reformulate the introduced alignments as an l_2-norm operator layer ($\|\vec{f}_i^1 - \vec{x}_j|_{t_0}^{t}\|_2^2$) followed by an activation-layer of $f(x) = e^{-x}$ units. This reformulation is conceptually similar to the convolution and ReLU layers of a regular CNN architecture. Therefore, designing a proper classification-based training scenario can find discriminative patterns in $\mathbf{V}^1$ with high activation values (close to 1 peak). In other words, the goal is to train filters $\vec{f}_i$ to distinguish the data classes based on their similarities to the local parts in the input channels $\{\vec{x}_j\}_{j=1}^{d}$. Hence, these filters can be seen as temporal patterns that signify relevant discriminative information in the input sequence's time axis.

Vanishing Gradient and Saturated Activation

The gradient of $g(\vec{x},\vec{f})$ in Equation 6.4 can be computed w.r.t. its parameter vector as

$$\nabla_{\vec{f}}g = -2e^{-\|\vec{f}-\vec{x}_j|_{t_0}^{t}\|_2^2}(\vec{f} - \vec{x}_j|_{t_0}^{t}). \tag{6.8}$$

Hence, when $\|\vec{f} - \vec{x}_j|_{t_0}^{t}\|_2^2$ becomes large, the activation function of Equation 6.7 and its gradient obtain infinitesimal values. This condition leads to zero updates of the Al-filter parameter $\vec{f}_i$ in a gradient-based optimization scheme (training phase). This behavior can be observed in Figure 6.5-a and Figure 6.5-b(blue curve) for the values of a length 2 filter and its elements gradient curve, respectively. Such a condition may occur when a filter $\vec{f}_i$ has a large distance to all subsequences in $\mathbf{X}$ (e.g., lousy initialization) or when the learning rate is too high.

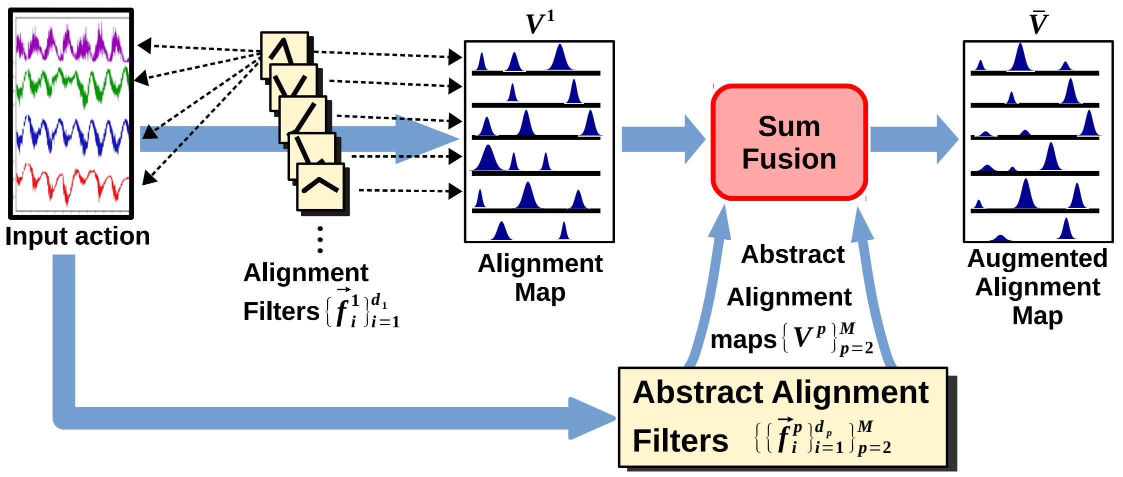

Input action
Alignment Filters $\{\vec{f}_i^1\}_{i=1}^{d_1}$
V^1
Alignment Map
Sum Fusion
Abstract Alignment maps $\{V^p\}_{p=2}^{M}$
$\bar{V}$
Augmented Alignment Map
Abstract Alignment Filters $\{\{\vec{f}_i^p\}_{i=1}^{d_p}\}_{p=2}^{M}$

Figure 6.4: The alignment layer of the network. Each Al-filter $\vec{f}_i^1$ is applied to all $\vec{f}$ channels in the input to form the i-th row in the alignment map $\mathbf{V}^1$. Adding $\mathbf{V}^1$ to the alignment maps of other Abs-filters derives the augmented map $\bar{\mathbf{V}}$.

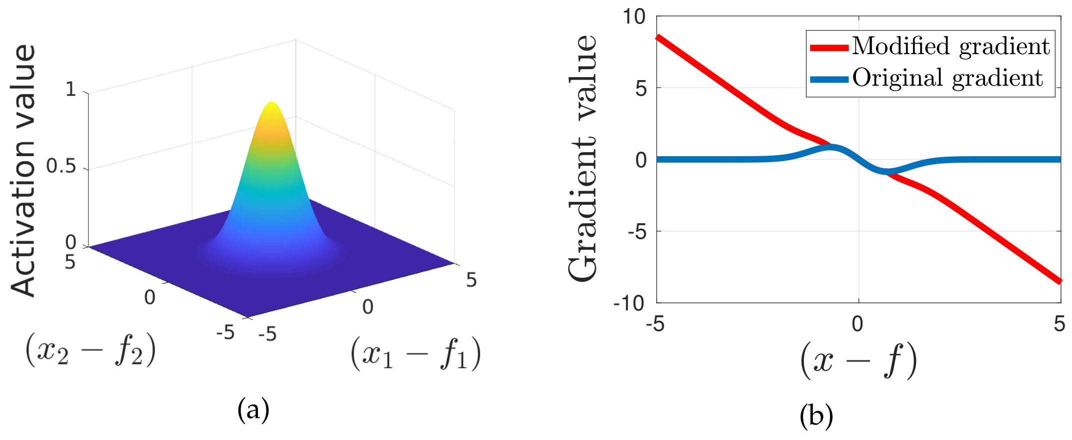


Figure 6.5: The value of the activation function $g(\vec{x}, \vec{f})$ for Al-filter of length 2 (a) and the original and modified gradients of each of its elements (b). The original gradient in Equation 6.8 becomes zero for large values of $\|\vec{x} - \vec{f}\|_2^2$, but its modified version in Equation 6.10 prevents the vanishing issue when the gradient tends to fade.

As a systematic workaround for computing the activation map $\mathbf{V}^1$, I replace the $g(\vec{x}, \vec{f})$ of Equation 6.7 with the following function

$$g(\vec{x}, \vec{f}) = (1 + a)e^{-\|\vec{x}-\vec{f}\|_2^2} - a, \tag{6.9}$$

where a is a small constant scalar (I use $a = 0.1$ in implementations). Hence, when $\|\vec{x} - \vec{f}\|$ is large, the activation function's tale in Equation 6.9 becomes saturated at a small negative value $-a$, which allows the filter $\vec{f}$ to become still updated in the backpropagation phase. This condition preserves the sparseness effect of the activation function when computing $\mathbf{V}^1$ and leads to faster convergence.

However, the gradient of $g(\vec{x}, \vec{f})$ in Equation 6.9 regarding each entry f_i is

$$\nabla_{f_i} g = -2(1 + a)e^{-\|\vec{x}-\vec{f}\|_2^2}(x_i - f_i),$$

and its second derivative with respect to f_i is computed as

$$\nabla_{f_i}^2 g = [-2 + 4(x_i - f_i)^2](1 + a)e^{-\|\vec{x}-\vec{f}\|_2^2}.$$

The value of $\nabla_{f_i}^2 g$ becomes zero when $(x_i - f_i)^2 = 0.5$, which also corresponds to the extremum points in Figure 6.5-b(blue). Extending this point to higher dimensional activation functions corresponds to the contour of $\|\vec{x} - \vec{f}\|_2^2 = 0.5$. Hence, I employ it as a threshold, after which the update of filter $\vec{f}$ becomes difficult due to the vanishing gradient issue. Therefore, in order to prevent this situation, we can compensate for the vanishing effect by adding a proportional term $-2(\vec{x} - \vec{f})$ to the gradient value, which is equivalent to assuming a regularization term of $-\|\vec{x} - \vec{f}\|_2^2$ being added to the activation function.

Based on the above analysis, I propose the modified gradient for updating $\vec{f}$ for the backpropagation phase of training as

$$\nabla_{\vec{f}} g = \begin{cases} -2(1 + a)e^{-\|\vec{\delta}\|_2^2}\vec{\delta} & \|\vec{\delta}\|_2^2 \leq 0.5 \\ -2[(1 + a)e^{-\|\vec{\delta}\|_2^2} + 1]\vec{\delta} + 2\sqrt{0.5}\, sign(\vec{\delta}) & \|\vec{\delta}\|_2^2 > 0.5 \end{cases}, \tag{6.10}$$

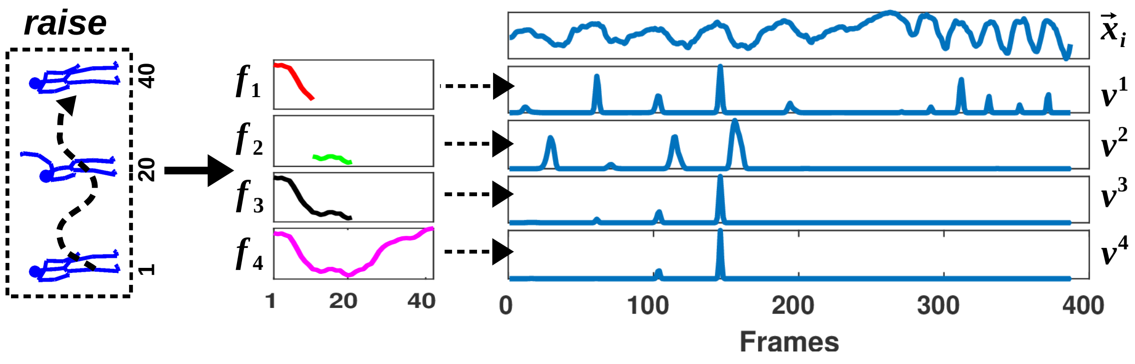


Figure 6.6: Abstract filters $\{\vec{f}_4, \vec{f}_3\}$ result in sparse alignment maps $\{v^4, v^3\}$, and $\vec{f}_4$ can be interpreted as the representative of a whole arm movement in the *raise* action.

where $\vec{\delta} = \vec{x} - \vec{f}$, and the switching threshold of $\sqrt{0.5}$ is related to the point from which the gradient of Equation 6.7 starts to decrease toward zero. In Equation 6.10, the regularization term is applied via the considered threshold contour. The extra term $2\sqrt{0.5}\ \mathrm{sign}(\vec{\Delta})$ in Equation 6.10 is used to preserve the gradient's continuity in the switching point. In Figure 6.5-b, we can compare the original gradient curve of Equation 6.8 to the modified one in Equation 6.10. The modified version compensates for the fading effect of the gradient values smoothly, starting from the point that it dramatically decreases. In such a case, the value of the gradient becomes $-2(\vec{x}_j|_{t_0}^{t} - \vec{f})$ when $\|\vec{f} - \vec{x}_j|_{t_0}^{t}\|_2^2$ has a relatively large value and prevents the filter weights from becoming saturated Figure 6.5-b(red curve).

Therefore, in the training phase, the activation values are computed based on Equation 6.9, while its gradient is obtained via Equation 6.10. This way, I can preserve the one-sided sparseness effect of the filter for faster convergence (analogous to a ReLU activation) while still preventing the filter weights from becoming saturated.

Abstract Filters

Although the proper training of the Al-filters can fit them to the small local patterns in $\mathbf{X}$ that are relevant to the given classification task, we are also interested in finding longer interpretable patterns in the action data. The benefits of finding these patterns are two folds:

1. Applying longer filters on the data leads to sparser activation maps (Figure 6.6: $\{v^4, v^3\}$ vs. $\{v^1, v^2\}$).

2. Long patterns are semantically more meaningful than short ones, and hence, enhancing the interpretability (Figure 6.6: v^4 represents a complete action).

To that aim, I define the abstract filters (Abs-filter) $\{\vec{f}_i^p\}_{i=1}^{d_p}$ with the 1D receptive field of length pt, where $p \in \mathbb{N}$. Each $\vec{f}_i^p$ is a temporal concatenation of p smaller Al-filter of size t as

$$\vec{f}_i^p = \bigoplus_{j \in I} \vec{f}_j^1,$$

where $\bigoplus_{j \in I}$ concatenates a selection of Al-filters $\vec{f}_j^1$ according to an index order given by a set I for $\vec{f}_i^p$. To find $\vec{f}_i^p$ filters automatically, I select the potential candidates among the Al-filters as the first step.

Definition 6.1. Two Al-filters $\{\vec{f}_i^1, \vec{f}_j^1\}$ of size t are candidates to form an Abs-filter if we can find a window of size t starting at the time-frame t_0 of a data channel $\vec{x}_k$ such that

$$\rho(\vec{f}_i^1, \vec{f}_j^1, \vec{x}_k|_{t_0}^{t}) := g(\vec{x}_k|_{t_0}^{t}, \vec{f}_i^1) g(\vec{x}_k|_{t_0+t}^{2t}, \vec{f}_j^1) \geq r, \tag{6.11}$$

where r is a meta-parameter scalar with a value sufficiently close to 1.

Based on definition 6.1, when $\rho(\vec{f}_i^1, \vec{f}_j^1, \vec{x}_k|_{t_0}^{t})$ has a value close to 1, there exists a temporal pattern in a channel of $\mathbf{X}$ that fits the concatenation of $(\vec{f}_i^1, \vec{f}_j^1)$. Accordingly, by using a moderate threshold in Equation 6.11 (I used $r = 0.8$ in experiments), I collect all the candidate Al-filters $\vec{f}_i^1$ in the forward pass (Figure 6.3) and form a binary-weighted graph $\mathcal{G}$. In this graph, the filters $\{\vec{f}_i^1\}_{i=1}^{d_1}$ are the nodes, those of which correspond to the definition 6.1 have undirected links of weight -1 between them. Now, I find the existing Abs-filters of different sizes via finding the shortest paths between connected nodes of $\mathcal{G}$, which is efficiently solved by the Floyd Warshall algorithm (Hougardy 2010). Hence, those Abs-filters that are subsets of the longer ones are detected and eliminated.

After finding $\mathbf{M}$ sets of Abs-filters as $\{\vec{f}_1^p, \cdots, \vec{f}_{d_p}^p\}_{p=2}^{\mathbf{M}}$, for each $p \leq \mathbf{M}$, I can apply the created filters $\{\vec{f}_i^p\}_{i=1}^{d_p}$ on the channels of $\mathbf{X}$ analogous to the description in Section 6.2. This application results in $\mathbf{M}$ abstract feature maps $\{\mathbf{V}^p \in \mathbb{R}^{d_p \times T}\}_{p=2}^{\mathbf{M}}$ with different first dimensions. Nevertheless, I enrich the content of $\mathbf{V}^1$ by fusing these abstract maps to the values of its rows resulting in the augmented map

$$\bar{\mathbf{V}} = fuse(\mathbf{V}^1, \mathbf{V}^p)_{p=2,\cdots,\mathbf{M}}. \tag{6.12}$$

The sub-fusion operator $fuse(\mathbf{V}^1, \mathbf{V}^p)$ adds the content of $\vec{v}^p(i,:)$ to the entries of $\vec{v}^1(s,:)$ if the corresponding Abs-filter $\vec{f}_i^p$ has a form of $[\vec{f}_s^1 \ldots]$. By doing the same for all rows of $\{\mathbf{V}^p\}_{p=2}^{M}$, I obtain the augmented alignment map $\bar{\mathbf{V}}$ in Figure 6.4. In that case, if $\vec{f}_i^p$ closely matches a significant pattern in the time-frame t_0 of a channel in $\mathbf{X}$, we should observe a relatively large peak (near 1) in both $\vec{v}^p(i, t_0)$ and $\vec{v}^1(s, t_0)$. But, we cannot expect the same observation in the t_0 frames of other rows in $\mathbf{V}^1$, even if any of them correspond to the constituent Al-filters in the remaining of the specific Abs-filter $\vec{f}_i^p$. As an illustration, although the Abs-filter $(\vec{f}_3, \vec{f}_4)$ in Figure 6.6 have $\vec{f}_2$ in their sequences, the alignment maps v^3 and $v2$ do not match in the time-frames of their peaks. However, they both have a large peak in the same time-frame that v^1 has a peak too, as they both start with $\vec{f}_1$. Consequently, the amplitude of the corresponding peak in the first row of the resulting $\bar{\mathbf{V}}$ is intensified by adding v^3 and $v4$ to it. Therefore, the immediate benefit of these long filters is the sparse alignment patterns we obtain in each row of $\bar{\mathbf{V}}$, which is an enriched feature map.

In the next section, I explain how to use the introduced temporal filters as an enriched feature extraction part of my proposed convolutional framework (Figure 6.3) to increase the outcome model's interpretability (Figure 6.6).

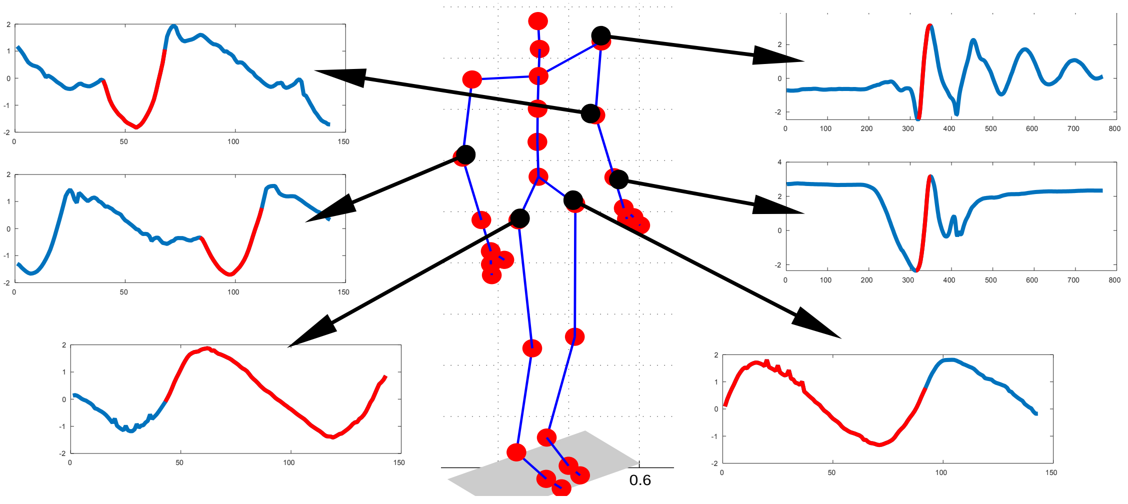


Figure 6.7: Examples of similar subsequences (red curves), which are found among different body joints and in different temporal locations related to a skeleton-based movement.

6.3 DEEP-ALIGNED CNN

In Section 6.2, I introduced the alignment kernels as the important feature extraction layer of my skeleton-based action recognition algorithm (Figure 6.3). Now, I discuss the role and rationale of the remaining parts in the DACNN architecture.

For each real-life skeleton-based motion data $\mathbf{X}$, different data channels (dimension) contain streams of continuous changes in the values of different joint's orientations. These values particularly lay in the range of $[0\ 2\pi]$ throughout normalization (or in $[\theta_0\ 2\pi]$ due to physical limitations). Therefore, it is highly expected to find short subsequences in different dimensions and temporal locations in $\mathbf{X}$ (or long patterns in symmetrical joints), which have similar shapes or curvatures (Figure 6.7). A similar characteristic can also be observed in the quaternion representation of the $\mathbf{X}$.

Based on the above observation, we can extract the relevant patterns from dimensions of $\mathbf{X}$ by applying each filter $\vec{f}_i^p$ to all the channels. As a direct benefit, this structure notably reduces the network's number of parameters and avoids an unnecessary model complexity. Therefore, the augmented activation map $\bar{\mathbf{V}}$ of Equation 6.12 is obtained by applying each filter $\vec{f}_i^p$ across all channels of $\mathbf{X}$.

I feed the derived $\bar{\mathbf{V}}$ (as in Figure 6.4) to a regular CNN, which contains 1D convolution filters (1D-conv.) with the specific architecture of Figure 6.8. Each deep layer q of the network contains two consecutive 1D-conv. layers following a max-pooling layer with the stride of 2. The 1D-conv. and pooling operations are similar to the vanilla CNN explained in Section 6.1 (Figure 6.1). The combination of conv. and pooling layers results in a d_q-channel feature map $\mathbf{O}_q$ for each deep layer q with the temporal size of $\frac{T}{2^q}$, i.e., $\mathbf{O}_q \in \mathbb{R}^{d_q \times \frac{T}{2^q}}$. Hence, the data representation becomes more abstract as we go through these deep layers.

As a complementary choice to Section 6.2, I apply the ELU operator (Clevert, Unterthiner, and Hochreiter 2015) to the input of each max-pooling layer as

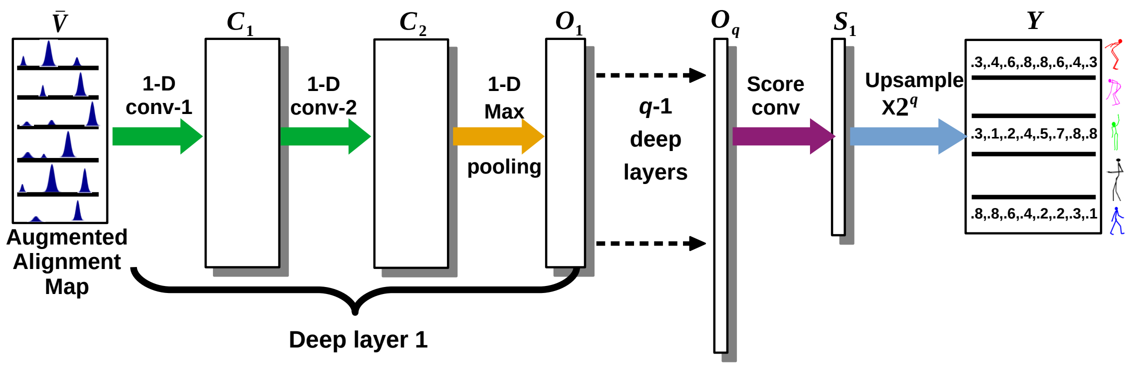


Figure 6.8: The architecture of the 1D-CNN unit in the DACNN framework. It maps the augmented alignment map $\bar{\mathbf{V}}$ (input) to the prediction map $\mathbf{O}$ (output).

$$\omega(x) = \begin{cases} x & x \geq 0 \\ e^x - 1 & x < 0 \end{cases}, \tag{6.13}$$

while $\frac{d\omega}{dx}|_{x=0} = 1$ lets the gradients flow through the layers of 1D-CNN even if they belong to any saturated filter.

Prediction Layer

I want my network structure to take inputs of different sizes and also to fit both the segmentation and classification problems. To that aim, I design a convolutional prediction layer (inspired by (J. Long, Shelhamer, and Darrell 2015)) to obtain a prediction map $\mathbf{Y} \in \mathbb{R}^{C \times T}$ as the output of 1D-CNN (last layer in Figure 6.8). Each entry y_{ct} should represent the network's confidence in assigning the t-th time-frame of $\mathbf{X}$ to class c.

To achieve the above, I first compute the score-map S_1 by applying a conv. layer with parameter matrix $\mathcal{W}_s \in \mathbb{R}^{d_q \times C \times k}$ (C output channels) to the last activation map ($\mathbf{O}_q$ in Figure 6.8). This process maps the abstract features to a score-map of size $C \times \frac{T}{2^q}$. I would like each entry i from channel c of S_1 to represent the likelihood of class c for the downsampled time-frame i. Hence, the prediction map $\mathbf{Y}$ can be computed by applying a $\times 2^q$ upsampling on the S_1 score-map. The upsampling is performed using C deconvolution filters with the stride of 2^q as in (Dumoulin and Visin 2016). Finally, to calculate the prediction error (cost) of the network, I compare the obtained prediction $\mathbf{Y}$ to the target label matrix $\mathbf{H}$ using a cross-entropies loss function as

$$LOSS = -\sum_{c=1}^{C} \sum_{t=1}^{T} h_{ct} log(y_{ct}). \tag{6.14}$$

Therefore, after DACNN is converged to an optimal point, we can predict the label matrix $\tilde{\mathbf{H}}$ of a test data $\mathbf{X}$ as

$$\tilde{h}_{ct} = \begin{cases} 1 & c = \arg\max_c \|1 - y_{ct}\|_2^2 \\ 0 & otherwise \end{cases} \qquad \forall c, t, \tag{6.15}$$

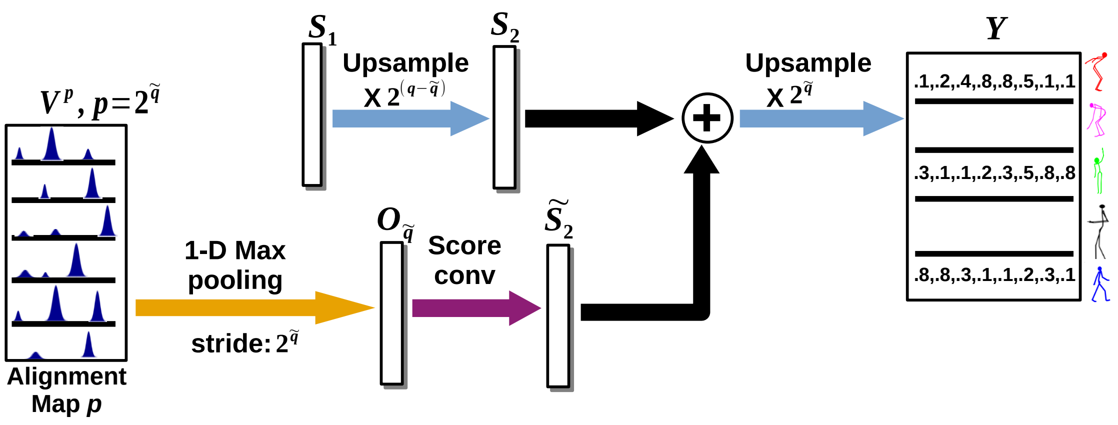


Figure 6.9: Fine-prediction module of DACNN. The skip-connections from score-maps of specific Abs-filters result in a fine-grained segmentation compared to Figure 6.8. The alignment map $\mathbf{V}^p$ in the block diagram corresponds to the largest Abs-filters of size $p = 2^{\tilde{q}}$ for $\tilde{q} \geq 2$. Nevertheless, the fine-grain process is also extended for the smaller Abs-filters of such size.

where $\tilde{h}_{ct}$ determines if an individual time-frame t in $\mathbf{X}$ belongs to the activity class c. However, in a classification setting where the whole $\mathbf{X}$ sequence belongs to only one class, I determine the class label of $\mathbf{X}$ as

$$c = \arg\min_{c} \|\vec{\mathbf{1}}_{1\times T} - y(c,:)\|_2^2, \tag{6.16}$$

in which $y(c,:)$ denotes the c-th row of $\mathbf{Y}$. Nevertheless, the training phase of DACNN is the same for both segmentation and classification tasks using the defined cost term *LOSS* in Equation 6.14.

Fine-Prediction Module

According to the structure of 1D-CNN, each pooling layer downsamples its input feature map by a factor of 2. Hence, for a network with q deep layers, the resulting score-map S_1 of width $\frac{T}{2^q}$ would be upsampled in one step to provide the prediction map $\mathbf{Y}$ with T time-frames. This one-step extreme upsampling, especially for large q, results in a large time-frame abstraction for the entries of $\mathbf{Y}$. Although this abstraction can be appreciated in a single-class activity recognition problem, it reduces the network's prediction resolution for segmentation problems. This issue escalates, especially in segment borders or in alternating activity repetitions.

As a workaround, I employ a specific skip-connection structure via the fine-prediction module of my DACNN framework (Figure 6.3) to have a fine-grained prediction map $\mathbf{Y}$. As illustrated in Figure 6.9, I obtain score-maps for the available alignment map (related to learned Abs-filters) and fuse them with the upsampling path of 1D-CNN. More precisely, I start with the largest Abs-filters of length p, such that $p = 2^{\tilde{q}}$, $\tilde{q} \geq 2$. Then, after downsampling their alignment map $\mathbf{V}^p$ by a max-pooling of size $2^{\tilde{q}}$ and applying a score-conv layer (with C output channels), we obtain the score-map of $\tilde{S}_2 \in \mathbb{R}^{C\times\frac{T}{2^{\tilde{q}}}}$. On the other side, the score-map S_1 of a q-layer 1D-CNN would be upsampled throughout the deconvolution of stride $2^{(q-\tilde{q})}$, which results in an intermediate score-map S_2 with the width of $\frac{T}{2^{\tilde{q}}}$. Hence, $\tilde{S}_1$ would be added to S_1 to improve its current resolution.

As a rationale, each $\mathbf{V}^p$ has a sparse activation-map with gap intervals of pt between the extracted matching patterns, especially when $p \geq 2$ (e.g., Figure 6.6). Hence, p-factor downsampling preserves the essential information existing in $\mathbf{V}^p$ while it still increases the resolution of $\mathbf{O}$ by order of $2^{(q-\tilde{q})}$ when $\tilde{q} < q$. Additionally, this skip-connection module intensifies the role of the Al-filters in the prediction task due to the weight-sharing between them and the Abs-filter, which increases their discriminative quality.

The above process is extended by also using other alignment maps $\{\mathbf{V}^p | p = 2^{\tilde{q}}\}_{\tilde{q}=2}^{q}$. The application of the downsampling and score-convolution on each of these $\mathbf{V}^p$ maps (in parallel passes) obtains their corresponding score-maps $\tilde{S}_i, i = 1, \ldots, \tilde{q}$. On the other side, S_1 is upsampled sequentially by the factor of 2 (after doing the first $2^{(q-\tilde{q})}$ upsampling) to provide the score-maps S_i with temporal resolutions corresponding to the $\tilde{S}_i$ maps in the fine-prediction paths.

Incremental Depth Increase

Besides finding different patterns in the input data, one aim of increasing the depth (number of middle layers) in a CNN model is to spatially/temporally compress the representation of the data. Consequently, the extracted feature maps are downscaled as deeper we go through the layers of a CNN. Nevertheless, the necessary level of compression depends on several factors, such as the complexity of the problem, the input size, network architecture, and several other factors. Considering this concern in our SBARS problem, one key element influencing the model's necessary depth is the temporal length of the essential patterns in the input data. Therefore, sequence length should help us better decide on the required depth of such a network in an SBARS task.

According to Figure 6.8, the depth of 1D-CNN (number of its deep layers) defines the extent of necessary temporal abstraction made throughout its layers. More precisely, the degree of this abstraction should coincide with the temporal length of the distinctive patterns in the dataset, which on the other hand, has a close relation to the length of the learned Abs-filters $\vec{f}^p$. To benefit from this relationship, I propose and add an incremental depth increase (IDI-module) to the DACNN framework (Figure 6.3). This module increases the depth of the 1D-CNN network as the training phase learns Abs-filters with bigger lengths.

Assume 1D-CNN has q deep layers in the current epoch, meaning that $S_1 \in \mathbb{R}^{C \times \frac{T}{2^q}}$ (Figure 6.8), and the biggest Abs-filter is $\vec{f}^p$ where $p < 2^{(q+1)}$. Now, if I find an Abs-layer $\vec{f}^{\hat{p}}$ in the next forward pass of DACNN where $p = 2^{(q+1)}$, I construct the layer $q+1$ by replacing the first score-conv of 1D-CNN with another stack of 1D-conv. and max-pooling layers followed by a new score-conv layer of the proper size. Hence, the new score-map becomes $S_1 \in \mathbb{R}^{C \times \frac{T}{2^{(q+1)}}}$.

However, before connecting the $(q+1)$-th layer to 1D-CNN, the IDI-module first pre-trains its layers' initial weights in an isolated backpropagation loop (Figure 6.10). Considering $\tilde{S}_1$ as the initial score-map of layer $q+1$ and $\hat{S}_1$ as its factor-2 upsampled map, I train the weights of the $(q+1)$-th layer to minimize the following cost function:

$$err(q, q+1) = \|S_1 - \hat{S}_1\|_F^2, \tag{6.17}$$

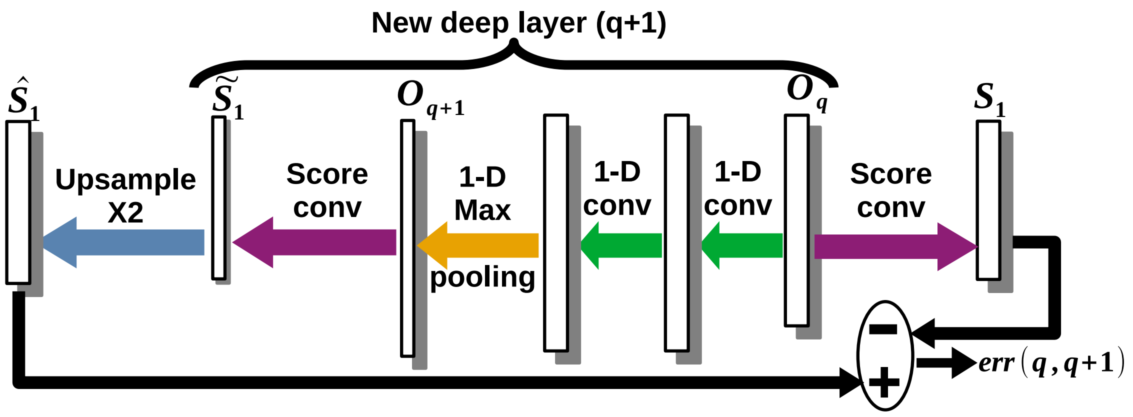


Figure 6.10: The IDI-module increases the deep layers of 1D-CNN while measuring the error of this extension.

where $err(q, q+1)$ indirectly indicates the current distance between the dynamic states of the new layer and DACNN.

I perform the above backpropagation parallel to the main training loop of DACNN until $err(q, q+1)$ becomes sufficiently small. Afterward, the weights of layer $q+1$ are updated by the main backpropagation loop of DACNN. Figure 6.13 provides an example of the smooth learning curve resulted from using IDI-module.

Sparse Activation Maps

As a crucial observation in SBARS tasks, it is rare for a temporal pattern to occur in many channels of $\{\vec{x}_i\}_{i=1}^d$ at an arbitrary time-frame t. Even in synchronized motions, this may happen only in 2-4 symmetrical joints (e.g., simultaneously raising both hands or both legs). In addition, in each action segment, a specific temporal pattern may occur a small number of times in each $\vec{x}_i$ channel. Even by considering the temporal repetitions of one action, this number would be neglectable compared to T as the temporal length of $\vec{x}_i$.

Considering that the alignment filters are meant to represent relevant temporal patterns in the input data, I can project the above concepts directly to the training of these filters.

Knowing that all Abs-filters $\{\vec{f}_i^p\}_{p\geq 2}$ share the same parameters with their constituent Al-filters $\vec{f}_i^1$, I apply a sparsity objective $Sp = \sum_{p=1}^M \|\mathbf{V}^p\|_1$ to the backpropagation of DACNN to compute the optimal value of each $\vec{f}_i^1$ as

$$\vec{f}_i^{1*} = \underset{\vec{f}_i^1}{\arg\min}\, LOSS(\vec{f}_i^1) + \lambda Sp(\vec{f}_i^1), \tag{6.18}$$

in which λ is the scalar that controls the sparsity gain in the training phase. I can reformulate $Sp(\vec{f}_i^1) = \sum_{t,k} \hat{v}_{tki}$, where $\hat{v}_{tki}$ considers the entries in all $\{\mathbf{V}^p\}_{p=1}^M$ in which $\vec{f}_i^1$ is involved. Consequently, I optimize Equation 6.18 based on the following chain rule:

$$\frac{\partial LOSS(\vec{f}_i^1)}{\partial \vec{f}_i^1} = \sum_{t,k} \left(\frac{\partial LOSS(\vec{f}_i^1)}{\partial \hat{v}_{tki}} + \lambda\right) \frac{\partial \hat{v}_{tki}}{\partial \vec{f}_i^1}, \tag{6.19}$$

where $\frac{\partial \hat{v}_{tki}}{\partial \vec{f}_i^1} = -2g(\vec{x}_j|_t, \vec{f}_i^1)(\vec{x}_j|_t - \vec{f}_i^1)$. In practice, the sparsity term Sp redirects the training path toward the efficient use of the resources (Al-filters) in finding the most distinctive local patterns in the given data, which can significantly reduce the network's complexity and prevent it from overfitting (Changpinyo, Sandler, and Zhmoginov 2017).

By putting the detail of all building units in Figure 6.3 together, we provide a complete description of the framework in Figure A.5. In the next section, In the next section, I empirically analyze the proposed DACNN network's performance with applications on mocap benchmarks.

6.4 EXPERIMENTS

In this section, I implement my DACNN framework on real-world action recognition benchmarks to obtain an empirical evaluation of its performance in SBARS tasks. I compare DACNN to different state-of-the-art alternatives in two segmentation and classification settings, for which I employ different performance measures. Furthermore, I perform additional analysis to study the proposed DACNN from various perspectives.

Implementation Setup

In the implementation of DACNN, I choose the kernel size of $t = 3$ for Al-filters and 1D kernels with the receptive field of $k = 3$ for the conv-filters. This choice of t for Al-filters provides them enough angular freedom to learn any local curvature in the data. I also apply dropout (N. Srivastava et al. 2014) with a rate of 0.30 to prevent DACNN from overfitting. Specifically, during the training phase, 30% of output channels in each conv. layer is removed by sampling from the Bernoulli distribution. I train DACNN using the Adam approach (Kingma and Ba 2014), and the skeleton data is normalized per dimension, which allows the filters to be aligned to local parts of different joints (channels of $\mathbf{X}$). The sparsity control parameter λ in Equation 6.18 is chosen via cross-validation on the training set for $\lambda \in [0.01\ 0.5]$. Regarding the implementation of the baseline algorithms, either I use their publicly available codes and tune their parameters with cross-validation on the training set or refer to their reported results in the relevant publications. For each dataset, I use its typical evaluation setting reported in the literature.

I use a warm initialization for the Al-filters before starting the training phase. I assign their values by random choice of t-length subsequences from different $\mathbf{X}$ data sequences. This strategy prevents the filters from becoming saturated initially and results in a convergence speed-up. Additionally, it is practical to perform subsampling for high-resolution inputs. This strategy reduces the required depth of the network. As another solution, it is possible to choose a larger t (e.g., $t \in [5, 7, 11, 13]$) as the basic length of the Al-filters $\{\vec{f}_i^1\}$ which reduces the required depth in the 1D-CNN unit without reducing the resolution of the data. However, the first solution is easier to implement in practice.

The Abs-filters in my network share weights with their constituent Al-filters $\{\vec{f}_i^1\}_{i=1}^{d_1}$. Therefore, we only need to update the weights of the base Al-filters in the training phase. For instance, if $\vec{f}_5^3 = [\vec{f}_4^1 \vec{f}_3^1 \vec{f}_1^1]$, in the backpropagation phase, directly learning the weights of $\{\vec{f}_4^1, \vec{f}_3^1, \vec{f}_1^1\}$ would also update the parameters of $\vec{f}_5^3$ as well. In addition, to have the same temporal length T for all alignment maps $\{\mathbf{V}^p\}_{p=2}^M$, I apply a zero padding of size $\frac{pt}{2}$ for each filter with the receptive field of size pt and put zeros for the remaining $\frac{pt}{2} + 1$

Table 6.1: Segmentation accuracy for Montalbano V2 and CMU Mocap based on Jaccard Index (*JI*).

Montalbano		CMU Mocap	
Method	*JI*	Method	*JI*
Terrier (2015)	53.9	HACA (2013)	71.4
Quads (2015)	74.6	SSSM (2017)	75.7
Ismar (2015)	74.7	TSC (2015)	73.5
LRT (2018)	74.8	LRT (2018)	82.1
Gesture Labeling (2014)	78.4	BiLSTM-CRF (2019)	86.1
BiLSTM-CRF (2019)	78.7	ConvS2S (2017)	86.4
CNN+LSTM2 (2018)	79.5	RNN-CRF (2016)	88.1
End2End (2016)	81.7	End2End (2016)	88.4
RNN-CRF (2016)	82.2		
Moddrop (2016)	83.3		
DACNN (**Proposed**)	**87.2**	DACNN (**Proposed**)	**92.5**

entries. This specific zero-padding gives the chance of checking the alignment between a filter and an unfinished input pattern at the input sequence's end-border.

Regarding the debugging of the framework, in general, the progress in learning the abs-filters shows whether the network structure suits the given task's complexity. As a common observation, when no (or limited number of) Abs-filter is formed in the training phase, it is required to increase the number of used Al-filters (d_1). Also, regarding the network's initial design, we can start the training without having the sparsity term ($\lambda = 0$) or the IDI-module. Without using the IDI-module, the learned Abs-filter's length and quality can be checked manually to see if the learning progress is sensible, especially regarding the size of convolutional layers in 1D-CNN and the number of Al-filters used in the Alignment-layer of DACNN. As the next step, I add the IDI-module and the sparsity term in the loop to let DACNN grow its depth appropriately. Generally, I advise trying for a satisfactory result without the sparsity term at first and tuning λ afterward to improve the outcome. Although the above guideline is not always the most optimal step for tuning and debugging, it provides a straightforward routine to initialize the experiments for any new dataset.

Action Segmentation

In this set of experiments, I evaluate my designed DACNN framework with respect to the segmentation task. The segmentation task is applied to CMU Mocap -segment and Montalbano V2 datasets (Section 2.4), for which I predict the label of each time-frame in the input motion $\mathbf{X}$ using Equation 6.15. The action segmentation performance of each algorithm would be evaluated by using the Jaccard index

$$JI = \frac{\tilde{\mathbf{H}} \cap \mathbf{H}}{\tilde{\mathbf{H}} \cup \mathbf{H}}. \tag{6.20}$$

CMU Mocap -segment:

The segmentation performance of DACNN on the CMU Mocap dataset is evaluated by its comparison to SSSM (B. Krüger et al. 2017), TSC (S. Li, K. Li, and Fu 2015), and HACA (F. Zhou, De la Torre, and Hodgins 2013) as unsupervised temporal segmentation approaches, and to LRT (Lichen Wang, Z. Ding, and Fu 2018), ConvS2S (Gehring et al. 2017), End2End (X. Ma and Hovy 2016), BiLSTM-CRF (Alzaidy, Caragea, and Giles 2019), and RNN-CRF (Z. Yang, Salakhutdinov, and Cohen 2016) as supervised segmentation frameworks. Except for the LRT algorithm, which is a transfer learning segmentation method, other selected supervised approaches are deep sequence labeling frameworks. In the methods that use word embedding input, I replace the embedding layer with the joints' quaternion values through the time-frames.

As reported in Table 6.1, deep learning algorithms' performances have substantial distances from the unsupervised methods. Compared to them, DACNN outperforms the best method (End2End) with a notable margin of 4.1%. This result supports the effectiveness of the DACNN architecture regarding the supervised segmentation of skeleton-based action data. Generally, the supervise methods show a netter segmentation accuracy compared to the unsupervised approaches. This difference is due to the supervised algorithms' access to prior knowledge in the form of training data. In comparison, the LRT method has a place between the above two groups as it benefits from annotated training data, but it still relies on a final unsupervised clustering step.

In Figure 6.11, the segmentation results of some baselines are visually evaluated on one of the challenging sequences from the CMU dataset (Subject 86, trial 03). Compared to the unsupervised method SSSM, the supervised algorithms better identify the gaps (insignificant actions) because they are optimized by the relevant label information in the training phase. Additionally, SSSM finds sub-clusters in some segments (e.g., *walk* and *kick*), which is not the desired outcome in supervised temporal segmentation. The LRT transfer learning has fewer segmentation mistakes than SSSM due to its supervised mapping. However, its regional regularizer produces dramatic mistakes regarding the correct location of the segment borders.

The algorithms DACNN and DACNN-nf (without fine-prediction unit) have fewer mistakes in the results, especially in the gap areas. Besides, DACNN obtains fewer overlapping segments and better predictions in the gap areas compared to DACNN-

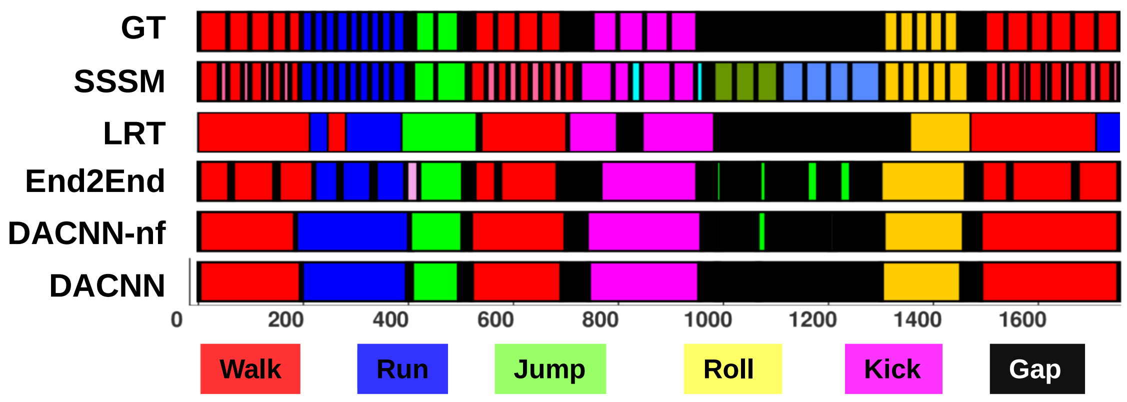


Figure 6.11: The segmentation results on the CMU dataset (Subject 86, trial 03). GT: ground truth segments. DACNN-nf: the DACNN framework without the fine-prediction unit.

nf, which emphasizes the positive effect of the fine-prediction module in DACNN. In comparison, End2End has lower performance than DACNN and DACNN-nf regarding the gap areas and segments.

Comparing DACNN and other supervised algorithms to SSSM and ground truth, the supervise methods do not split the long segments (e.g., several continuous walking cycles) into their small sub-sequences (each walking cycle). This behavior is due to the annotation regime used for training data, which labels all time-frames of each action the same way. Nevertheless, given that an algorithm can recognize a wide segment correctly, it is easy to identify the repeated subsequences inside it using self-similarity information (B. Krüger et al. 2017). Such an application is particularly simplified for most unsupervised segmentation methods when we know all the subsequences belong to one action cluster.

Montalbano V2 Dataset:

For this dataset, I compare DACNN to the following baselines regarding the temporal segmentation accuracy: Terrier (Escalera et al. 2014), Quads (Escalera et al. 2014), Ismar (Escalera et al. 2014), Gesture Labeling (J. Y. Chang 2014), Moddrop (Neverova et al. 2016), CNN+LSTM2 algorithm (Núñez et al. 2018), End2End (X. Ma and Hovy 2016), BiLSTM-CRF (Alzaidy, Caragea, and Giles 2019), and RNN-CRF (Z. Yang, Salakhutdinov, and Cohen 2016). The algorithms YNL, Terrier, Quads, and Ismar are the officially reported segmentation methods designed and proposed for the Montalbano competition (Escalera et al. 2014). The original Moddrop network uses the three different modalities (video, Mocap, and audio) in parallel paths, and its architecture is specifically designed by considering the domain knowledge for this dataset. However, I only use its mocap-based version in my experiments, such that its outcome could be comparable to other baselines. The CNN+LSTM2 algorithm does not perform any segmentation and can only be applied to a pre-segmented version of Montalbano. Hence, I only use it as a baseline classifier that is applied to this dataset in the literature (Núñez et al. 2018).

According to Table 6.1, my proposed method obtained a higher performance than the best baseline (2.9% higher Jaccard index compared to Moddrop). The best alternative approach (Moddrop) has a convolutional architecture designed based on the specific geometrical description of the given gesture segmentation problem, which explains its relatively high performance on this dataset. The other general sequence-based labeling methods, such as End2End and RNN-CRF, obtained the next best places after Moddrop. Although DACNN is a general sequence labeling method too, its specific prototype-based architecture could perform effective labeling of data frames even better than a domain-specific method such as Moddrop.

Table 6.2: Recognition accuracy (%) for SYSU-3D dataset.

Method	Acc.	Method	Acc.
LAFF(SKL) (2016)	55.2	CNN+DPRL (2018)	76.7
Dynamic Sk. (2015)	75.2	GCA-LSTM (2017)	78.6
ST-LSTM+TG (2018)	76.7	VA-LSTM (2017)	77.8
SR-TSL (2018)	82.0	DACNN (Proposed)	**84.3**

Action Recognition Results

To empirically evaluate my DACNN algorithm for action recognition tasks, I evaluate it on SYSU and NTU datasets (Section 2.4) in a classification setting. Hence, for each test data sample $\mathbf{X}$, the network predicts its corresponding class label c via Equation 6.16. To recognition performance of the utilized algorithms are evaluated based on the classification accuracy

$$Acc = 100 \times \frac{\#\text{correctly classified sequences}}{N}. \tag{6.21}$$

SYSU Dataset:

For the empirical evaluations, I compare my DACNN algorithm to other baselines, including CNN+DPRL (Y. Tang et al. 2018), ST-LSTM+Trust Gate (J. Liu, Shahroudy, et al. 2018), Dynamic Skeletons (J.-F. Hu, W.-S. Zheng, Lai, et al. 2015), LAFF(SKL) (J.-F. Hu, W.-S. Zheng, L. Ma, et al. 2016), SR-TSL (Si et al. 2018), VA-LSTM (P. Zhang et al. 2017), and GCA-LSTM (J. Liu, G. Wang, et al. 2017). Several of these methods are constructed from the combination of powerful yet complex deep learning algorithms, such as LSTM, graph-based NN, and deep RL. In Table 6.2, my proposed DACNN framework outperforms the best state-of-the-art method (SR-TSL) with a 2.3% margin. It is important to consider that SR-TSL benefits from graph-based spatial analysis of the skeleton information before applying its parallel LSTM-based temporal data processing blocks.

NTU Dataset:

For the NTU dataset, I evaluate DACNN in comparison to the state-of-the-art methods from the literature: HBRNN-L (Y. Du, Wei Wang, and Liang Wang 2015), Dynamic Skeletons (J.-F. Hu, W.-S. Zheng, Lai, et al. 2015), LieNet-3Blocks (Zhiwu Huang et al. 2017), Part-aware LSTM (Shahroudy et al. 2016), ST-LSTM+Trust Gate(J. Liu, Shahroudy, et al. 2018), CNN+LSTM2 (Núñez et al. 2018), Two-Stream RNN (H. Wang and Liang Wang 2017), STA-LSTM (S. Song et al. 2017), GCA-LSTM (stepwise) (J. Liu, G. Wang, et al. 2017), Clips+CNN+MTLN (Ke et al. 2017), View invariant (M. Liu, Hong Liu, and Chen Chen 2017), CNN+DPRL (Y. Tang et al. 2018), VA-LSTM (P. Zhang et al. 2017),

Table 6.3: Recognition accuracy (%) for NTU dataset regarding Cross-View (CV) and Cross-Subject (CS).

Method	CS	CV	Method	CS	CV
HBRNN-L (2015)	59.1	64.0	Clips+CNN (2017)	79.6	84.8
Dynamic Sk. (2015)	60.2	65.2	View invariant (2017)	80.0	87.2
LieNet-3Blocks (2017)	63.1	68.4	CNN+DPRL (2018)	82.3	87.7
Part-aware LSTM (2016)	62.9	70.3	VA-LSTM (2017)	79.5	87.9
CNN+LSTM2 (2018)	67.5	76.2	ST-GCN (2018)	81.5	88.3
ST-LSTM+TG (2018)	69.2	78.7	Two-Stream CNN (2015)	83.1	89.1
Two-Stream RNN (2017)	72.1	79.7	CNN+LSTM (2017)	82.9	91.0
STA-LSTM (2017)	74.1	81.8	SR-TSL (2018)	**84.8**	**92.4**
GCA-LSTM (2017)	76.3	84.5	DACNN (Proposed)	83.8	90.7

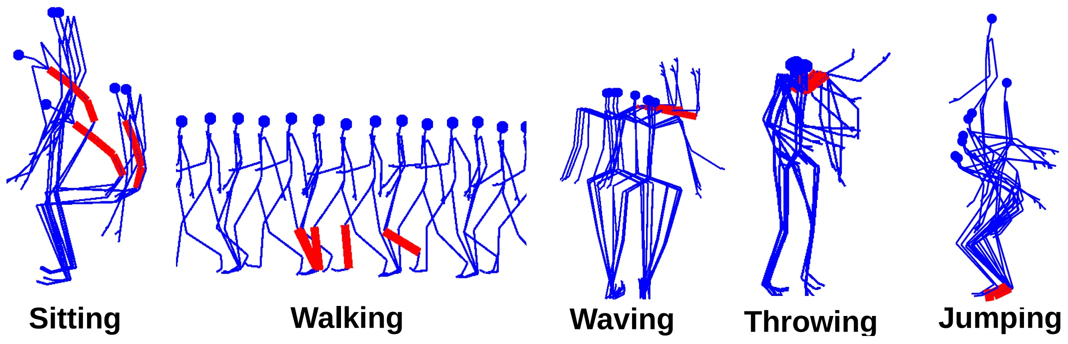


Figure 6.12: Visualization of the Abs-filters learned by DACNN on the NTU dataset and the classes to which they are mostly related. Red links indicate the body parts in which the Abs-filters have high alignment values.

ST-GCN (Sijie Yan, Xiong, and D. Lin 2018), CNN+LSTM (C. Li et al. 2017), Two-Stream CNN (Y. Du, Fu, and Liang Wang 2015), and SR-TSL (Si et al. 2018).

As I can see in Table 6.3, DACNN did not beat SR-TSL and CNN+LSTM in recognition accuracy. Nevertheless, It still achieves a competitive result compared to other recent state-of-the-art algorithms, such as ST-CGN, CNN+DRPL, and VA-LSTM, and even outperforms CNN+LSTM in the CS settings. It is important to note that the NTU dataset is recorded in a very constrained experimental setting, which is an advantage for the methods that considerably rely on the spatial processing of human poses (such as SR-TSL and CNN+LSTM).

Further Empirical Evaluations

In this section, I analyze my DACNN framework also from other empirical perspectives. Mainly, my concern is to study the interpretation of the Abs-filters, the performance of IDI-module, and the effect of sparsity regulation. I also investigate the role of different modules of the DACNN architecture and their effect on the outcome.

Interpreting the Abs-filters

Apart from the action recognition and segmentation performance, one important motivation for the specific design of DACNN was its interpretable model. Accordingly, we are interested in visualizing the learned Abs-filters of DACNN to investigate any semantic (meaningful) pattern among them. Relatively, a particular strength of DACNN compared to other deep neural networks is its simplicity in visualization and interpretation of its trained filters (Abs-filters). To that aim, I associate each filter $\vec{f}_i^p$ to a class c if

$$c = \underset{c}{\arg\max} \sum_{\mathbf{X} \in \text{class } c} \|\vec{v}^p(i,:)|_{\mathbf{X}}\|_2^2, \tag{6.22}$$

in which $\vec{v}^p(i,:)|_{\mathbf{X}}$ denotes the i-th row of the alignment map $\mathbf{V}^p$ after applying $\vec{f}_i^p$ on all channels (dimensions) of $\mathbf{X}$. In Figure 6.12, I visualize some of the Abs-filters learned by DACNN after being trained on the NTU dataset. These filters are mostly related to the action classes *walking*, *waving*, *sitting*, *jumping*, and *throwing*. It is clear that each filter has learned a semantic subsequence from one joint of the whole action. For instance,

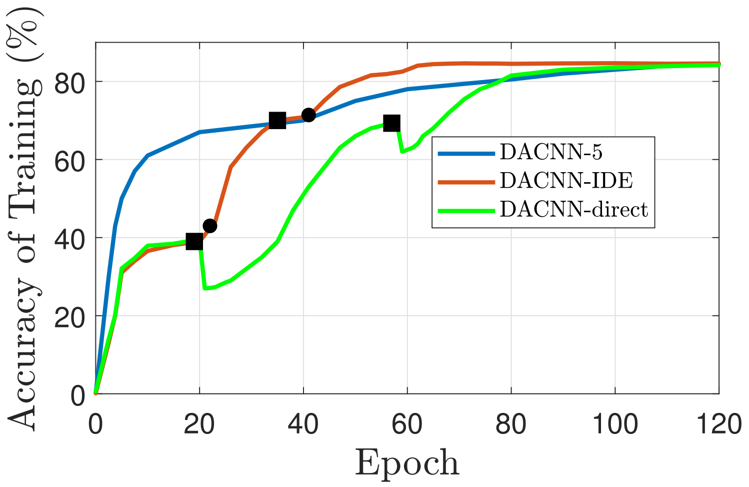


Figure 6.13: The effect of IDI-module in the learning curve of DACNN on Montalbano V2. The squares show the epochs when a new deep layer is initialized, and the circles indicate when the IDI-module finishes the pre-training of that layer and adds it to the main loop of DACNN.

one Abs-filter is aligned with the *left foot*'s movement in the *jumping* action before the subject's foot is detached from the ground. As another example from Figure 6.12, another filter has learned half of a *walking* cycle on the *right foot* as a relevant temporal pattern in distinguishing walking sequences from other motion types. Considering other Abs-filters illustrated in Figure 6.12, each of them has recognized a specific temporal pattern that facilitates the separation of that class from others.

Incremental Layer Extension

I investigate the IDI-module's individual effect on training performance via the implementation of DACNN on the Montalbano V2 dataset. As in Figure 6.13, I initialize DACNN with 2 deep layers in its 1D-CNN module (depth 2). Then, as the network constructs the Abs-filters during the training, the IDI-module increases its depth incrementally until this progress becomes saturated at a depth of 4. Figure 6.13 also contains the accuracy curve related to the training of DACNN-5 and DACNN-direct without IDI-modules. DACNN-5 has a fixed depth of 5 during its training, while I incrementally increase the depth of DACNN-direct, but without the pre-training (Figure 6.10) of its new layers.

According to Figure 6.13, although DACNN and DACNN-5 both converge to the same accuracy performance, DACNN (with IDI-module) presents a notably faster convergence and also chooses only up to 4 deep layers as the necessary level of complexity for training DACNN on this dataset. Also, the trained DACNN-5 learns Abs-filters up to the size of $54 = 3 \times 18$. However, the minimum filter size that triggers the initiation of depth 5 is 3×2^5. Hence, DACNN-5 could not find any discriminative pattern that is large enough to justify the necessity of having the 5th depth in its structure. Furthermore, DACNN-direct shows dramatic decreases in its learning curve each time a new deep layer is added to its network, which is due to the disturbance the new untrained initial weights cause in training.

Role of Sparsity

To better study the role of the sparsity loss Sp in the recognition performance, I repeat the experiment on the NTU dataset by varying the parameter λ of Equation 6.18 in the range [0 1]. Based on Table 6.4, this term positively affects the accuracy and convergence speed when λ is in the range [0.2 0.5]. However, very large or minimal values of λ lead to low accuracy and slow convergence, respectively.

Ablation Study

To study the individual role of each module in DACNN (Figure 6.3), I perform an ablation study by repeating the action recognition experiments for the DACNN framework variants (Table 6.5). In DACNN-nf, I remove the fine-prediction module, and I do not employ any Abs-filter in DACNN-Al. The DACNN-1D network is similar to DACNN-Al, but it uses 1D-conv. filters instead of the Al-filters. According to the results in Table 6.5, the lower performance of DACNN-nf compared to DACNN shows the positive effect of the Abs-filter skip-connections (Figure 6.9) in improving the accuracy of the final prediction in 1D-CNN.

Nevertheless, the accuracy of DACNN-nf is still close to DACNN and is even higher than the state-of-the-art for SYS-3D and Montalbano. Removing the Abs-filters from DACNN causes a notable decrease in the performance of DACNN-Al, which justifies the significant role of these filters in extracting the essential patterns in the data. However, DACNN-Al obtains higher accuracies than DACNN-1D, which proves how effective the alignment filters are regarding the SBARS problems. I already demonstrated the individual effect of the fine-prediction unit (in segmentation) and IDI-module by Figures 6.11 and 6.13, respectively.

Stability of DACNN

The DACNN framework has a stable training phase due to its specific structure. The 1D-CNN module has a similar structure to the fully-convolutional neural networks (J. Long, Shelhamer, and Darrell 2015), and the alignment layer contains convex operation units, which are fully differentiable. Hence, the main body of DACNN follows a routine training procedure similar to other typical CNN networks. One example of the learning curve is provided by Figure 6.13, which studies the progress of the network's training with and without the IDI-module.

Regarding the construction of the Abs-filters, fine-prediction layers, and the IDI-module, it is crucial to notice that the functionality of these designed modules principally relies on some Al-filters that are trained sufficiently and already reached the stable regions of the optimization space. For instance, a portion of $\{\vec{f}_i^1\}_{i=1}^{d_1}$ can possess specific

Table 6.4: Effect of the sparsity parameter λ on accuracy (%) and convergence epoch (C.ep.) for the NTU dataset.

λ	Acc.	C.ep.	λ	Acc.	C.ep.	λ	Acc.	C.ep.
0	89.3	310	0.4	90.4	231	0.8	87.8	176
0.1	89.8	289	0.5	90.0	210	0.9	87.7	169
0.2	90.5	273	0.6	88.4	195	1.0	86.5	**161**
0.3	**90.7**	254	0.7	88.1	190	-	-	-

Table 6.5: Ablation Study: Prediction accuracies (%) for partial implementations of DACNN on three selected datasets.

Method	SYS-3D	NTU CS	NTU CV	Mont.
DACNN-1D	71.8	70.5	75.5	70.54
DACNN-Al	78.4	77.4	85.3	82.53
DACNN-nf	82.5	81.8	88.9	84.15
DACNN (Figure 6.3)	**84.3**	**83.0**	**90.7**	**85.2**

patterns inside the data, but they need to be fine-tuned. In particular, the IDI-module starts to learn the initial parameters of a new deep layer when DACNN already reached the low-slope part of its learning curve (Figure 6.13). Additionally, the new deep layer has fewer parameters to update than the main DACNN framework, making their fusion notably fast and smooth.

6.5 conclusions

In this chapter, I proposed a deep-aligned convolutional neural network for skeleton-based action recognition and temporal segmentation. As a significant difference between this framework and those of previous chapters, DACNN directly analyzes the temporal content of input sequences and seeks relevant information in that axis specifically for the given classification task. This network is constructed upon introducing the novel concept of temporal alignment filters for CNN architectures. Employing these filters in the first layer of a CNN model is an efficient choice for skeleton-based motion data classification compared to regular convolution filters. They extract crucial local patterns in the temporal dimensions of the data to better discriminate the action classes. Besides the competitive performance of my DACNN framework compared to the state-of-the-art, its extracted features (learned Abs-filters) are easily interpretable regarding their semantic contents. On the other hand, the existing advanced state-of-the-art frameworks are typically combinations of CNN models and other deep architectures. Therefore, we can expect that incorporating such novel filter types can also enhance the performance of such advanced architectures.

Furthermore, I designed an IDI-module to smoothly increase my network's depth according to the data structure without disrupting the training process. My empirical evaluation of DCANN on different SBARS benchmarks supports my claims regarding the performance and benefits of my network. I believe that the idea of incorporating alignments in CNNs can be further studied in other relevant areas, such as relevance analysis and generative adversarial networks. The layer extension idea can also be further studied regarding its application to other general deep architectures such as RNN or LSTM.

7 CONCLUSIONS AND OUTLOOK

In this dissertation, I have addressed the motion data analysis problem with a specific focus on the interpretability and explainability aspects. To that aim, I have proposed novel semantically interpretable models in four machine learning areas of metric learning, sparse embedding, feature selection, and deep learning. The proposed models are empirically evaluated in each category by implementations on real-world motion benchmarks and using appropriate performance measures.

In Chapter 3, I have applied metric learning on motion data to improve its representation in favor of distance-based supervised tasks. More specifically, I have proposed a novel distance-based metric learning framework, which benefits from DTW as a robust alignment technique for motion sequences. The proposed framework transfers motion data to another space in which semantically similar motions are located in tighter neighborhoods while semantically different motions are pushed further away from each other. Empirically, I have demonstrated that the nearest-neighbor classification of motion benchmarks is improved in the space obtained by the learned metric. Therefore, the learned metric has improved the representation of motion data in local neighborhoods of the distance space.

Furthermore, I have shown in Chapter 3 that the proposed distance-based metric learning algorithm gives us the possibility to perform also auxiliary analysis such as metric regularization on motion data. This post-processing regularization step interprets the obtained metric in terms of the most relevant body joints. To that aim, it reduces the typically existing correlations between motion dimensions (body joints movements) and reveals the semantically significant dimensions to the given supervised task. As another presented topic of Chapter 3, I have discussed the effect of target selections on the performance of neighborhood-based metric learning. Mathematically, I have addressed the connection between the geometric formation of the selected targets and the feasibility of obtaining an optimal metric. Furthermore, I have shown how we can benefit from the introduced concept to improve the target selection and, consequently, the learned metric's quality for real-world benchmarks.

Another practical concern in the processing of motion data is obtaining a sparse embedding to the vector space. Such embedding can considerably reduce the representation's space complexity and also opens up the possibility of applying advanced algorithms on motion data, which are mainly designed for a vectorial source of information. The existing sparse embedding models for data types similar to motion are not meaningfully interpretable w.r.t. the original motion resources. However, this is an essential property required by a practitioner or a domain expert. In Chapter 4, I have shown that by benefiting from the semantic similarity between motion sequences of the same category, we can obtain a sparse and interpretable encoding for each motion sample. To that aim, I have proposed a kernel-based non-negative dictionary learning framework that encodes each motion data by its connections to other similar motion sequences by means of learning an intermediate dictionary. I have shown that the proposed framework's non-negativity property improves the interpretation of the resulting sparse encoding relying on its meaningful (motion-based) building blocks.

I also have extended the proposed non-negative sparse coding framework of Chapter 4 to two supervised models, which can enrich the sparse encodings in case supervised information such as data labeling is available. These frameworks have different formulations and consequently differ in their discriminative performance as well as their computational complexity. I have proposed suitable optimization algorithms o train each encoding framework efficiently. Experimentally, I have demonstrated that both proposed supervised algorithms obtain enriched sparse encodings of motion data, which outperform the other alternatives in terms of the model's semantic interpretability and the discriminative quality of the obtained encodings. Furthermore, as an unsupervised extension of the base proposed non-negative sparse coding framework, I have proposed a novel kernel-based subspace sparse clustering. This algorithm encodes each motion sequence directly in terms of a sparse set of other semantically similar samples in the feature space's local neighborhoods. The non-negativity term makes the obtained self-representative graph of the mocap dataset interpretable regarding its local connections between similar motion sequences. Through empirical evaluations, I have demonstrated that the obtained unsupervised encoding can reveal the underlying subspaces in which individual motion categories lie.

When considering the multivariate description of motion data, another challenge in motion data analysis is to obtain interpretable feature selection models. A motion sequence is represented in such desired models by selecting specific features that provide semantic connections between the movement of particular body joints and the given overarching analysis task. In Chapter 5, benefited from the multiple-kernel representation of motion data and the proposed frameworks of Chapters (3 and 4), I have designed interpretable multiple-kernel models, which perform feature selection (or scaling) for motion data with different supervised or unsupervised objectives.

As the first MKL algorithm, I have extended the metric learning concept to the multiple-kernel representation, aiming to increase the local separation of the motion classes in the feature space. Specifically, my proposed LMMK framework employs a diagonal metric that allows us to perform metric learning as the scaling of the feature space. Furthermore, using a sparsity objective in this formulation leads to scaling a sparse set of dimensions in the RKHS, which can be interpreted as relevant motion dimensions to the given discriminative task. Based on my empirical evaluations on real-world mocap benchmarks, LMMK outperforms other multiple-kernel learning algorithms in terms of discriminative feature selection. Although its discriminative performance is comparable to the DTW-LMNN algorithm of Chapter 3, it results in a smaller set of relevant features.

As another part of Chapter 5, I have transferred the multiple-kernel learning problem to the prototype-based representation of motion data. More specifically, I have proposed a framework that provides an interpretable prototype-based representation of motion data in a combined RKHS. This model, which is specifically beneficial to practitioners and domain experts, learns interpretable motion prototypes as the local representatives of motion classes in RKHS and effectively discriminates the classes from each other in that space. The feature selection part of the IMKPL framework selects the base kernels that are relevant to the aimed prototype-based representation. My experimental evaluations have shown this model's superiority over other prototype-based alternatives in providing semantically interpretable and discriminative multiple-kernel prototypes. Although IMKPL is outperformed by LMMK based on discriminative feature selection performance, its highly interpretable prototype-based model is of substantial value to practitioners.

In the last part of Chapter 5, I have mitigated the typical real-world challenge of confronting unseen motion classes with respect to a given annotated mocap dataset. Accordingly, I have designed a novel multiple-kernel dictionary learning framework that learns semantic attributes, which rely on the exiting similarities and dissimilarities in particular body joints' movements in different motion classes. These attributes are particularly interpretable based on their connections to particular dimensions of known motion categories. Based on experimental evaluations, I have shown that the learned semantic attributes can be used for interpretable encoding of unseen motion classes. I have also proposed an incremental clustering method, which can efficiently cluster unseen motion sequences upon their partial or complete encoding results.

Another important focus of motion analysis is the segmentation of long motion sequences into understandable shorter temporal parts. This is of particular interest when motions are recorded as long streams in uncontrolled environments, making their manual annotation considerably time-consuming. Furthermore, a temporal analysis of motion data can reveal the motion time-series' relevant regions to the given classification task. In Chapter 6, to address that concern using a deep neural network, I have proposed a deep-aligned convolutional network that performs temporal segmentation and sequence labeling of skeleton motions. In this architecture, I have introduced the novel concept of temporal alignment filters for CNN models. By using such filters, the network can learn significant temporal patterns in long motions, which are semantically understandable while also discriminative in distinguishing different motion categories from each other. The empirical evaluation of my deep learning architecture demonstrates its competitive performance in the recognition and segmentation of skeleton-based human actions, while it also finds interpretable temporal prototypes for the given mocap dataset.

Limitations: The proposed methods in this thesis can still benefit from further improvement in several aspects.

First, all the algorithms in Chapters 3, 4, and 5 highly rely on having the motion data segmented in advance and synchronized according to the number of action repetitions per sequence. Although this requirement is easily provided in laboratory-based experiments, it may bring difficulties while capturing data from open-world and public environments.

Second, the proposed kernel-based or distance-based methods in Chapters 3, 4, and 5 require storing the training data for the prediction of test samples. Such a requirement can quadratically increase the spacious complexity of the algorithms, especially for large-scale datasets.

Third, despite the effective power of the proposed deep architecture in Chapter 6, similar to other deep neural networks, it requires a relatively large number of annotated mocap samples for its training phase. Such a prerequisite can limit its applicable benchmarks as the production and annotation of mocap databases is dramatically more time-consuming than other data forms such as videos or images.

Fourth, even though the proposed non-negative quadratic optimization algorithm in Section 4.3 has linear complexity (compared to the quadratic complexity of its alternatives), the whole sparse coding framework has a quadratic computational complexity (or cubic for high-dimensional data). Although this complexity is one degree smaller than their alternative methods, it can become prohibitive for large-scale high-dimensional datasets. Nevertheless, due to the large share of kernel pre-computation steps in the complexity of each algorithm, one general effective workaround can be using kernel linearization methods similar to (Golts and Michael Elad 2016)

Finally, besides the significant improvements of the proposed methods compared to their predecessors in terms of their interpretability, that property is still ill-posed regarding its definition in the field of machine learning. Also, despite the measure that I used to quantify it in the experimental sessions approximately, measuring interpretability as an absolute value is still an open challenge in this field of science. Therefore, it is most often not possible to compare the models from different works, which are claimed to be interpretable, w.r.t. that property. Furthermore, the concept of semantic and the way one should apply it to structured information such as motion data are not always well defined. Even in many cases, this concept is still ill-posed among practitioners and domain experts, which makes it subject to the person's point of view and prone to be changed over time and experience.

Outlook: Besides mathematically enhancing the specific algorithms proposed in this thesis, my work opens significant research possibilities from several scopes.

First, in different parts of this work, I showed how we could obtain interpretable models for motion data analysis from different perspectives and with different objectives. Nevertheless, the major part of this work is applicable to any structured or even vectorial data, given that a pairwise similarity-based relationship between data entities is available. Therefore, some of these methods are already tried on non-temporal datasets as reported in the relevant publications (Hosseini and Hammer 2018b; Hosseini and Hammer 2019c; Hosseini and Hammer 2018c; Hosseini and Hammer 2019a). This characteristic motivates us to employ the proposed methods in various applications. For instance, a practically usable data representation in health-cares systems highly relies on obtaining clinically understandable models, which depends on the interpretability and explainability of the model and its entities by a domain expert (Velikova et al. 2014; Lopez and Blobel 2008; Stiglic et al. 2020). Therefore the particular connections that my proposed methods find between the model's entities and semantically understandable input knowledge can be incorporated as the underlying building blocks of overarching large-scale applications to advance the current state of model interpretation in this domain. Considering interpretation as the quality of a machine learning model to explain its specific decisions (Molnar 2020), such characteristic is the necessary building block of many different traits such as fairness, privacy, reliability, causality, and trust, which have their own specific applications and methods (Doshi-Velez and B. Kim 2017). Therefore, the application of my proposed views toward interpretable modeling can be further investigated as a constituent early-stage building block of the solutions in those areas.

Second, in Chapter 3, I have focused on the efficiency of target selections for metric learning algorithms (specifically for LMNN). Even though this dimension of the problem generally converts it to an NP-hard problem, I have shown that we can incorporate geometrical analysis on the global data distribution to mitigate this issue. On the other hand, it has been shown that investigating local metrics or formulating learning in a multi-task framework can better approach multi-class problems (Kilian Q Weinberger and Lawrence K Saul 2008; Parameswaran and Kilian Q Weinberger 2010). Therefore, an interesting research line is to investigate further the target analysis path that I have opened up in more advanced variants of metric learning frameworks. Furthermore, my introduced component-wise view to the distance-based metric learning can be transferred to such frameworks as well.

Fourth, although the sparse coding frameworks in Chapter 4 work on the similarity-based relationship of data entities, one can investigate the proposed methods' extension to frame-based models. Inspired by methods similar to (Lichen Wang, Z. Ding, and Fu 2018; S. Li, K. Li, and Fu 2015; T. Zhou et al. 2020), it could be possible to incorporate the novelties of this thesis into such frameworks. The resulting model would perform frame-based processing of motion data while also providing interpretable characteristics regarding prototype-base representation, feature selection, interpretable encoding.

Fifth, in Chapter 6, I have demonstrated the positive effect of alignment kernels on the performance and interpretability of convolutional neural architectures for motion data analysis. Therefore, an interesting research line is to incorporate such form of filters in other more complex and more advanced deep neural networks which have convolutional modules as their building blocks (Núñez et al. 2018; Y. Tang et al. 2018; Ke et al. 2017; Sijie Yan, Xiong, and D. Lin 2018; C. Li et al. 2017). Those frameworks have shown notable performance in the classification of motion data. Therefore, it is expected that the proposed incorporation can improve their accuracy and interpretability while reducing their complexity. As a reason for this reduction, these filters process such specific input more effectively while using fewer parameters in comparison. Additionally, the introduced idea of the conditional extension of a neural network's depth during its training phase can be further investigating in other architectures such as LSTM and RCNN. However, appropriate trigger mechanisms should be employed to increase the network's complexity when required.

PUBLICATIONS IN THE CONTEXT OF THIS THESIS

Hosseini, Babak and Barbara Hammer (2015). "Efficient metric learning for the analysis of motion data". In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.

Hosseini, Babak, Felix Hülsmann, et al. (2016). "Non-negative kernel sparse coding for the analysis of motion data". In: *International Conference on Artificial Neural Networks (ICANN)*. Springer, pp. 506–514.

Hosseini, Babak and Barbara Hammer (2018a). "Confident kernel sparse coding and dictionary learning". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1031–1036.

— (2018b). "Feasibility Based Large Margin Nearest Neighbor Metric Learning". In: *26th European Symposium on Artificial Neural Networks (ESANN)*.

— (2018c). "Non-negative Local Sparse Coding for Subspace Clustering". In: *Advances in Intelligent Data Analysis XVII. (IDA)*. Ed. by Ukkonen A. Duivesteijn W. Siebes A. Vol. 11191. Lecture Notes in Computer Science. Springer, pp. 137–150. DOI: 10.1007/978-3-030-01768-2_12.

— (2019b). "Interpretable Multiple-Kernel Prototype Learning for Discriminative Representation and Feature Selection". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM.

— (2019c). "Large-Margin Multiple Kernel Learning for Discriminative Features Selection and Representation Learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

— (2019d). "Multiple-Kernel Dictionary Learning for Reconstruction and Clustering of Unseen Multivariate Time-series". In: *27th European Symposium on Artificial Neural Networks (ESANN)*.

Hosseini, Babak, Romain Montagne, and Barbara Hammer (2019). "Deep-Aligned Convolutional Neural Network for Skeleton-based Action Recognition and Segmentation". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE.

— (2020). "Deep-Aligned Convolutional Neural Network for Skeleton-Based Action Recognition and Segmentation (extended article)". In: *Data Science and Engineering*. ISSN: 2364-1541. URL: https://doi.org/10.1007/s41019-020-00123-3.

REFERENCES

Adistambha, K., C.H. Ritz, and I.S. Burnett (Oct. 2008). “Motion classification using Dynamic Time Warping”. In: *MMSP’08 Workshop*, pp. 622–627.

Afrasiabi, Mahlagha, Muharram Mansoorizadeh, et al. (2019). “DTW-CNN: time series-based human interaction prediction in videos using CNN-extracted features”. In: *The Visual Computer*, pp. 1–13.

Aggarwal, Jake K and Michael S Ryoo (2011). “Human activity analysis: A review”. In: *ACM Computing Surveys (CSUR)* 43.3, p. 16.

Aharon, M., M. Elad, and A. Bruckstein (2006). “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation”. In: *IEEE Transactions on Signal Processing* 54.11, pp. 4311–4322. ISSN: 1053587X.

Ahmed, Faisal, Padma Polash Paul, and Marina L Gavrilova (2015). “DTW-based kernel and rank-level fusion for 3D gait recognition using Kinect”. In: *The Visual Computer* 31.6-8, pp. 915–924.

Aiolli, Fabio and Michele Donini (2014). “Learning anisotropic RBF kernels”. In: *International Conference on Artificial Neural Networks*. Springer, pp. 515–522.

— (2015). “EasyMKL: a scalable multiple-kernel learning algorithm”. In: *Neurocomputing* 169, pp. 215–224.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). “Contextual string embeddings for sequence labeling”. In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Alabdulmohsin, Ibrahim, Moustapha Cisse, and Xiangliang Zhang (2016). “Is Attribute-Based Zero-Shot Learning an Ill-Posed Strategy?” In: *ECML/PKDD’16*. Springer, pp. 749–760.

Alelyani, Salem, Jiliang Tang, and Huan Liu (2013). “Feature selection for clustering: a review.” In: *Data clustering: algorithms and applications* 29.1.

Alpaydin, Ethem (2020). *Introduction to machine learning*. Cambridge, Mass.: MIT Press.

Althloothi, Salah et al. (2014). “Human activity recognition using multi-features and multiple-kernel learning”. In: *Pattern recognition* 47.5, pp. 1800–1812.

Alzaidy, Rabah, Cornelia Caragea, and C Lee Giles (2019). “Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents”. In: *The world wide web conference*, pp. 2551–2557.

Ana, LNF and Anil K Jain (2003). “Robust data clustering”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE, pp. II–128.

Anada, Satoshi et al. (2019). “Sparse coding and dictionary learning for electron hologram denoising”. In: *Ultramicroscopy* 206, p. 112818.

Anagnostopoulos, Aris et al. (2006). “Global distance-based segmentation of trajectories”. In: *Proceedings of SIGKDD’06*. ACM, pp. 34–43.

Anguita, Davide et al. (2012). “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine”. In: *International workshop on ambient assisted living*. Springer, pp. 216–223.

Arami, Arash et al. (2019). “Prediction of gait freezing in Parkinsonian patients: a binary classification augmented with time series prediction”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.9, pp. 1909–1919.

Arikan, Okan and David A Forsyth (2002). “Interactive motion generation from examples”. In: *ACM Transactions on Graphics (TOG)* 21.3, pp. 483–490.

Arlt, W. et al. (2011). “Urine steroid metabolomics as a biomarker tool for detecting malignancy in adrenal tumors”. In: *J Clinical Endocrinology and Metabolism* 96, pp. 3775–3784.

Baak, Andreas et al. (2013). “A data-driven approach for real-time full body pose reconstruction from a depth camera”. In: *Consumer Depth Cameras for Computer Vision*. Springer, pp. 71–98.

Bach, F et al. (2008). “Learning discriminative dictionaries for local image analysis”. In: CVPR.

Bach, Francis R, Gert RG Lanckriet, and Michael I Jordan (2004). “Multiple kernel learning, conic duality, and the SMO algorithm”. In: *ICML'04*.

Backhaus, Andreas and Udo Seiffert (2014). “Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size”. In: *Neurocomputing* 131, pp. 15–22.

Bahlmann, Claus, Bernard Haasdonk, and Hans Burkhardt (2002). “Online handwriting recognition with support vector machines-a kernel approach”. In: *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*. IEEE, pp. 49–54.

Bahrampour, Soheil et al. (2015). “Kernel task-driven dictionary learning for hyperspectral image classification”. In: *ICASSP'15*. IEEE, pp. 1324–1328.

Basharat, Arslan and Mubarak Shah (2009). “Time series prediction by chaotic modeling of nonlinear dynamical systems”. In: *2009 IEEE 12th international conference on computer vision*. IEEE, pp. 1941–1948.

Batista, Gustavo EAPA et al. (2014). “CID: an efficient complexity-invariant distance for time series”. In: *Data Mining and Knowledge Discovery* 28.3, pp. 634–669.

Beck, Amir and Marc Teboulle (2009). “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *Society* 2.1, pp. 183–202.

Beggs, Joseph Stiles (1983). *Kinematics*. CRC Press.

Bellet, Aurélien, Amaury Habrard, and Marc Sebban (2013). “A survey on metric learning for feature vectors and structured data”. In: *arXiv preprint arXiv:1306.6709*.

Bergroth, Lasse, Harri Hakonen, and Timo Raita (2000). “A survey of longest common subsequence algorithms”. In: *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*. IEEE, pp. 39–48.

Bernard, M. et al. (2008). “Learning probabilistic models of tree edit distance”. In: *Pattern Recognition* 41.8, pp. 2611–2629. ISSN: 00313203. DOI: 10.1016/j.patcog.2008.01.011.

Berndt, Donald J and James Clifford (1994). “Using dynamic time warping to find patterns in time series.” In: *KDD workshop*. Vol. 10. 16. Seattle, WA, pp. 359–370.

Bian, Xiao, Feng Li, and Xia Ning (2016). “Kernelized Sparse Self-Representation for Clustering and Recommendation”. In: *SIAM International Conference on Data Mining*, pp. 10–17.

Biehl, Michael, Kerstin Bunte, and Petra Schneider (2013). “Analysis of flow cytometry data by matrix relevance learning vector quantization”. In: *PLoS One* 8.3, e59401.

Bien, Jacob, Robert Tibshirani, et al. (2011). “Prototype selection for interpretable classification”. In: *The Annals of Applied Statistics* 5.4, pp. 2403–2424.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.

Bodor, Robert et al. (2009). “View-independent human motion classification using image-based reconstruction”. In: *Image and Vision Computing* 27.8, pp. 1194–1206.

Bortolini, Marco et al. (2020). “Motion Analysis System (MAS) for production and ergonomics assessment in the manufacturing processes”. In: *Computers & Industrial Engineering* 139, p. 105485.

Boyd, Stephen et al. (2011). "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1, pp. 1–122.

Brand, Matthew and Donghui Chen (2011). "Parallel quadratic programming for image processing". In: *2011 18th IEEE International Conference on Image Processing*. IEEE, pp. 2261–2264.

Bregler, Christoph (2014). "Kinematic Motion Models". In: *Computer Vision, A Reference Guide*, pp. 437–440.

Bro, Rasmus and Sijmen De Jong (1997). "A fast non-negativity-constrained least squares algorithm". In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 11.5, pp. 393–401.

Bunte, Kerstin et al. (2012). "Limited Rank Matrix Learning, discriminative dimension reduction and visualization". In: *Neural Networks* 26, pp. 159–173. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2011.10.001. URL: http://www.sciencedirect.com/science/article/pii/S0893608011002632.

Butepage, Judith et al. (2017). "Deep representation learning for human motion prediction and classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6158–6166.

Cai, Hongyun, Vincent W Zheng, and Kevin Chen-Chuan Chang (2018). "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9, pp. 1616–1637.

Cai, Jian-Feng, Emmanuel J Candès, and Zuowei Shen (2010). "A singular value thresholding algorithm for matrix completion". In: *SIAM Journal on Optimization* 20.4, pp. 1956–1982.

Camps-Valls, Gustavo and Lorenzo Bruzzone (2005). "Kernel-based methods for hyperspectral image classification". In: *IEEE TGRS* 43.6, pp. 1351–1362.

Cao, Dongwei et al. (2004). "Online motion classification using support vector machines". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 3. IEEE, pp. 2291–2296.

Chambers, Graeme S et al. (2004). "Segmentation of intentional human gestures for sports video annotation". In: *Multimedia Modelling Conference, 2004. Proceedings. 10th International*. IEEE, pp. 124–129.

Chan, Kai-Chi, Cheng-Kok Koh, and CS George Lee (2014). "A 3-D-point-cloud system for human-pose estimation". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44.11, pp. 1486–1497.

Chang, Chih-Chung and Chih-Jen Lin (2011). "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3, p. 27.

Chang, Ju Yong (2014). "Nonparametric gesture labeling from multi-modal data". In: *Workshop at ECCV'14*. Springer, pp. 503–517.

Changpinyo, Soravit, Mark Sandler, and Andrey Zhmoginov (2017). "The power of sparsity in convolutional neural networks". In: *arXiv preprint arXiv:1702.06257*.

Chen, Cheng et al. (2010). "Learning a 3D human pose distance metric from geometric pose descriptor". In: *IEEE Transactions on Visualization and Computer Graphics* 17.11, pp. 1676–1689.

Chen, Lei and Raymond Ng (2004). "On the marriage of lp-norms and edit distance". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 792–803.

Chen, Tianshui et al. (2018). "Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding". In: *Proceedings of the 26th ACM international conference on Multimedia*, pp. 2023–2031.

Chen, Tsu-Wei, Meena Abdelmaseeh, and Daniel Stashuk (2015). "Affine and regional dynamic time warping". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, pp. 440–448.

Chen, Z. et al. (2015). "Kernel sparse representation for time series classification". In: *Information Sciences* 292, pp. 15–26.

Cheng, Heng-Tze et al. (2013). "Nuactiv: Recognizing unseen new activities using semantic attribute-based learning". In: *MobiSys'13*. ACM, pp. 361–374.

Cho, KyungHyun (2013). "Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images". In: *International Conference on Machine Learning*, pp. 432–440.

Cho, Kyunghyun and Xi Chen (2014). "Classifying and visualizing motion capture sequences using deep neural networks". In: *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*. Vol. 2. IEEE, pp. 122–130.

Choi, Jeong et al. (2019). "Zero-shot learning for audio-based music classification and tagging". In: *arXiv preprint arXiv:1907.02670*.

Chollet, François (2017). "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289*.

CMU. (Mar. 2007). "Carnegie Mellon University Graphics Lab: Motion Cap- ture Database". In: URL: http://mocap.cs.cmu.edu.

Coelho, David N and Guilherme A Barreto (2019). "Approximate Linear Dependence as a Design Method for Kernel Prototype-Based Classifiers". In: *International Workshop on Self-Organizing Maps*. Springer, pp. 241–250.

Connie, Tee et al. (2017). "Facial expression recognition using a hybrid CNN–SIFT aggregator". In: *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*. Springer, pp. 139–149.

Cortes, Corinna and Vladimir Vapnik (Sept. 1995). "Support-vector networks". In: *Machine Learning* 20.3, pp. 273–297. DOI: 10.1007/bf00994018.

Coskun, Huseyin et al. (2018). "Human motion analysis with deep metric learning". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 667–683.

Crammer, Koby and Yoram Singer (2001). "On the algorithmic implementation of multiclass kernel-based vector machines". In: *Journal of machine learning research* 2.Dec, pp. 265–292.

Cristianini, Nello, John Shawe-Taylor, et al. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

Cuturi, Marco et al. (2007). "A kernel for time series based on global alignments". In: *ICASSP 2007*. Vol. 2. IEEE, pp. II–413.

Das, Santanu et al. (2010). "Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 47–56.

Dash, Manoranjan and Huan Liu (1997). "Feature selection for classification". In: *Intelligent data analysis* 1.3, pp. 131–156.

Demmel, James, Ioana Dumitriu, and Olga Holtz (2007). "Fast linear algebra is stable". In: *Numerische Mathematik* 108.1, pp. 59–91.

Demuth, Bastian et al. (2006). "An Information Retrieval System for Motion Capture Data". In: *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*, pp. 373–384.

Deri, Luca, Simone Mainardi, and Francesco Fusco (2012). "tsdb: A compressed database for time series". In: *International Workshop on Traffic Monitoring and Analysis*. Springer, pp. 143–156.

Dernbach, Stefan et al. (2012). "Simple and complex activity recognition through smart phones". In: *2012 eighth international conference on intelligent environments*. IEEE, pp. 214–221.

Dileep, Aroor Dinesh and C Chandra Sekhar (2009). "Representation and feature selection using multiple-kernel learning". In: *IJCNN 2009*. IEEE, pp. 717–722.

Ding, Wenwen et al. (2018). "Tensor-based linear dynamical systems for action recognition from 3D skeletons". In: *Pattern Recognition* 77, pp. 75–86.

Doshi-Velez, Finale and Been Kim (2017). "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608*.

Drimus, Alin et al. (2014). "Design of a flexible tactile sensor for classification of rigid and deformable objects". In: *Robotics and Autonomous Systems* 62.1, pp. 3–15.

Du, Peijun et al. (2017). "Multiple composite kernel learning for hyperspectral image classification". In: *Geoscience and Remote Sensing Symposium (IGARSS), 2017 IEEE International*. IEEE, pp. 2223–2226.

Du, Yong, Yun Fu, and Liang Wang (2015). "Skeleton based action recognition with convolutional neural network". In: *ACPR'15*. IEEE, pp. 579–583.

Du, Yong, Wei Wang, and Liang Wang (2015). "Hierarchical recurrent neural network for skeleton based action recognition". In: *Proceedings of CVPR'15*, pp. 1110–1118.

Duarte-Carvajalino, Julio Martin and Guillermo Sapiro (2009). "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization". In: *IEEE Transactions on Image Processing* 18.7, pp. 1395–1408.

Duda, Richard O and Peter E Hart (1973). "Pattern classification and scene analysis". In: *A Wiley-Interscience Publication, New York: Wiley, 1973*.

Dumoulin, Vincent and Francesco Visin (2016). "A guide to convolution arithmetic for deep learning". In: *arXiv preprint arXiv:1603.07285*.

Durantin, Gautier, Scott Heath, and Janet Wiles (2017). "Social moments: a perspective on interaction for social robotics". In: *Frontiers in Robotics and AI* 4, p. 24.

Elad, Michael and Michal Aharon (2006). "Image denoising via sparse and redundant representations over learned dictionaries". In: *IEEE Transactions on Image processing* 15.12, pp. 3736–3745.

Elert, Glenn (1998). "The physics hypertextbook". In: *Found July* 9, p. 2008.

Elhamifar, Ehsan and Rene Vidal (2013). "Sparse subspace clustering: Algorithm, theory, and applications". In: *IEEE transactions on pattern analysis and machine intelligence* 35.11, pp. 2765–2781.

Escalera, Sergio et al. (2014). "Chalearn looking at people challenge 2014: Dataset and results". In: *Workshop at ECCV'14*. Springer, pp. 459–473.

Fawaz, Hassan Ismail et al. (2019). "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4, pp. 917–963.

Ferdinands, R (2010). "Advanced applications of motion analysis in sports biomechanics". In: *ISBS-Conference Proceedings Archive*.

Finlay, Janet (1997). "Machine learning: A tool to support usability?" In: *Applied Artificial Intelligence* 11.7-8, pp. 633–651.

Fong, Ruth C and Andrea Vedaldi (2017). "Interpretable explanations of black boxes by meaningful perturbation". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437.

Frénay, Benoît et al. (2014). "Valid interpretation of feature relevance for linear data mappings". In: *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, pp. 149–156.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York.

Gan, Le et al. (2018). "Multiple feature kernel sparse representation classifier for hyperspectral imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.9, pp. 5343–5356.

Gao, Shenghua, Ivor Wai-Hung Tsang, and Liang-Tien Chia (2010). "Kernel sparse representation for image classification and face recognition". In: *European Conference on Computer Vision*. Springer, pp. 1–14.

Gao, Shenghua, Ivor Wai-hung Tsang, and Liang-tien Chia (2012). "Laplacian Sparse Coding , Hypergraph Laplacian Sparse Coding , and Applications ". In: *IEEE TPAMI* 35.October, pp. 92–104. ISSN: 1939-3539.

Gates, Allison et al. (2019). "Performance and usability of machine learning for screening in systematic reviews: a comparative evaluation of three tools". In: *Systematic reviews* 8.1, p. 278.

Gehring, Jonas et al. (2017). "Convolutional Sequence to Sequence Learning". In: *Proceedings of ICML'17*, pp. 1243–1252.

Georghiades, Athinodoros S., Peter N. Belhumeur, and David J. Kriegman (2001). "From few to many: Illumination cone models for face recognition under variable lighting and pose". In: *IEEE transactions on pattern analysis and machine intelligence* 23.6, pp. 643–660.

Ghanem, Bernard and Narendra Ahuja (2010). "Maximum margin distance learning for dynamic texture recognition". In: *ECCV'10*. Springer, pp. 223–236.

Gillies, Marco et al. (2016). "Human-centred machine learning". In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 3558–3565.

Girshick, Ross (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Glardon, Pascal, Ronan Boulic, and Daniel Thalmann (2004). "PCA-based walking engine using motion capture data". In: *Proceedings Computer Graphics International, 2004.* IEEE, pp. 292–298.

Goldberger, Jacob et al. (2005). "Neighbourhood components analysis". In: *Advances in neural information processing systems*, pp. 513–520.

Golts, Alona and Michael Elad (2016). "Linearized kernel dictionary learning". In: *IEEE Journal of Selected Topics in Signal Processing* 10.4, pp. 726–739.

Gönen, Mehmet and Ethem Alpaydın (2011). "Multiple kernel learning algorithms". In: *JMLR* 12.Jul, pp. 2211–2268.

Göpfert, Christina, Benjamin Paassen, and Barbara Hammer (2016). "Convergence of Multi-pass Large Margin Nearest Neighbor Metric Learning". In: *International Conference on Artificial Neural Networks*. Springer, pp. 510–517.

Grant, Michael, Stephen Boyd, and Yinyu Ye (2008). *CVX: Matlab software for disciplined convex programming.*

Gross, Ralph et al. (2010). "Multi-pie". In: *Image and Vision Computing* 28.5, pp. 807–813.

Gu, Yanfeng, Jocelyn Chanussot, et al. (2017). "Multiple kernel learning for hyperspectral image classification: A review". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.11, pp. 6547–6565.

Gu, Yanfeng, Guoming Gao, et al. (2014). “Model selection and classification with multiple-kernel learning for hyperspectral images via sparsity”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.6, pp. 2119–2130.

Gu, Yanfeng, Chen Wang, et al. (2012). “Representative multiple-kernel learning for classification in hyperspectral imagery”. In: *IEEE Transactions on Geoscience and Remote Sensing* 50.7, pp. 2852–2865.

Gu, Yanfeng, Qingwang Wang, et al. (2015). “Multiple kernel learning via low-rank nonnegative matrix factorization for classification of hyperspectral imagery”. In: *IEEE J-STARS* 8.6, pp. 2739–2751.

Guan, Renchu et al. (2011). “Text clustering with seeds affinity propagation”. In: *Knowledge and Data Engineering, IEEE Transactions on* 23.4, pp. 627–637.

Guerra-Filho, Gutemberg (2005). “Optical Motion Capture: Theory and Implementation.” In: *RITA* 12.2, pp. 61–90.

Guillaumin, Matthieu, Jakob Verbeek, and Cordelia Schmid (2009). “Is that you? Metric learning approaches for face identification”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, pp. 498–505.

Guo, Junqi et al. (2016). “Smartphone-based patients’ activity recognition by using a self-learning scheme for medical monitoring”. In: *Journal of medical systems* 40.6, p. 140.

Guo, Yi, Junbin Gao, and Feng Li (2013). “Spatial subspace clustering for hyperspectral data segmentation”. In: *Conference of The Society of Digital Information and Wireless Communications (SDIWC)*. Vol. 1. 2, p. 3.

Hachaj, Tomasz and Marek R Ogiela (2014). “Rule-based approach to recognizing human body poses and gestures in real time”. In: *Multimedia Systems* 20.1, pp. 81–99.

Hammer, Barbara, Alexander Hasenfuss, et al. (2007). “Intuitive clustering of biological data”. In: *2007 International Joint Conference on Neural Networks*. IEEE, pp. 1877–1882.

Hammer, Barbara, Daniela Hofmann, et al. (2014). “Learning vector quantization for (dis-) similarities”. In: *Neurocomputing* 131, pp. 43–51.

Harandi, Mehrtash et al. (2013). “Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution”. In: *ICCV’13*. IEEE, pp. 3120–3127.

Harris, Gerald F and Peter A Smith (2000). *Pediatric gait: a new millennium in clinical care and motion analysis technology*. Institute of Electrical & Electronics Engineers (IEEE).

Hauser, Kris et al. (2008). “Using motion primitives in probabilistic sample-based planning for humanoid robots”. In: *Algorithmic foundation of robotics VII*. Springer, pp. 507–522.

Hazan, Tamir, Simon Polak, and Amnon Shashua (2005). “Sparse image coding using a 3D non-negative tensor factorization”. In: *ICCV’05*. IEEE, pp. 50–57.

He, Kaiming et al. (2015). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

— (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Zhenyu (2010). “Activity recognition from accelerometer signals based on wavelet-ar model”. In: *2010 IEEE International Conference on Progress in Informatics and Computing*. Vol. 1. IEEE, pp. 499–502.

Hirst, Graeme (1992). *Semantic interpretation and the resolution of ambiguity*. Cambridge University Press.

Hofmann, Daniela et al. (2014). “Learning interpretable kernelized prototype-based models”. In: *Neurocomputing* 141, pp. 84–96.

Hosseini, Babak and Barbara Hammer (2019a). "Interpretable discriminative dimensionality reduction and feature selection on the manifold". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 310–326.

Hougardy, Stefan (2010). "The Floyd–Warshall algorithm on graphs with negative cycles". In: *Information Processing Letters* 110.8-9, pp. 279–281.

Hu, Jian-Fang, Wei-Shi Zheng, Jianhuang Lai, et al. (2015). "Jointly learning heterogeneous features for RGB-D activity recognition". In: *Proceedings of CVPR'15*, pp. 5344–5352.

Hu, Jian-Fang, Wei-Shi Zheng, Lianyang Ma, et al. (2016). "Real-time RGB-D activity prediction by soft regression". In: *ECCV'16*. Springer, pp. 280–296.

Hua, Yuxiu et al. (2019). "Deep learning with long short-term memory for time series prediction". In: *IEEE Communications Magazine* 57.6, pp. 114–119.

Huang, Ke and Selin Aviyente (2007). "Sparse representation for signal classification". In: *NIPS'07*, pp. 609–616.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). "Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991*.

Huang, Zhiwu et al. (2017). "Deep learning on lie groups for skeleton-based action recognition". In: *Proceedings of CVPR'17*. IEEE computer Society, pp. 1243–1252.

Hueter-Becker, Antje and Mechthild Doelken (2014). *Physical Therapy Examination and Assessment*. Thieme.

Hussein, Mohamed E et al. (2013). "Human Action Recognition Using a Temporal Hierarchy of Covariance Descriptors on 3D Joint Locations." In: *IJCAI'13*. Vol. 13, pp. 2466–2472.

Ikizler-Cinbis, Nazli and Stan Sclaroff (2010). "Object, scene and actions: Combining multiple features for human action recognition". In: *European conference on computer vision*. Springer, pp. 494–507.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167*.

Jalal, Ahmad, Majid Ali Khan Quaid, and Kibum Kim (2019). "A wrist worn acceleration based human motion analysis and classification for ambient smart home system". In: *Journal of Electrical Engineering & Technology* 14.4, pp. 1733–1739.

Janet, H Carr and B Shepherd Roberta (2003). *Stroke Stroke Rehabilitation Guidelil1es for Exercise and Training to Optimize Motor Skill*. Butterworth-Heinemann.

Jenatton, Rodolphe et al. (2010). "Proximal Methods for Sparse Hierarchical Dictionary Learning." In: *ICML'10*. 2010. Citeseer, pp. 487–494.

Ji, Cun et al. (2019). "A fast shapelet selection algorithm for time series classification". In: *Computer Networks* 148, pp. 231–240.

Ji, Zhong et al. (2019). "Query-aware sparse coding for web multi-video summarization". In: *Information Sciences* 478, pp. 152–166.

Jia, Lei et al. (2016). "Adaptive neighborhood propagation by joint L2, 1-norm regularized sparse coding for representation and classification". In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, pp. 201–210.

Jiang, Wenhao and Fu-lai Chung (2014). "A trace ratio maximization approach to multiple-kernel-based dimensionality reduction". In: *Neural Networks* 49, pp. 96–106.

Jiang, Zhuolin, Zhe Lin, and Larry S Davis (2013). "Label consistent K-SVD: Learning a discriminative dictionary for recognition". In: *IEEE TPAMI* 35.11, pp. 2651–2664.

Johnson, Charles R and Herbert A Robinson (1981). "Eigenvalue inequalities for principal submatrices". In: *Linear Algebra and its Applications* 37, pp. 11–22.

Jolliffe, Ian (2002). *Principal component analysis*. Wiley Online Library.

Joon, Jong Sze (Aug. 2010). "Reviewing Principles and Elements of Animation for Motion Capture-Based Walk, Run and Jump". In: *Computer Graphics, Imaging and Visualization*

(CGIV), 2010 Seventh International Conference on, pp. 55–59. DOI: 10.1109/CGIV.2010.16.

Kandola, Jaz, Nello Cristianini, and John Shawe-taylor (2002). "Learning semantic similarity". In: *Advances in neural information processing systems* 15, pp. 673–680.

Kapadia, Mubbasir et al. (2013). "Efficient motion retrieval in large motion databases". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 19–28.

Karpathy, Andrej et al. (2014). "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732.

Ke, Qiuhong et al. (2017). "A new representation of skeleton sequences for 3d action recognition". In: *CVPR'17*. IEEE, pp. 4570–4579.

Keogh, Eamonn, Themistoklis Palpanas, et al. (2004). "Indexing large human-motion databases". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 780–791.

Keogh, Eamonn and Chotirat Ann Ratanamahatana (2005). "Exact indexing of dynamic time warping". In: *Knowledge and information systems* 7.3, pp. 358–386.

Keogh, Eamonn, Li Wei, et al. (2009). "Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures". In: *The VLDB journal* 18.3, pp. 611–630.

Keogh, Eamonn J and Michael J Pazzani (2001). "Derivative Dynamic Time Warping." In: *SDM*. Vol. 1. SIAM, pp. 5–7.

Keskin, Cem, Ali Taylan Cemgil, and Lale Akarun (2011). "DTW based clustering to improve hand gesture recognition". In: *International Workshop on Human Behavior Understanding*. Springer, pp. 72–81.

Kim, Mijung et al. (Dec. 2019). *Storing time series data for a search query*. US Patent 10,503,732.

Kim, Seung-Jean, Alessandro Magnani, and Stephen Boyd (2006). "Optimal kernel selection in kernel fisher discriminant analysis". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 465–472.

Kim, Taehwan, Gregory Shakhnarovich, and Raquel Urtasun (2010). "Sparse coding for learning interpretable spatio-temporal primitives". In: *Advances in neural information processing systems*, pp. 1117–1125.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kirrinnis, Peter (2001). "Fast algorithms for the Sylvester equation AX- XBT= C". In: *Theoretical Computer Science* 259.1-2, pp. 623–638.

Ko, M. H. et al. (2005). "Online context recognition in multisensor systems using dynamic time warping". In: *ISSNIP'05*. IEEE, pp. 283–288.

Kohonen, Teuvo (1995). "Learning vector quantization". In: *Self-organizing maps*. Springer, pp. 175–189.

Kolouri, Soheil et al. (2017). "Joint dictionaries for zero-shot learning". In: *arXiv preprint arXiv:1709.03688*.

Kong, Chen and Simon Lucey (2019). "Deep non-rigid structure from motion". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1558–1567.

Kong, Shu and Donghui Wang (2012). "A dictionary learning approach for classification: separating the particularity and the commonality". In: *European Conference on Computer Vision*. Springer, pp. 186–199.

Koppula, Hema S and Ashutosh Saxena (2015). “Anticipating human activities using object affordances for reactive robotic response”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1, pp. 14–29.

Kovar, Lucas, Michael Gleicher, and Frédéric Pighin (2008). “Motion graphs”. In: *ACM SIGGRAPH 2008 classes*, pp. 1–10.

Kratzer, Philipp, Marc Toussaint, and Jim Mainprice (2018). “Towards combining motion optimization and data driven dynamical models for human motion prediction”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 202–208.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2017). “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6, pp. 84–90.

Krüger, Björn et al. (2017). “Efficient unsupervised temporal segmentation of motion data”. In: *IEEE TMM* 19.4, pp. 797–812.

Krüger, Volker et al. (2007). “The meaning of action: A review on action recognition and mapping”. In: *Advanced robotics* 21.13, pp. 1473–1501.

Kuehne, Hildegard et al. (2011). “HMDB: a large video database for human motion recognition”. In: *2011 International Conference on Computer Vision*. IEEE, pp. 2556–2563.

Kulić, Dana et al. (2012). “Incremental learning of full body motion primitives and their sequencing through human motion observation”. In: *The International Journal of Robotics Research* 31.3, pp. 330–345.

Kulis, Brian (2012). “Metric learning: A survey”. In: *Foundations and Trends in Machine Learning* 5.4, pp. 287–364.

Kumar, Vipin and Sonajharia Minz (2014). “Feature selection: a literature review”. In: *SmartCR* 4.3, pp. 211–229.

Kuo, C-C Jay (2016). “Understanding convolutional neural networks with a mathematical model”. In: *Journal of Visual Communication and Image Representation* 41, pp. 406–413.

Kuo, C-C Jay et al. (2019). “Interpretable convolutional neural networks via feedforward design”. In: *Journal of Visual Communication and Image Representation* 60, pp. 346–359.

Kusakunniran, Worapan et al. (2010). “Support vector regression for multi-view gait recognition based on local motion feature selection”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 974–981.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of ICML'01*, pp. 282–289. ISBN: 1-55860-778-1.

Lampert, Christoph H, Hannes Nickisch, and Stefan Harmeling (2009). “Learning to detect unseen object classes by between-class attribute transfer”. In: *CVPR'09*. IEEE, pp. 951–958.

Lample, Guillaume et al. (2016). “Neural Architectures for Named Entity Recognition”. In: *Proceedings of NAACL-HLT*, pp. 260–270.

Latombe, Jean-Claude (2012). *Robot motion planning*. Vol. 124. Springer Science & Business Media.

Lea, Colin et al. (2016). “Temporal convolutional networks: A unified approach to action segmentation”. In: *European Conference on Computer Vision*. Springer, pp. 47–54.

LeCun, Yann et al. (1989). “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4, pp. 541–551.

Lee, Daniel D and H Sebastian Seung (2001). “Algorithms for non-negative matrix factorization”. In: *Advances in neural information processing systems*, pp. 556–562.

Lee, H. et al. (2006). “Efficient sparse coding algorithms”. In: *Advances in neural information processing systems*, pp. 801–808.

Letham, Benjamin, Cynthia Rudin, and David Madigan (2013). “Sequential event prediction”. In: *Machine learning* 93.2-3, pp. 357–380.
Li, Ao et al. (2018). “Self-supervised sparse coding scheme for image classification based on low rank representation”. In: *PloS one* 13.6, e0199141.
Li, Chuankun et al. (2017). “Skeleton-based action recognition using LSTM and CNN”. In: *ICMEW'17*. IEEE, pp. 585–590.
Li, Chun-Guang, Chong You, and René Vidal (2017). “Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework”. In: *IEEE Transactions on Image Processing* 26.6, pp. 2988–3001.
Li, Ning et al. (2018). “Deep joint semantic-embedding hashing.” In: *IJCAI*, pp. 2397–2403.
Li, Sheng, Kang Li, and Yun Fu (2015). “Temporal subspace clustering for human motion segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4453–4461.
Li, Xuelong, Guosheng Cui, and Yongsheng Dong (2017). “Graph regularized non-negative low-rank matrix factorization for image clustering”. In: *IEEE transactions on cybernetics* 47.11, pp. 3840–3853.
Li, Yang and Tao Yang (2018). “Word embedding for understanding natural language: a survey”. In: *Guide to Big Data Applications*. Springer, pp. 83–104.
Li, Yifeng and Alioune Ngom (2012). “A new kernel non-negative matrix factorization and its application in microarray data analysis”. In: *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, pp. 371–378.
Liang, Phyllis et al. (2020). “An Asian-centric human movement database capturing activities of daily living”. In: *Scientific Data* 7.1, pp. 1–13.
Lin, Chih-Jen (2007). “Projected gradient methods for nonnegative matrix factorization”. In: *Neural computation* 19.10, pp. 2756–2779.
Lin, Yen-Yu, Tyng-Luh Liu, and Chiou-Shann Fuh (2011). “Multiple kernel learning for dimensionality reduction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.6, pp. 1147–1160.
Lines, Jason and Anthony Bagnall (2014). “Ensembles of elastic distance measures for time series classification”. In: *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, pp. 524–532.
Liu, Feng et al. (2003). “3D motion retrieval with motion index tree”. In: *Computer Vision and Image Understanding* 92.2-3, pp. 265–284.
Liu, Guangcan et al. (2013). “Robust recovery of subspace structures by low-rank representation”. In: *IEEE TPAMI* 35.1, pp. 171–184.
Liu, Guodong and Leonard McMillan (2006). “Segment-based human motion compression”. In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 127–135.
Liu, Huaping, Di Guo, and Fuchun Sun (2016). “Object recognition using tactile measurements: Kernel sparse coding methods”. In: *IEEE Transactions on Instrumentation and Measurement* 65.3, pp. 656–665.
Liu, Jun, Amir Shahroudy, et al. (2018). “Skeleton-based action recognition using spatio-temporal LSTM network with trust gates”. In: *IEEE TPAMI* 40.12, pp. 3007–3021.
Liu, Jun, Gang Wang, et al. (2017). “Global context-aware attention lstm networks for 3d action recognition”. In: *CVPR'17*. Vol. 7, p. 43.
Liu, Mengyuan, Hong Liu, and Chen Chen (2017). “Enhanced skeleton visualization for view invariant human action recognition”. In: *Pattern Recognition* 68, pp. 346–362.
Liu, Qiang, Edmond C Prakash, et al. (2003). “The parameterization of joint rotation with the unit quaternion”. In: *DICTA*.

Liu, Tianzhu et al. (2016). “Class-specific sparse multiple-kernel learning for spectral–spatial hyperspectral image classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* 54.12, pp. 7351–7365.

Liu, Ting et al. (2005). “An investigation of practical approximate nearest neighbor algorithms”. In: *Advances in neural information processing systems*, pp. 825–832.

Liu, Weiyang et al. (2015). “Joint kernel dictionary and classifier learning for sparse coding via locality preserving K-SVD”. In: *Multimedia and Expo (ICME'15)*. IEEE, pp. 1–6.

Liu, Yebin et al. (2013). “Markerless Motion Capture of Multiple Characters Using Multiview Image Segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.11, pp. 2720–2735. DOI: 10.1109/TPAMI.2013.47. URL: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.47.

Lofberg, J (n.d.). “A toolbox for modeling and optimization in MATLAB”. In: *Proc. of the CACSD Conf.*

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). “Fully convolutional networks for semantic segmentation”. In: *Proceedings of CVPR'15*, pp. 3431–3440.

Long, Yang et al. (2018). “Towards Affordable Semantic Searching: Zero-Shot Retrieval via Dominant Attributes.” In: *AAAI*, pp. 7210–7217.

Lopez, Diego M and BGME Blobel (2008). “Enhanced semantic interpretability by healthcare standards profiling”. In: *Studies in health technology and informatics* 136, p. 735.

Lord, Phillip W. et al. (2003). “Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation”. In: *Bioinformatics* 19.10, pp. 1275–1283.

Lu, ChunMei and Nicola J Ferrier (2004). “Repetitive motion analysis: Segmentation and event classification”. In: *IEEE transactions on pattern analysis and machine intelligence* 26.2, pp. 258–263.

Lu, Di, Junqi Guo, and Xi Zhou (2016). “Self-learning Based Motion Recognition Using Sensors Embedded in a Smartphone for Mobile Healthcare”. In: *WASA'16*. Springer, pp. 343–355.

Lu, Tung-Wu and Chu-Fen Chang (2012). “Biomechanics of human movement and its clinical applications”. In: *The Kaohsiung journal of medical sciences* 28, S13–S25.

Lu, Xiaoqiang et al. (2012). “Geometry constrained sparse coding for single image super-resolution”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1648–1655.

Lun, Roanna and Wenbing Zhao (2015). “A survey of applications and human motion recognition with microsoft kinect”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 29.05, p. 1555008.

Ma, Xuezhe and Eduard Hovy (2016). “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: *Proceedings of ACL'16*. Vol. 1, pp. 1064–1074.

Ma, Yunqian and Yun Fu (2011). *Manifold learning theory and applications*. CRC press.

Maaten, L.J.P. van der and G.E. Hinton (2008). “Visualizing High-Dimensional Data Using t-SNE”. In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

Mairal, J, F Bach, and J Ponce (2012). “Task-driven dictionary learning”. In: *IEEE TPAMI* 34.4, pp. 791–804.

Mairal, Julien, Francis Bach, Jean Ponce, and Guillermo Sapiro (2009). “Online dictionary learning for sparse coding”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 689–696.

Mairal, Julien, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman (2008). “Discriminative learned dictionaries for local image analysis”. In: *CVPR'08*. IEEE, pp. 1–8.

Mairal, Julien, Jean Ponce, et al. (2009). "Supervised dictionary learning". In: *NIPS'09*, pp. 1033–1040.

Mandery, Christian et al. (2015). "The KIT whole-body human motion database". In: *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, pp. 329–336.

Marteau, Pierre-François (2008). "Time warp edit distance with stiffness adjustment for time series matching". In: *IEEE transactions on pattern analysis and machine intelligence* 31.2, pp. 306–318.

Martín, Henar et al. (2013). "Activity logging using lightweight classification techniques in mobile devices". In: *Personal and ubiquitous computing* 17.4, pp. 675–695.

Matan, Ofer et al. (1992). "Multi-digit recognition using a space displacement neural network". In: *Advances in neural information processing systems*, pp. 488–495.

McFee, Brian and Gert Lanckriet (2010). "Metric learning to rank". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 775–782.

Merriaux, Pierre et al. (2017). "A study of vicon system positioning performance". In: *Sensors* 17.7, p. 1591.

Mika, Sebastian, Gunnar Rätsch, and Klaus-Robert Müller (2001). "A mathematical programming approach to the kernel fisher algorithm". In: *Advances in neural information processing systems*, pp. 591–597.

Mohseni, Sina, Niloofar Zarei, and Eric D Ragan (2018). "A survey of evaluation methods and measures for interpretable machine learning". In: *arXiv preprint arXiv:1811.11839*.

Mokbel, Bassam et al. (2015). "Metric learning for sequences in relational LVQ". English. In: *Neurocomputing* (accepted/in press).

Molnar, Christoph (2020). *Interpretable Machine Learning*. Lulu. com.

Montavon, Grégoire, Sebastian Lapuschkin, et al. (2017). "Explaining non-linear classification decisions with deep taylor decomposition". In: *Pattern Recognition* 65, pp. 211–222.

Montavon, Grégoire, Wojciech Samek, and Klaus-Robert Müller (2018). "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73, pp. 1–15.

Morales, Jafet, David Akopian, and Sos Agaian (2014). "Human activity recognition by smartphones regardless of device orientation". In: *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*. Vol. 9030. International Society for Optics and Photonics, p. 90300I.

Mukundan, Arun, Giorgos Tolias, and Ondrej Chum (2017). "Multiple-kernel local-patch descriptor". In: *arXiv preprint arXiv:1707.07825*.

Müller, Meinard (2007). *Information retrieval for music and motion*. Vol. 2. Springer.

Müller, Meinard and Tido Röder (2006). "Motion templates for automatic classification and retrieval of motion capture data". In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 137–146.

Müller, Meinard, Tido Röder, et al. (2007). "Documentation mocap database hdm05". In.

Murdoch, W James et al. (2019). "Interpretable machine learning: definitions, methods, and applications". In: *arXiv preprint arXiv:1901.04592*.

Muybridge, Eadweard (2012). *Animals in motion*. Courier Corporation.

Al-Naser, Mohammad et al. (2018). "Hierarchical Model for Zero-shot Activity Recognition using Wearable Sensors." In: *ICAART (2)*, pp. 478–485.

Al-Naymat, Ghazi, Sanjay Chawla, and Javid Taheri (2012). "SparseDTW: A Novel Approach to Speed up Dynamic Time Warping". In: *CoRR* abs/1201.2969. URL: http://arxiv.org/abs/1201.2969.

Nesterov, Yu (2012). “Efficiency of coordinate descent methods on huge-scale optimization problems”. In: *SIAM Journal on Optimization* 22.2, pp. 341–362.

Neverova, Natalia et al. (2016). “Moddrop: adaptive multi-modal gesture recognition”. In: *IEEE TPAMI* 38.8, pp. 1692–1706.

Nguyen, Anh, Jason Yosinski, and Jeff Clune (2016). “Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks”. In: *arXiv preprint arXiv:1602.03616*.

Nguyen, H. V. et al. (2013). “Design of non-linear kernel dictionaries for object recognition”. In: *IEEE Transactions on Image Processing* 22.12, pp. 5123–5135. ISSN: 10577149. DOI: 10.1109/TIP.2013.2282078.

Nguyen, Nam and Yunsong Guo (2007). “Comparisons of sequence labeling algorithms and extensions”. In: *Proceedings of the 24th international conference on Machine learning*, pp. 681–688.

Niazmardi, Saeid, Abdolreza Safari, and Saeid Homayouni (2017). “A novel multiple-kernel learning framework for multiple feature classification”. In: *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens* 10, pp. 3734–3743.

Nienkötter, Andreas and Xiaoyi Jiang (2016). “Improved prototype embedding based generalized median computation by means of refined reconstruction methods”. In: *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, pp. 107–117.

Ning, Feng et al. (2005). “Toward automatic phenotyping of developing embryos from videos”. In: *IEEE Transactions on Image Processing* 14.9, pp. 1360–1371.

Núñez, Juan C et al. (2018). “Convolutional Neural Networks and Long Short-Term Memory for skeleton-based human activity and hand gesture recognition”. In: *Pattern Recognition* 76, pp. 80–94.

Ohn-Bar, Eshed and Mohan Manubhai Trivedi (2014). “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations”. In: *IEEE transactions on intelligent transportation systems* 15.6, pp. 2368–2377.

Parameswaran, Shibin and Kilian Q Weinberger (2010). “Large margin multi-task metric learning”. In: *Advances in neural information processing systems*, pp. 1867–1875.

Patel, Vishal M and René Vidal (2014). “Kernel sparse subspace clustering”. In: *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, pp. 2849–2853.

Pati, Y. C., R. Rezaiifar, and P. S. Krishnaprasad (1993). “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition”. In: *ACSSC'93*. IEEE, pp. 40–44.

Patterson, Josh and Adam Gibson (2017). *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc."

Pawlyta, Magdalena and Przemysław Skurowski (2016). “A survey of selected machine learning methods for the segmentation of raw motion capture data into functional body mesh”. In: *Conference of Information Technologies in Biomedicine*. Springer, pp. 321–336.

Pekalska, E. and B. Duin (2005). *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific.

Peng, Peixi et al. (2018). “Joint semantic and latent attribute modelling for cross-class transfer learning”. In: *TPAMI* 40.7, pp. 1625–1638.

Peng, Xi et al. (2018). “Structured autoencoders for subspace clustering”. In: *IEEE Transactions on Image Processing* 27.10, pp. 5076–5086.

Petitjean, François, Germain Forestier, Geoffrey I Webb, et al. (2016). “Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm”. In: *Knowledge and Information Systems* 47.1, pp. 1–26.

Petitjean, François, Germain Forestier, Geoffrey I. Webb, et al. (2014). "Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification". In: *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*. IEEE, pp. 470–479.

Pinar, Anthony et al. (2015). "Approach to explosive hazard detection using sensor fusion and multiple-kernel learning with downward-looking GPR and EMI sensor data". In: *Detection and sensing of mines, explosive objects, and obscured targets XX*. Vol. 9454. International Society for Optics and Photonics, 94540B.

Pinheiro, Pedro and Ronan Collobert (2014). "Recurrent convolutional neural networks for scene labeling". In: *International conference on machine learning*, pp. 82–90.

Pouyanfar, Samira et al. (2018). "Multimedia big data analytics: A survey". In: *ACM Computing Surveys (CSUR)* 51.1, pp. 1–34.

Qian, M. et al. (2015). "Structured Sparse Regression for Recommender Systems". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pp. 1895–1898.

Qiu, Qiang, Zhuolin Jiang, and Rama Chellappa (2011). "Sparse dictionary-based representation and recognition of action attributes". In: *ICCV'11*. IEEE, pp. 707–714.

Quan, Yuhui, Chenglong Bao, and Hui Ji (2016). "Equiangular kernel dictionary learning with applications to dynamic texture analysis". In: *CVPR'16*, pp. 308–316.

Quan, Yuhui, Yong Xu, et al. (2016). "Supervised dictionary learning with multiple classifier integration". In: *Pattern Recognition* 55, pp. 247–260.

Rahul, M (2018). "Review on motion capture technology". In: *Global Journal of Computer Science and Technology*.

Raine, Sue, Linzi Meadows, and Mary Lynch-Ellerington (2013). *Bobath concept: theory and clinical practice in neurological rehabilitation*. John Wiley & Sons.

Rakotomamonjy, Alain et al. (2008). "SimpleMKL". In: *Journal of Machine Learning Research* 9.Nov, pp. 2491–2521.

Rakthanmanon, Thanawin and Eamonn J Keogh (2013). "Data Mining a Trillion Time Series Subsequences Under Dynamic Time Warping." In: *IJCAI*, pp. 3047–3051.

Ramirez, Ignacio, Pablo Sprechmann, and Guillermo Sapiro (2010). "Classification and clustering via dictionary learning with structured incoherence and shared features". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, pp. 3501–3508.

Ratanamahatana, Chotirat Ann and Eamonn Keogh (2004). "Everything you know about dynamic time warping is wrong". In: *Third workshop on mining temporal and sequential data*. Vol. 32. Citeseer.

Ren, Zhou, Junsong Yuan, and Zhengyou Zhang (2011). "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera". In: *Proceedings of the 19th ACM international conference on Multimedia*, pp. 1093–1096.

Risse, Benjamin et al. (2017). "FIMTrack: An open source tracking and locomotion analysis software for small animals". In: *PLoS computational biology* 13.5, e1005530.

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407.

Roetenberg, D et al. (2013). "full 6DOF human motion tracking using miniature inertial sensors". In: *MVN white paper*.

Rosch, Eleanor (1975). "Cognitive reference points". In: *Cognitive psychology* 7.4, pp. 532–547.

Rosenblatt, Frank (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rosenhahn, Bodo, Reinhard Klette, and Dimitris Metaxas, eds. (2008). *Human Motion*. Springer Netherlands. DOI: 10.1007/978-1-4020-6693-1.

Rubinstein, Ron, Michael Zibulevsky, and Michael Elad (2008). "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit". In: *Cs Technion* 40.8, pp. 1–15.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Saha, Swapnil Sayan, Sandeep Singh Sandha, and Mani Srivastava (2020). "Deep Convolutional Bidirectional LSTM for Complex Activity Recognition with Missing Data". In: *Human Activity Recognition Challenge*. Springer, pp. 39–53.

Saigo, Hiroto, Jean-Philippe Vert, and Tatsuya Akutsu (2006). "Optimizing amino acid substitution matrices with a local alignment kernel". In: *BMC bioinformatics* 7.1, p. 246.

Sanchez-Martinez, Sergio et al. (2017). "Characterization of myocardial motion patterns by unsupervised multiple-kernel learning". In: *Medical image analysis* 35, pp. 70–82.

Saveriano, Matteo, Felix Franzel, and Dongheui Lee (2019). "Merging position and orientation motion primitives". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7041–7047.

Schleif, F-M et al. (2011). "Efficient kernelized prototype-based classification". In: *International Journal of Neural Systems* 21.06, pp. 443–457.

Schneider, Petra, Michael Biehl, and Barbara Hammer (2009). "Adaptive Relevance Matrices in Learning Vector Quantization". In: *Neural Computation* 21.12, pp. 3532–3561.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

Sedmidubsky, Jan and Jakub Valcik (2013). "Retrieving Similar Movements in Motion Capture Data". English. In: *Similarity Search and Applications*. Ed. by Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula. Vol. 8199. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 325–330. ISBN: 978-3-642-41061-1.

Seiler, Mary C and Fritz A Seiler (1989). "Numerical recipes in C: the art of scientific computing". In: *Risk Analysis* 9.3, pp. 415–416.

Şenel, Lütfi Kerem et al. (2018). "Semantic structure and interpretability of word embeddings". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10, pp. 1769–1779.

Sermanet, Pierre et al. (2013). "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229*.

Sha, Long et al. (2018). "Interactive sports analytics: An intelligent interface for utilizing trajectories for interactive sports play retrieval and analytics". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 25.2, pp. 1–32.

Shahroudy, Amir et al. (2016). "NTU RGB+ D: A large scale dataset for 3D human activity analysis". In: *CVPR'16*, pp. 1010–1019.

Shalev-Shwartz, Shai, Yoram Singer, and Andrew Y Ng (2004). "Online and batch learning of pseudo-metrics". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 94.

Shawe-Taylor, John and Nello Cristianini (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Shen, Shiwen et al. (2019). "An interpretable deep hierarchical semantic convolutional neural network for lung nodule malignancy classification". In: *Expert systems with applications* 128, pp. 84–95.

Shepherd, D (2005). "BBC sport academy cricket umpire signals". In.

Shi, Kejian et al. (2019). “Dynamic barycenter averaging kernel in RBF networks for time series classification”. In: *IEEE Access* 7, pp. 47564–47576.

Shokoohi-Yekta, Mohammad et al. (2015). “On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case.” In: *SDM*.

Shrivastava, Ashish, Jaishanker K Pillai, and Vishal M Patel (2015). “Multiple kernel-based dictionary learning for weakly supervised classification”. In: *Pattern Recognition* 48.8, pp. 2667–2675.

Si, Chenyang et al. (2018). “Skeleton-Based Action Recognition with Spatial Reasoning and Temporal Stack Learning”. In: *arXiv preprint arXiv:1805.02335*.

Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.

Singh, Chandan and Jaspreet Singh (2019). “Geometrically invariant color, shape and texture features for object recognition using multiple-kernel learning classification approach”. In: *Information Sciences* 484, pp. 135–152.

Sivalingam, Ravishankar et al. (2011). “Positive definite dictionary learning for region covariances”. In: *ICCV'11*. IEEE, pp. 1013–1019.

Smisek, Jan, Michal Jancosek, and Tomas Pajdla (2013). “3D with Kinect”. In: *Consumer depth cameras for computer vision*. Springer, pp. 3–25.

Socher, Richard et al. (2013). “Zero-shot learning through cross-modal transfer”. In: *Advances in neural information processing systems*, pp. 935–943.

Song, Sijie et al. (2017). “An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data.” In: *AAAI*. Vol. 1. 2, pp. 4263–4270.

Song, Yale and Mohammad Soleymani (2019). “Polysemous visual-semantic embedding for cross-modal retrieval”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1979–1988.

Song, Yan et al. (2011). “Localized multiple-kernel learning for realistic human action recognition in videos”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.9, pp. 1193–1202.

Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah (2012). “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *CoRR, abs/1212.0402*.

Spiro, Ian, Thomas Huston, and Christoph Bregler (2012). “Markerless Motion Capture in the Crowd”. In: *CoRR* abs/1204.3596. URL: `http://arxiv.org/abs/1204.3596`.

Spriggs, Ekaterina H, Fernando De La Torre, and Martial Hebert (2009). “Temporal segmentation and activity classification from first-person sensing”. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, pp. 17–24.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Stiglic, Gregor et al. (2020). “Interpretability of machine learning based prediction models in healthcare”. In: *arXiv preprint arXiv:2002.08596*.

Stollenwerk, Katharina et al. (2016). “Automatic temporal segmentation of articulated hand motion”. In: *International Conference on Computational Science and Its Applications*. Springer, pp. 433–449.

Strayer, James K (2012). *Linear programming and its applications*. Springer Science & Business Media.

Strickert, Marc et al. (2013). “Regularization and improved interpretation of linear data mappings and adaptive distance measures”. In: *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013*, pp. 10–17. DOI: `10.1109/CIDM.2013.6597211`. URL: `http://dx.doi.org/10.1109/CIDM.2013.6597211`.

Sun, Ju et al. (2009). "Hierarchical spatio-temporal context modeling for action recognition". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2004–2011.

Sun, Lin et al. (2015). "Human action recognition using factorized spatio-temporal convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*, pp. 4597–4605.

Switonski, Adam, Henryk Josinski, and Konrad Wojciechowski (2019). "Dynamic time warping in classification and selection of motion capture data". In: *Multidimensional Systems and Signal Processing* 30.3, pp. 1437–1468.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tahmassebpour, Mahmoudreza (2017). "A new method for time-series big data effective storage". In: *Ieee Access* 5, pp. 10694–10699.

Takeishi, Naoya and Takehisa Yairi (2014). "Anomaly detection from multivariate time-series with sparse representation". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, pp. 2651–2656.

Tang, Yansong et al. (2018). "Deep Progressive Reinforcement Learning for Skeleton-Based Action Recognition". In: *CVPR'18*, pp. 5323–5332.

Teng, Xian, Yu-Ru Lin, and Xidao Wen (2017). "Anomaly detection in dynamic networks using multi-view time-series hypersphere learning". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, pp. 827–836.

Thiagarajan, J. J., K. N. Ramamurthy, and A. Spanias (2014). "Multiple kernel sparse representations for supervised and unsupervised learning". In: *IEEE TIP* 23.7, pp. 2905–2915.

Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso". In: *J. Royal Stat. Soc. Series B (Methodological)*, pp. 267–288.

Tierney, Stephen, Junbin Gao, and Yi Guo (2014). "Subspace clustering for sequential data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1019–1026.

Tompson, Jonathan J et al. (2014). "Joint training of a convolutional network and a graphical model for human pose estimation". In: *Advances in neural information processing systems*, pp. 1799–1807.

Tong, Jijun et al. (2019). "MRI brain tumor segmentation based on texture features and kernel sparse coding". In: *Biomedical Signal Processing and Control* 47, pp. 387–392.

Torr, Philip HS and David W Murray (1994). "Stochastic motion clustering". In: *European Conference on Computer Vision*. Springer, pp. 328–337.

Torresani, Lorenzo and Kuang-chih Lee (2007). "Large margin component analysis". In: *Advances in neural information processing systems*, pp. 1385–1392.

Tran, Du and Alexander Sorokin (2008). "Human activity recognition with metric learning". In: *European conference on computer vision*. Springer, pp. 548–561.

Tsai, Henry et al. (2019). "Small and practical bert models for sequence labeling". In: *arXiv preprint arXiv:1909.00100*.

Tsang, Ivor W, Andras Kocsor, and James T Kwok (2006). "Efficient kernel feature extraction for massive data sets". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 724–729.

Uddin, Md Zia et al. (2020). "A body sensor data fusion and deep recurrent neural network-based behavior recognition approach for robust healthcare". In: *Information Fusion* 55, pp. 105–115.

Van Loan, Charles F (1996). *Matrix computations (Johns Hopkins studies in mathematical sciences)*.

Varma, Manik and Bodla Rakesh Babu (2009). “More generality in efficient multiple-kernel learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 1065–1072.

Velikova, Marina et al. (2014). “Exploiting causal functional relationships in Bayesian network modelling for personalised healthcare”. In: *International Journal of Approximate Reasoning* 55.1, pp. 59–73.

Vemulapalli, Raviteja, Felipe Arrate, and Rama Chellappa (2014). “Human action recognition by representing 3D skeletons as points in a lie group”. In: *CVPR'14*.

Vidal, René and Paolo Favaro (2014). “Low rank subspace clustering (LRSC)”. In: *Pattern Recognition Letters* 43, pp. 47–61.

Vieira, A.W. et al. (Nov. 2012). “Distance matrices as invariant features for classifying MoCap data”. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 2934–2937.

Vigliocco, Gabriella, David P Vinson, and Simona Siri (2005). “Semantic similarity and grammatical class in naming actions”. In: *Cognition* 94.3, B91–B100.

Von. Luxburg, Ulrike (2007). “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4, pp. 395–416.

Vu, Tiep Huu and Vishal Monga (2017). “Fast low-rank shared dictionary learning for image classification”. In: *IEEE Transactions on Image Processing* 26.11, pp. 5160–5175.

Waltemate, Thomas et al. (2015). “Realizing a low-latency virtual reality environment for motor learning”. In: *VRST'15*. ACM, pp. 139–147.

Wang, Hongsong and Liang Wang (2017). “Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks”. In: *CVPR'17*.

Wang J. Yang, J., H. Bensmail, and X. Gao (2014). “Feature selection and multi-kernel learning for sparse representation on a manifold”. In: *Neural Networks* 51, pp. 9–16.

Wang, Jian et al. (2017). “Deep metric learning with angular loss”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2593–2601.

Wang, Jiang et al. (2012). “Mining actionlet ensemble for action recognition with depth cameras”. In: *CVPR'12*. IEEE, pp. 1290–1297.

Wang, Jin et al. (2013). “Biomedical time series clustering based on non-negative sparse coding and probabilistic topic model”. In: *Computer methods and programs in biomedicine* 111.3, pp. 629–641.

Wang, Jun et al. (2013). “Word recognition from continuous articulatory movement time-series data using symbolic representations”. In: *SLPAT'13 Workshop*, pp. 119–127.

Wang, Liang, Li Cheng, and Guoying Zhao (2010). *Machine learning for human motion analysis: theory and practice*. Medical Information Science Reference.

Wang, Lichen, Zhengming Ding, and Yun Fu (2018). “Low-rank transfer human motion segmentation”. In: *IEEE Transactions on Image Processing* 28.2, pp. 1023–1034.

Wang, Qifei et al. (2015). “Unsupervised temporal segmentation of repetitive human actions based on kinematic modeling and frequency analysis”. In: *2015 international conference on 3D vision*. IEEE, pp. 562–570.

Wang, Qingwang, Yanfeng Gu, and Devis Tuia (2016). “Discriminative multiple-kernel learning for hyperspectral image classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* 54.7, pp. 3912–3927.

Wang, Shusen, Alex Gittens, and Michael W Mahoney (2019). “Scalable kernel K-means clustering with Nyström approximation: relative-error bounds”. In: *The Journal of Machine Learning Research* 20.1, pp. 431–479.

Wang, Wei et al. (2019). “A survey of zero-shot learning: Settings, methods, and applications”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2, pp. 1–37.

Wang, Wenjun et al. (2017). "Kernel framework based on non-negative matrix factorization for networks reconstruction and link prediction". In: *Knowledge-Based Systems* 137, pp. 104–114.

Wang, Xinying and Min Han (2014). "Multivariate time series prediction based on multiple-kernel extreme learning machine". In: *2014 international joint conference on neural networks (IJCNN)*. IEEE, pp. 198–201.

Wang, Yiwei et al. (2017). "Optimal collision-free robot trajectory generation based on time series prediction of human motion". In: *IEEE Robotics and Automation Letters* 3.1, pp. 226–233.

Wang, Zhao, Yinfu Feng, Shuang Liu, et al. (2016). "A 3D human motion refinement method based on sparse motion bases selection". In: *Proceedings of the 29th International Conference on Computer Animation and Social Agents*, pp. 53–60.

Wang, Zhao, Yinfu Feng, Tian Qi, et al. (2016). "Adaptive multi-view feature selection for human motion retrieval". In: *Signal Processing* 120, pp. 691–701.

Weinberger, Kilian Q and Lawrence K Saul (2008). "Fast solvers and efficient implementations for distance metric learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 1160–1167.

— (2009). "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *Journal of Machine Learning Research* 10, pp. 207–244. DOI: 10.1145/1577069.1577078. URL: http://doi.acm.org/10.1145/1577069.1577078.

Weinzaepfel, Philippe, Xavier Martin, and Cordelia Schmid (2016). "Human Action Localization with Sparse Spatial Supervision". In: *arXiv preprint arXiv:1605.05197*.

Woiwood, Ian, Donald Russell Reynolds, and Chris D Thomas (2001). *Insect Movement: Mechanisms and Consequences: Proceedings of the Royal Entomological Society's 20th Symposium*. CABI.

Wolf, Ralph and John C Platt (1994). "Postal address block location using a convolutional locator network". In: *Advances in Neural Information Processing Systems*, pp. 745–752.

Xia, Guiyu et al. (2016). "Keyframe extraction for human motion capture data based on joint kernel sparse representation". In: *IEEE Transactions on Industrial Electronics* 64.2, pp. 1589–1599.

Xia, Lu, Chia-Chih Chen, and JK Aggarwal (2012). "View invariant human action recognition using histograms of 3d joints". In: *CVPRW'12 Workshops*. IEEE, pp. 20–27.

Xiao, Qinkun and Chaoqin Chu (2017). "Human motion retrieval based on deep learning and dynamic time warping". In: *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, pp. 426–430.

Xiao, Qinkun and Ren Song (2017). "Motion retrieval based on Motion Semantic Dictionary and HMM inference". In: *Soft Computing* 21.1, pp. 255–265.

Xiao, Shijie et al. (2016). "Robust kernel low-rank representation". In: *IEEE transactions on neural networks and learning systems* 27.11, pp. 2268–2281.

Xiao, Xiao and Donghui Chen (2014). "Multiplicative iteration for nonnegative quadratic programming". In: *arXiv preprint arXiv:1406.1008*.

Xie, Christopher et al. (2019). "Object discovery in videos as foreground motion clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9994–10003.

Xie, Zhengtai et al. (2020). "A data-driven cyclic-motion generation scheme for kinematic control of redundant manipulators". In: *IEEE Transactions on Control Systems Technology*.

Xu, Bing et al. (2015). "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853*.

Xu, Huile et al. (2016). "Wearable sensor-based human activity recognition method with multi-features extracted from Hilbert-Huang transform". In: *Sensors* 16.12, p. 2048.

Xu, Long et al. (2014). "Violent video detection based on MoSIFT feature and sparse coding". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3538–3542.

Xu, Rui and Donald Wunsch (2005). "Survey of clustering algorithms". In: *IEEE Transactions on neural networks* 16.3, pp. 645–678.

Xu, Zenglin, Rong Jin, Haiqin Yang, et al. (2010). "Simple and efficient multiple-kernel learning by group lasso". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. Citeseer, pp. 1175–1182.

Xu, Zenglin, Rong Jin, Jieping Ye, et al. (2009). "Non-monotonic feature selection". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 1145–1152.

Xue, Hui, Yu Song, and Hai-Ming Xu (2017). "Multiple indefinite kernel learning for feature selection". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 3210–3216.

Yabukami, Shin et al. (2000). "Motion capture system of magnetic markers using three-axial magnetic field sensor". In: *IEEE transactions on magnetics* 36.5, pp. 3646–3648.

Yan, Fei et al. (2009). "A comparison of l_1 norm and l_2 norm multiple-kernel SVMs in image and video classification". In: *2009 Seventh International Workshop on Content-Based Multimedia Indexing*. IEEE, pp. 7–12.

Yan, Sijie, Yuanjun Xiong, and Dahua Lin (2018). "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *arXiv preprint arXiv:1801.07455*.

Yan, Yichao et al. (2017). "Skeleton-aided articulated motion generation". In: *Proceedings of the 25th ACM international conference on Multimedia*, pp. 199–207.

Yan, Zhiguo, Zhizhong Wang, and Hongbo Xie (2008). "The application of mutual information-based feature selection and fuzzy LS-SVM-based classifier in motion classification". In: *Computer Methods and Programs in Biomedicine* 90.3, pp. 275–284.

Yang, Jingjing et al. (2012). "Group-sensitive multiple-kernel learning for object recognition". In: *IEEE Transactions on Image Processing* 21.5, pp. 2838–2852.

Yang, Meng, Heyou Chang, and Weixin Luo (2017). "Discriminative analysis-synthesis dictionary learning for image classification". In: *Neurocomputing* 219, pp. 404–411.

Yang, Meng, Lei Zhang, et al. (2011). "Fisher discrimination dictionary learning for sparse representation". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 543–550.

Yang, Yingzhen, Jiashi Feng, et al. (2016). "ℓ^0-Sparse-Sparse Subspace Clustering". In: *European conference on computer vision*. Springer, pp. 731–747.

Yang, Yingzhen, Zhangyang Wang, et al. (2014). "Data Clustering by Laplacian Regularized L1-Graph." In: *AAAI*, pp. 3148–3149.

Yang, Zhilin, Ruslan Salakhutdinov, and William Cohen (2016). "Multi-task cross-lingual sequence tagging from scratch". In: *arXiv preprint arXiv:1603.06270*.

Yao, Lina et al. (2015). "Freedom: Online activity recognition via dictionary-based sparse representation of rfid sensing data". In: *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, pp. 1087–1092.

Ye, Jieping, Shuiwang Ji, and Jianhui Chen (2008). "Multi-class discriminant kernel learning via convex programming". In: *Journal of Machine Learning Research* 9.Apr, pp. 719–758.

Ye, Lexiang and Eamonn Keogh (2009). "Time series shapelets: a new primitive for data mining". In: *Proceedings of SIGKDD'09*. ACM, pp. 947–956.

Yeh, Chin-Chia Michael (2018). "Towards a Near Universal Time Series Data Mining Tool: Introducing the Matrix Profile". In: *arXiv preprint arXiv:1811.03064*.

Yeh, Chin-Chia Michael, Nickolas Kavantzas, and Eamonn Keogh (2017). "Matrix profile vi: meaningful multidimensional motif discovery". In: *2017 IEEE international conference on data mining (ICDM)*. IEEE, pp. 565–574.

Yin, Ming et al. (2016). "Kernel sparse subspace clustering on symmetric positive definite manifolds". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5157–5164.

Yogatama, Dani et al. (2015). "Learning word representations with hierarchical sparse coding". In: *International Conference on Machine Learning*, pp. 87–96.

You, Chong, Daniel Robinson, and René Vidal (2016). "Scalable sparse subspace clustering by orthogonal matching pursuit". In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 3918–3927.

Yuan, Xiao-Tong, Xiaobai Liu, and Shuicheng Yan (2012). "Visual classification with multitask joint sparse representation". In: *IEEE Transactions on Image Processing* 21.10, pp. 4349–4360.

Zeng, Zhengxin, Moeness G Amin, and Tao Shan (2020). "Arm Motion Classification Using Time-Series Analysis of the Spectrogram Frequency Envelopes". In: *Remote Sensing* 12.3, p. 454.

Zhang, Chunjie et al. (2011). "Image classification by non-negative sparse coding, low-rank and sparse decomposition". In: *CVPR'11*. IEEE, pp. 1673–1680.

Zhang, Daoqiang, Zhi-Hua Zhou, and Songcan Chen (2006). "Non-negative matrix factorization on kernels". In: *Pacific Rim International Conference on Artificial Intelligence*. Springer, pp. 404–412.

Zhang, Fuzhen, Qingling Zhang, et al. (2006). "Eigenvalue inequalities for matrix product". In: *IEEE Transactions on Automatic Control* 51.9, p. 1506.

Zhang, Jiaqi and Xiaoyi Jiang (2020). "Improved Computation of Affine Dynamic Time Warping". In: *Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, pp. 1–5.

Zhang, Li et al. (2011). "Kernel sparse representation-based classifier". In: *IEEE Transactions on Signal Processing* 60.4, pp. 1684–1695.

Zhang, Mi and Alexander A Sawchuk (2013). "Human daily activity recognition with sparse representation using wearable sensors". In: *IEEE journal of Biomedical and Health Informatics* 17.3, pp. 553–560.

Zhang, Pengfei et al. (2017). "View adaptive recurrent neural networks for high performance human action recognition from skeleton data". In: *arXiv, no. Mar*.

Zhang, Quanshi, Ying Nian Wu, and Song-Chun Zhu (2018). "Interpretable convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8827–8836.

Zhang, Xiaoqian et al. (2019). "Robust low-rank kernel multi-view subspace clustering based on the schatten p-norm and correntropy". In: *Information Sciences* 477, pp. 430–447.

Zhang, Yungang, Tianwei Xu, and Jieming Ma (2017). "Image categorization using non-negative kernel sparse representation". In: *Neurocomputing* 269, pp. 21–28.

Zhang, Zhao et al. (2017). "Jointly learning structured analysis discriminative dictionary and analysis multiclass classifier". In: *IEEE transactions on neural networks and learning systems* 29.8, pp. 3798–3814.

Zhang, Zheng et al. (2017). "Dynamic time warping under limited warping path length". In: *Information Sciences* 393, pp. 91–107.

Zhang, Ziming and Venkatesh Saligrama (2015). "Zero-shot learning via semantic similarity embedding". In: *Proceedings of the IEEE international conference on computer vision*, pp. 4166–4174.

Zhen, Xiantong et al. (2013). "Embedding motion and structure features for action recognition". In: *IEEE Transactions on Circuits and Systems for Video Technology* 23.7, pp. 1182–1190.

Zheng, Yi et al. (2014). "Time series classification using multi-channels deep convolutional neural networks". In: *WAIM'14*. Springer, pp. 298–310.

Zhou, Baoding, Jun Yang, and Qingquan Li (2019). "Smartphone-based activity recognition for indoor localization using a convolutional neural network". In: *Sensors* 19.3, p. 621.

Zhou, Bolei et al. (2016). "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929.

Zhou, Feng, Fernando De la Torre, and Jessica K Hodgins (2008). "Aligned cluster analysis for temporal segmentation of human motion". In: *2008 8th IEEE international conference on automatic face & gesture recognition*. IEEE, pp. 1–7.

— (2012). "Hierarchical aligned cluster analysis for temporal clustering of human motion". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3, pp. 582–596.

— (2013). "Hierarchical aligned cluster analysis for temporal clustering of human motion". In: *IEEE TPAMI* 35.3, pp. 582–596.

Zhou, Feng and Fernando De la Torre Frade (June 2012). "Generalized Time Warping for Multi-modal Alignment of Human Motion". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, Huiyu and Huosheng Hu (2008). "Human motion tracking for rehabilitation—A survey". In: *Biomedical signal processing and control* 3.1, pp. 1–18.

Zhou, Ning et al. (2012). "Learning inter-related visual dictionary for object recognition". In: *CVPR'12*. IEEE, pp. 3490–3497.

Zhou, Tao et al. (2020). "Multi-mutual consistency induced transfer subspace learning for human motion segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10277–10286.

Zhu, Wencheng, Jiwen Lu, and Jie Zhou (2018). "non-linear subspace clustering for image clustering". In: *Pattern Recognition Letters* 107, pp. 131–136.

Zhu, Wentao et al. (2016). "Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks." In: *AAAI*. Vol. 2. 5, p. 6.

Zhu, X. et al. (2017). "Multi-Kernel Low-Rank Dictionary Pair Learning for Multiple Features Based Image Classification." In: *AAAI*, pp. 2970–2976.

Zhuang, L. et al. (2012). "Non-negative low rank and sparse graph for semi-supervised learning". In: *CVPR 2012*. IEEE, pp. 2328–2335.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

APPENDIX A

A.1 PROOF OF THEOREM 3.1

In Section 3.3, I formulated the following theorem.

A triplet $(\vec{x}_i, \vec{x}_j, \vec{x}_l)$results in Equation 3.12 being infeasible if $(\vec{x}_i - \vec{x}_j)$ and $(\vec{x}_i - \vec{x}_l)$ are linearly dependent vectors.

Proof. A matrix $\mathbf{Q} := \mathbf{Q}_{ijl}$ as in Equation 3.12 can be written in the form of $\vec{a}\vec{a}^\top - \vec{b}\vec{b}^\top$, i.e., its eigenvectors are obviously located in the span of $\vec{a}$ and $\vec{b}$. Hence, the rank of $\mathbf{Q}$ is at most 2 by denoting its two possibly non-zero eigenvalues as $\lambda_{\min}(\mathbf{Q}) \leq \lambda_{\max}(\mathbf{Q})$. Therefore, we can find a basis of $\mathbb{R}^n$ whose first two elements are $\vec{a}$ and $\vec{b}$. With respect to this basis, the matrix $\mathbf{Q}$ has the form

$$\begin{bmatrix} \vec{a}\cdot\vec{a} & \vec{a}\cdot\vec{b} & * & \cdots & * \\ -\vec{a}\cdot\vec{b} & -\vec{b}\cdot\vec{b} & * & \cdots & * \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Since this is a block upper triangular matrix, the $\lambda_{\min}(\mathbf{Q})\lambda_{\max}(\mathbf{Q})$ product is equal to the determinant of its first diagonal block as

$$-\|\vec{a}\|^2\|\vec{b}\|^2 + (\vec{a}\cdot\vec{b})^2 = -\|\vec{a}\|^2\|\vec{b}\|^2 \sin^2\theta \tag{A.1}$$

in which θ is the angle between the two vectors $\vec{a}$ and $\vec{b}$.

Considering the sign and possible values of Equation A.1, $\lambda_{\min}(\mathbf{Q}) \leq 0 < \lambda_{\max}(\mathbf{Q})$ if the two vectors are linearly independent (unless vectors themselves are degenerate). The equality $\lambda_{\min}(\mathbf{Q}) = 0$ corresponds to linearly dependent vectors $\vec{a}$ and $\vec{b}$, namely $\theta = 0$. In that case, $\mathbf{Q}$ is a PSD matrix, and Equation 3.12 becomes infeasible. □

A.2 PROOF OF LEMMA 3.1

In Section 3.3, I formulated the following lemma.

Denote the eigenvalues of a matrix $\mathbf{Q} \in \mathrm{R}^{d\times d}$ by $\lambda_1(\mathbf{Q}) \geq \lambda_2(\mathbf{Q}) \geq \ldots$, its smallest/largest eigenvalue is denoted $\lambda_{min}(\mathbf{Q})$ and $\lambda_{max}(\mathbf{Q})$, respectively. Then, for hermitian $\mathbf{Q} \in \mathrm{R}^{d\times d}$ and symmetric PSD $\mathbf{M} \in \mathrm{R}^{d\times d}$, it holds $\lambda_k(\mathbf{Q})\lambda_{min}(\mathbf{M}) \leq \lambda_k(\mathbf{QM})$ for all k.

Proof. $\mathbf{M}$ is PSD, and $\mathbf{Q}$ and $\mathbf{M}$ are symmetric. Hence,

$$\lambda_k(\mathbf{QM}) = \lambda_k(\mathbf{Q}\sqrt{\mathbf{M}}\sqrt{\mathbf{M}}) = \lambda_k(\sqrt{\mathbf{M}}\mathbf{Q}\sqrt{\mathbf{M}}),$$

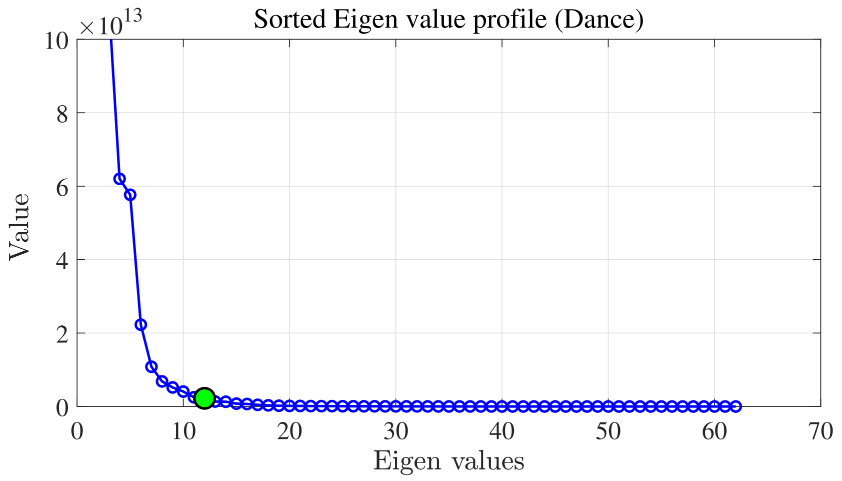


Figure A.1: Eigenvalue profile of the learned metric for the Dance dataset, which is sorted according to the size of matrix $\mathbf{DD}^\top$'s eigenvalues. The green circle indicates the selected dimension as the effective dimension for the regularized coefficients Φ.

where $\sqrt{\mathbf{M}}$ is the principal square root of $\mathbf{M}$. Using the min-max theorem we find

$$\begin{aligned}\lambda_k(\mathbf{QM}) = & \min_{\dim(F)=k} \left(\max_{x\in F\setminus\{0\}} \frac{\langle Q\sqrt{\mathbf{M}}x, \sqrt{\mathbf{M}}x\rangle}{\langle \sqrt{\mathbf{M}}x, \sqrt{\mathbf{M}}x\rangle} \frac{\langle \mathbf{M}x, x\rangle}{\langle x, x\rangle}\right) \\ & \geq \lambda_{min}(\mathbf{M}) \min_{\dim(F)=k} \left(\max_{x\in F\setminus\{0\}} \frac{\langle \mathbf{Q}\sqrt{\mathbf{M}}x, \sqrt{\mathbf{M}}x\rangle}{\langle \sqrt{\mathbf{M}}x, \sqrt{\mathbf{M}}x\rangle}\right),\end{aligned}$$

because $\frac{\langle \mathbf{M}x, x\rangle}{\langle x, x\rangle} \geq \lambda_{min}(\mathbf{M})$. Again using the min-max theorem we get

$$\lambda_k(\mathbf{QM}) \geq \lambda_{min}(\mathbf{M})\lambda_k(\mathbf{Q}).$$

□

A.3 ADDITIONAL FIGURES FOR SECTION 3.5

Related to the experiments of Section 3.5 for regularizing the relevance profiles, Figure A.1 shows how I choose 12 effective dimensions (eigenvectors) based on the corresponding eigenvalue profile of $\mathbf{DD}^\top$ for the Dance dataset to construct the regularization matrix Φ in Equation 3.18. As the result of feature selection on the Dance dataset in Section 3.5, the relevant body joints to the selected 9 features are depicted on the skeleton structure in Figure A.2. Relevantly, Figure A.3 illustrates that for a wide range of effective dimensions in Equation 3.18, test data classification accuracy stays at its maximum point (the Dance dataset).

The feature selection part of Section 3.5 on the Walking dataset results 7 selected features with the corresponding body joints as depicted on the skeleton structure of Figure A.4.

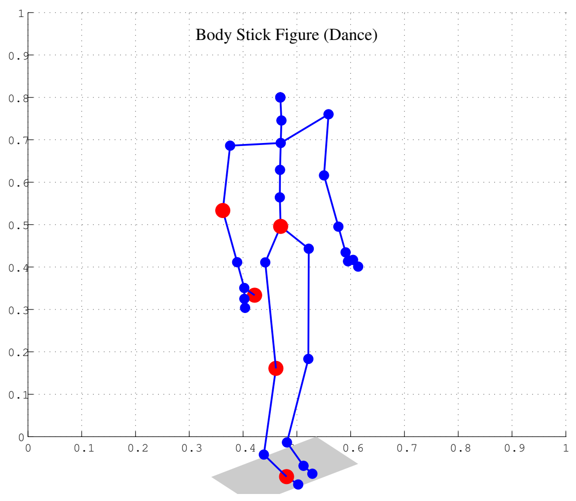


Figure A.2: Stick figure of different body parts related to the Dance dataset. Red markers are the selected important inputs according to the regularized relevance profile of the features.

A.4 PROOF OF PROPOSITION 4.1

In Section 4.2, I formulated the following proposition.

If $rank(\Phi(\mathcal{X})) < N$, there exist $\mathbf{U}^* \in \mathbb{R}^{N \times k}, \mathbf{\Gamma}^* \in \mathbb{R}^{k \times N}$ $k < N$ such that $\Phi(\mathcal{X})$ can be reconstructed as $\Phi(\mathcal{X}) = \Phi(\mathcal{X})\mathbf{U}^*\mathbf{\Gamma}^*$.

Proof. Knowing that $rank(\Phi(\mathcal{X})) < N$, there exists $\mathbf{U}^* \in \mathbb{R}^{N \times k}$ $k < N$ such that $\Phi(\mathcal{X}) \in span\{\Phi(\mathcal{X})\mathbf{U}^*\}$. This means that the columns of $\Phi(\mathcal{X})$ can be reconstructed in a linear combination as $\Phi(\mathcal{X}) = \Phi(\mathcal{X})\mathbf{U}^*\mathbf{\Gamma}^*$, where $\mathbf{\Gamma} \in \mathbb{R}^{k \times N}$. $\square$

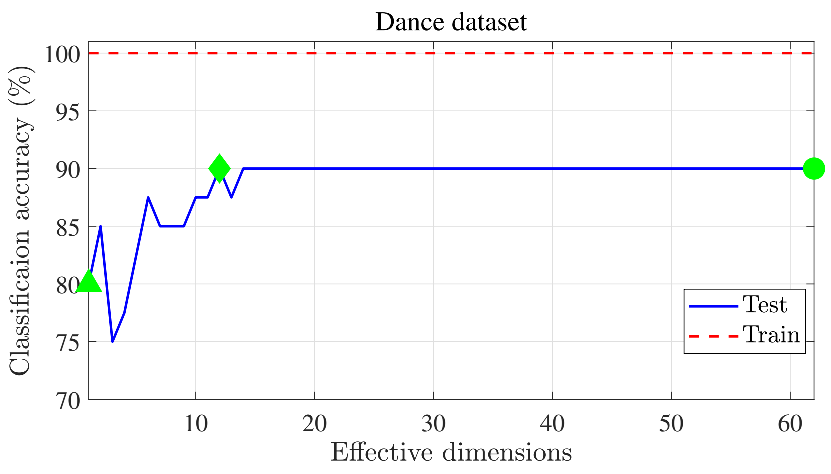


Figure A.3: Classification accuracy for training and test set of the dance dataset based on the selected effective dimensions in Equation 3.18. The green diamond represents the highest accuracy for the test set for 12 effective dimensions. The green circle refers to the non-regularized coefficients, and the triangle to only one effective dimension.

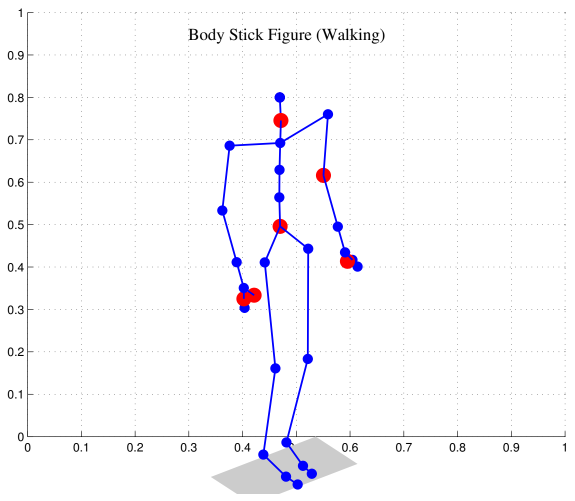


Figure A.4: Stick figure of different body parts related to the Walking dataset. Red markers are the selected important inputs according to the regularized relevance profile of the features.

A.5 THE K-NNLS ALGORITHM

In Section 4.2, I proposed the K-NNLS algorithm by kernelizing the active set fast non-negative least square optimization method (FNNLS) from (Bro and De Jong 1997).

Algorithm A.1 The K-NNLS algorithm: finds an approximate solution to step 8 of Algorithm 4.1 as a non-negative encoding of a data sample $\vec{z}$ in the feature space given a subset dictionary matrix.

```
1: **Input:** Subset dictionary matrix $\mathbf{U}_I \in \mathbb{R}^{N\times k}$, kernel matrix $\mathcal{K}(\mathbf{X},\mathbf{X})$
2: **Output:** Solution $\vec{\gamma}$ to $\arg\min_{\vec{\gamma}} \|\Phi(\vec{z}) - \Phi(\mathbf{X})\mathbf{U}_I\vec{\gamma}\|_2^2$, s.t $\gamma_j \geq 0,\ \forall j$
3: Initialization: $\vec{\gamma} = 0, P = \emptyset, R = \{1,\dots,k\}, \vec{w} = \mathbf{U}_I^\top \mathcal{K}(\vec{z},\mathbf{X})^\top$
4: **while** $R \neq \emptyset$ **do**
5:     $j = \arg\max_{i\in R}(w_i)$
6:     $P = P \cup \{j\}$, $R = R\backslash\{j\}$
7:     $\vec{s}^P = [(\mathbf{U}_I^\top \mathcal{K}(\mathbf{X},\mathbf{X})\mathbf{U}_I)^p]^{-1}[(\mathcal{K}(\vec{z},\mathbf{X})\mathbf{U}_I)^p]^\top$
8:     **if** $\min(\vec{s}^P) < 0$ **then**
9:         $Q = \{i | s_i^p < 0, \forall i \in P\}$
10:        $\alpha = -\min_i[\frac{\gamma_i}{\gamma_i - s_i}], \quad \forall i \in Q$
11:        $\vec{\gamma} := \vec{\gamma} + \alpha(\vec{s} - \vec{\gamma})$
12:        $Q = \{i | \gamma_i < 0, \forall i \in P\}$
13:        $R = R \cup Q$, $P = P \backslash Q$
14:        $\vec{s}^P = [(\mathbf{U}_I^\top \mathcal{K}(\mathbf{X},\mathbf{X})\mathbf{U}_I)^p]^{-1}[(\mathcal{K}(\vec{z},\mathbf{X})\mathbf{U}_I)^p]^\top$
15:        $\vec{s}^R = 0$
16:    **end if**
17:    $\vec{\gamma} = \vec{s}$
18:    $\vec{w} = \mathbf{U}_I^\top [\mathcal{K}(\vec{z},\mathbf{X}) - \mathcal{K}(\mathbf{X},\mathbf{X})\mathbf{U}_I\vec{\gamma}]^\top$
19: **end while**
```

A.6 PROOF OF PROPOSITION 4.2

In Section 4.3, I formulated the following proposition.

Using the dictionary structure of Equation 4.22, sparse reconstruction of sequence $\Phi(\mathbf{X})$ necessitates to bound the value of $\|\vec{u}_i\|_0$.

Proof. For each training sample $\Phi(\mathbf{X})$, we have

$$\Phi(\mathbf{X}) = \Phi(\mathcal{X})\mathbf{U}\vec{\gamma} = \Phi(\mathcal{X})\vec{s},$$

where $\vec{s} \in \mathbb{R}^N$ denotes the weighting vector for the reconstruction of $\mathbf{X}$ based on other training samples in $\mathcal{X}$. Therefore,

$$\|\vec{s}\|_0 = \|\mathbf{U}\vec{\gamma}\|_0 \le \{\max_i \|\vec{u}_i\|_0\} \times \|\vec{\gamma}\|_0.$$

With $\mathbf{U}$ being unbounded as $\|\vec{u}_i\|_0 < T_{\mathbf{U}}$, we have $\|\vec{s}\|_0 \le T_{\mathbf{U}}T$. Therefore, having no specif bound on the column-carnality of $\mathbf{U}$ leads to $\|\vec{s}\|_0 \le N$, i.e., it practically removes any upper bound on the reconstruction of $\Phi(\mathbf{X})$. □

A.7 PROOF OF PROPOSITION 4.3

In Section 4.3, I formulated the following proposition.

If sequence $\Phi(\mathbf{X})$ belongs to the class q and is lying on a union of subspaces with arbitrarily small contributions from the subspaces $s \neq q$, then the non-negative discriminant combination $\{\mathbf{U}, \vec{\gamma}\}$ can reconstruct $\Phi(\mathbf{X})$ such that

$$\frac{\sum_{s\neq q} \vec{h}^s \mathbf{U}\vec{\gamma}}{\vec{h}^q \mathbf{U}\vec{\gamma}} \le \epsilon$$

for an arbitrarily small ϵ.

Proof. Denote $\mathcal{X}_q \subset \mathcal{X}$ as the set of sequences from class q and $\mathcal{X}_{\bar{q}}$ as its complement. Based on the assumption, we have

$$\Phi(\mathbf{X}) = \Phi(\mathbf{X})^q + \Phi(\mathbf{X})^\perp,$$

such that $\Phi(\mathbf{X})^q \in span\{\Phi(\mathcal{X}_q)\}$ and $\Phi(\mathbf{X})^\perp \in span\{\Phi(\mathcal{X}_{\bar{q}})\}$, while $\|\Phi(\mathbf{X})^\perp\|_2$ is arbitrarily small.
Therefore, we can write

$$\Phi(\mathbf{X}) = \Phi(\mathcal{X}_q)\vec{s} + \Phi(\mathcal{X}_{\bar{q}})\vec{\bar{s}},$$

such that the spanning vectors $(\vec{s}, \vec{\bar{s}})$ are non-negative and $\frac{\sum_i \bar{s}_i}{\sum_i s_i} \le \epsilon$ for an arbitrarily small ϵ.
Now, denote $\mathbf{U}_q$ as the sub-matrix of $\mathbf{U}$ with non-zero entries corresponding to members of $\Phi(\mathcal{X})_q$ and $\mathbf{U}_{\bar{q}}$ as its complement. Hence, we can obtain the non-negative matrices $(\mathbf{U}, \vec{\gamma})$ such that $\vec{s} = \mathbf{U}_q\vec{\gamma}$ and $\vec{\bar{s}} = \mathbf{U}_{\bar{q}}\vec{\gamma}$, which holds $\Phi(\mathbf{X}) = \Phi(\mathcal{X})\mathbf{U}\vec{\gamma}$.
Consequently, since $\vec{\bar{s}} = \sum_{s\neq q} \vec{h}^s \mathbf{U}\vec{\gamma}$, we derive

$$\frac{\sum_{s\neq q} \vec{h}^s \mathbf{U}\vec{\gamma}}{\vec{h}^q \mathbf{U}\vec{\gamma}} \le \epsilon$$

□

A.8 PROOF OF PROPOSITION 4.4

In Section 4.3, I formulated the following proposition.

The proposed loss term $\mathcal{G}$ in Equation 4.26 is non-convex and has a non-negative gradient.

Proof. We can rewrite

$$\mathcal{G}(\mathbf{H}, \vec{\gamma}, \mathbf{U}) = \vec{\gamma}^\top \mathbf{U}^\top \mathbf{H}^\top (\mathbf{1} - \mathbf{I}) \mathbf{H} \mathbf{U} \vec{\gamma}, \tag{A.2}$$

where $\mathbf{1} \in \mathbb{R}^{C \times C}$ is the matrix of ones, and $\mathbf{I}$ is the identity matrix. Hence, the matrix $(\mathbf{1} - \mathbf{I})$ has one positive eigenvalue C and $C-1$ eigenvalues with the magnitude of -1. Hence, according to the quadratic form of $\mathcal{G}$ w.r.t. $\vec{\gamma}$, it has $(k-C)$ zero eigenvalues and C non-zero eigenvalues with the same structure as in $(\mathbf{1} - \mathbf{I})$, which makes $\mathcal{G}(\vec{\gamma})$ a non-convex function. In addition, its gradient w.r.t. $\vec{\gamma}$ is computed as $\nabla_{\vec{\gamma}} \mathcal{G} = 2(\mathbf{1} - \mathbf{I}) \mathbf{H} \mathbf{U} \vec{\gamma}$, which has non-negative entries given that $\mathbf{H} \mathbf{U} \vec{\gamma}$ is a non-negative vector. □

A.9 PROOF OF PROPOSITION 4.5

In Section 4.3, I formulated the following proposition.

Define

$$\mathbf{V} := \mathcal{K}(\mathcal{X}, \mathcal{X}) + \alpha \mathbf{H}^\top (\mathbf{1} - \mathbf{I}_{C \times C}) \mathbf{H}$$

and $\beta := -\min_i \lambda_i$, with $\{\lambda_i\}_{i=1}^N$ as the eigenvalues of $\mathbf{V}$. Adding $\beta \|\mathbf{U}\vec{\gamma}\|_2^2$ to the objective term $\mathcal{G}$ (Equation 4.26) makes Equation 4.21 a convex optimization problem.

Proof. After adding $\beta \|\mathbf{U}\vec{\gamma}\|_2^2$ to objective terms $\mathcal{G}(\mathbf{H}, \mathbf{U}, \vec{\gamma})$ and $\mathcal{R}(\mathcal{X}, \mathbf{Z}, \mathbf{U}, \vec{\gamma})$ from Equation 4.26 [1] and Equation 4.24, the quadratic terms can be rewritten as

$$\vec{\gamma}^\top \mathbf{U}^\top (\mathbf{V} + \beta \mathbf{I}_{N \times N}) \mathbf{U} \vec{\gamma}.$$

Based on Proposition 4.4, the eigenvalues of $\mathbf{V}$ can include both negative and positive values. Therefore, choosing $\beta = -\min_i \lambda_i$ makes $(\mathbf{V} + \beta \mathbf{I}_{N \times N})$ a positive semi-definite matrix (PSD), and consequently, the whole objective becomes PSD due to its quadratic form. Hence, Equation 4.21 becomes a convex problem via adding this term. □

A.10 PROOF OF THEOREM 4.1

In Section 4.3, I formulated the following theorem.

The Non-negative Quadratic Pursuit algorithm (Algorithm 4.5) converges to a local minimum of Equation 4.32 in a limited number of iterations.

Proof. The algorithm consists of 3 main parts:

1. Gradient-based dimension selection
2. Closed-form solution

1 Using the reformulation of $\mathcal{G}$ from Equation A.2 can facilitate this algebraic derivation.

3. Non-negative line search and updating $\mathcal{I}$.

It is clear that the closed-form solution $\vec{\gamma}$ via selecting a negative direction of the gradient $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ always reduces the current value of $f(\vec{\gamma}^t)$ as $\vec{\gamma}^t$ has to be non-negative and initially $\gamma_j = 0$. Moreover, the zero-crossing line search in iteration t can guarantee to reduce the value of $f(\vec{\gamma}^{(t-1)})$ strictly. It finds a non-negative $\vec{\gamma}^t_{new}$ between the line connecting $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$ to $\vec{\gamma}^t_{\mathcal{I}}$, and since $f(\vec{\gamma})$ is convex, $f(\vec{\gamma}^t_{new}) < f(\vec{\gamma}^{(t-1)}_{\mathcal{I}})$

Consequently, each of the above steps guarantees a monotonic decrease in the value of $f(\vec{\gamma})$. Therefore, having $\|\vec{\gamma}^{(t+i)}\|_0 > \|\vec{\gamma}^{(t)}\|_0$ implies $f(\vec{\gamma}^{(t+i)}) < f(\vec{\gamma}^{(t)})$. Also, the algorithm structure guarantees that in any iteration t, $\mathcal{I}_t \neq \mathcal{I}_i \; \forall i < t$, meaning that NQP never gets trapped into a loop of repeated dimension selections. Furthermore, we have $\|\vec{\gamma}\|_0 \leq \; nT$, meaning that the total number of possible selections in $\mathcal{I}$ is bounded. Inferring from the above, the NQP algorithm converges in a limited number of iterations. □

A.11 PROOF OF PROPOSITION 5.1

In Section 5.3, I formulated the following proposition.

The objective $\mathcal{J}_{dis}$ in Equation 5.12 has its minimum if $\forall \mathbf{X}_i$, $\hat{\Phi}(\mathbf{X}_i) \approx \hat{\Phi}(\mathcal{X})\mathbf{U}\vec{\gamma}_i$, such that $\forall t : \gamma_{ti} \neq 0, \forall s : u_{st} \neq 0, \vec{h}_i = \vec{h}_s$ and $\|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 \approx 0$.

Proof. The objective term $\mathcal{J}_{dis}$ is constructed upon summation and multiplication of non-negative elements. Hence, its global minima would lie where $\mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}) = 0$ holds. This condition can be fulfilled if for each $\vec{\gamma}_i$:

$$[\sum_{s=1}^{N} \vec{u}^s (\vec{h}_i^\top \vec{h}_s \|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \|\vec{h}_i - \vec{h}_s\|_2^2)]\vec{\gamma}_i = 0.$$

Since the trivial solution $\vec{\gamma}_i = 0$ is avoided due to $\mathcal{J}_{rec}$ in Equation 5.11, we can find a set $\mathcal{I}$ s.t. $\forall t \in \mathcal{I}, \gamma_{ti} \neq 0$ holds. Therefore, $\forall t \in \mathcal{I}, \sum_{s=1}^{N} u_{st}\Omega_{si} = 0$, where

$$\Omega_{si} = \vec{h}_i^\top \vec{h}_s \|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \|\vec{h}_i - \vec{h}_s\|_2^2.$$

It is clear that

$$\Omega_{si} = \begin{cases} 2 & \vec{h}_i \neq \vec{h}_s \\ \|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 & \vec{h}_i = \vec{h}_s, \end{cases}$$

which means that $\forall s$, $u_{st}\Omega_{si} = 0$ holds in either of the following cases:

1. $u_{st} = 0$, meaning that the data point $\mathbf{X}_s$ does not contribute to the t-th prototype (e.g., consider the squares in Figure 5.2-b that are not a part of $\vec{u}_1$) .
2. $\vec{u}_t$ uses $\mathbf{X}_s$ that lies in the same class as $\mathbf{X}_i$ (e.g., the circles in Figure 5.2-b as the main constituents of $\vec{u}_1$).

Putting all the above conditions together, $\mathcal{J}_{dis} = 0$ happens only in case the condition described by the proposition is fulfilled. □

A.12 PROOF OF PROPOSITION 5.2

In Section 5.3, I formulated the following proposition.

Denoting $\mathbf{U} \in \mathbb{R}^{N\times k}$, $\boldsymbol{\Gamma} \in \mathbb{R}^{k\times N}$, $\vec{\beta} \in \mathbb{R}^{d}$, and

$$\begin{aligned}\mathcal{G}(\mathbf{U},\boldsymbol{\Gamma},\vec{\beta}) = \; & \|\hat{\Phi}(\mathcal{X}) - \hat{\Phi}(\mathcal{X})\mathbf{U}\boldsymbol{\Gamma}\|_F^2 \\ & +\lambda\tfrac{1}{2}\sum_{i=1}^{N}[\sum_{s=1}^{N}\vec{u}^{s}\vec{\gamma}_i(\vec{h}_i^{\top}\vec{h}_s\|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \|\vec{h}_i - \vec{h}_s\|_2^2)] \\ & +\mu\sum_{i=1}^{N}[\sum_{s\in\mathcal{N}_i^k}\|\hat{\Phi}(\mathbf{X}_i) - \hat{\Phi}(\mathbf{X}_s)\|_2^2 + \sum_{s\in\overline{\mathcal{N}_i^k}}\hat{\Phi}(\mathbf{X}_i)^{\top}\hat{\Phi}(\mathbf{X}_s)] + \tau\|\mathbf{H}\mathbf{U}\|_1,\end{aligned}$$

the objective function $\mathcal{G}(\mathbf{U},\boldsymbol{\Gamma},\vec{\beta})$ is multi-convex in terms of $\{\boldsymbol{\Gamma},\mathbf{U},\vec{\beta}\}$.

Proof. Each of the defined functions in $\{\mathcal{J}_{rec},\mathcal{J}_{dis},\mathcal{J}_{ls},\mathcal{J}_{ip}\}$ is convex w.r.t. any individual member of $\{\mathbf{U},\boldsymbol{\Gamma},\vec{\beta}\}$ while the other parameters are fixed. This conclusion is derived because:

1. Matrices $\mathcal{K}_i \;\; \forall i = 1,\dots,d$ are positive semi-definite by definition.
2. The objective $\mathcal{J}_{ls}$ is linear in terms of $\vec{\beta}$.
3. The term $\mathcal{J}_{rec}$ is an F-norm operator.

Therefore, the total objective $\mathcal{G}(\mathbf{U},\boldsymbol{\Gamma},\vec{\beta})$ is multi-convex in terms of $\{\boldsymbol{\Gamma},\mathbf{U},\vec{\beta}\}$. □

A.13 PROOF OF THEOREM 5.1

In Section 5.3, I proposed the following Theorem.

The iterative updating procedure in Algorithm 5.1 converges to a locally optimal point in a limited number of iterations.

Proof. Based on Proposition 5.2 and Theorem 4.1, each optimization sub-problem in Algorithm 5.1 reduces the objective function of Equation 5.11 monotonically. In addition, all the individual objective terms in Equation 5.11 are bounded from below by zero according to their definitions. Therefore, convergence to at least a local minimum solution is guaranteed under a limited number of iterations. □

A.14 COMPLETE ARCHITECTURE OF DACNN FROM SECTION 6.3

The complete architecture of DACNN proposed in Section 6.3 is depicted in Figure A.5, including the detail of building units of Figure 6.3.

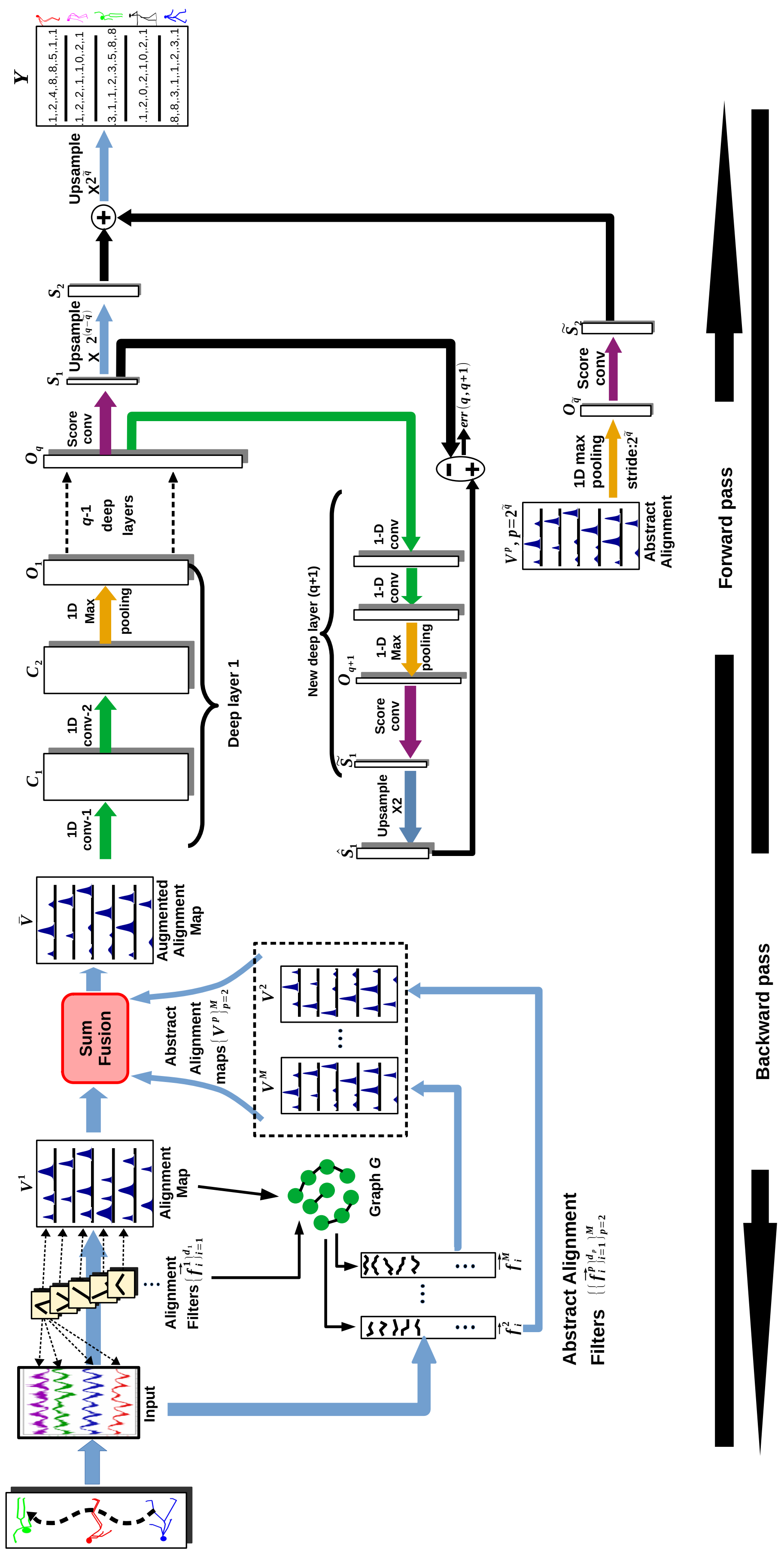


Figure A.5: Detailed description of the DACNN framework, including all of its constituents (better to be seen in colors).