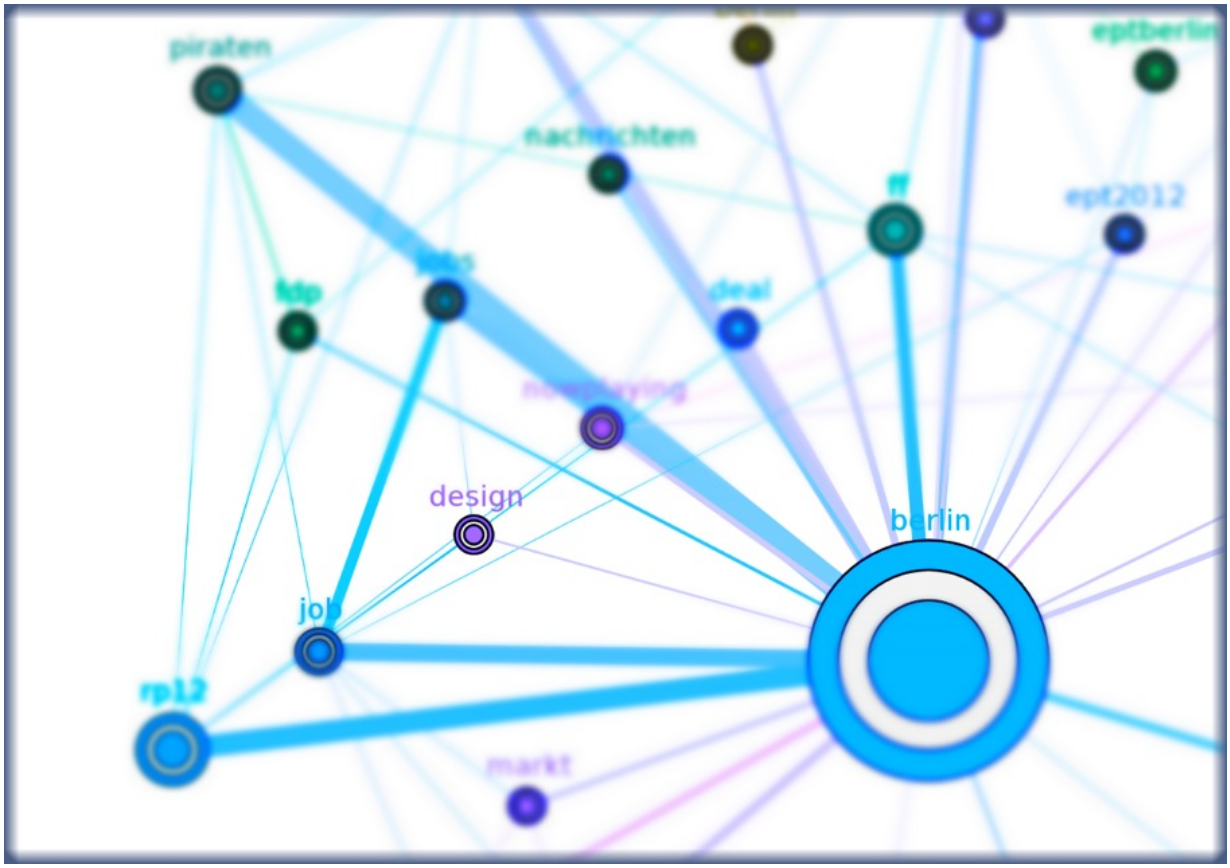
Life-long Learning with

Growing Conceptual Maps

Oliver Beyer



Life-long Learning with Growing Conceptual Maps

Technische Fakultät der Universität Bielefeld

zur Erlangung des Grades

Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt von

Oliver Beyer

Bielefeld - September 2013

Gutachter:
Prof. Dr. Philipp Cimiano
Prof. Dr. Barbara Hammer

Prüfungsausschuss:
Prof. Dr. -Ing. Franz Kummert
Dr. -Ing. Sebastian Wrede

Abstract

A number of challenges arise when transferring the concept of life-long learning to technical systems. This thesis addresses the challenges of learning with continuous data streams, learning with weak supervision, learning with non-stationary data and learning with multi-view data. We introduce the Growing Conceptual Maps Framework as a solution to these challenges, allowing the incremental online learning of a classification model without the requirement of storing training data explicitly. The framework is based on topological feature maps, *i.e.* Growing Neural Gas, and therefore allows a straight forward visualization of the trained model.

With the rapidly increasing amount of data in many domains, such as news feeds, social media, or sensory networks *etc.*, nowadays, assistive systems are required to process a theoretically infinite stream of data in order to help us in our daily tasks. While existing approaches coming from data mining mostly do not scale up to such large and complex tasks, new paradigms are required which allow the model to grow task-dependently and adapt to a changing environment, in order to learn in a life-long fashion.

In this thesis we thus stepwise extend Growing Neural Gas with appropriate novel online labeling and prediction strategies, as well as novel neuron insertion strategies, according to meet these challenges. We evaluate the introduced approaches on benchmarking, artificial and real stream datasets, showing the benefit of our architecture and proving that the Growing Conceptual Maps Framework renders itself ideal for life-long learning by outperforming similar existing approach, and delivering comparable results to other well established classifiers such as a Support Vector Machine. As an application for our framework, we furthermore develop an online human activity classifier based on two Growing Conceptual Maps that can compete with state-of-the-art (offline) human activity classifiers.

As a final contribution, we introduce a straight forward visualization schema for Growing Conceptual Maps that allows the user to track emerging categories and their relation according to the underlying map, and furthermore demonstrate its usability of identifying trends and events in stream data.

Acknowledgements

This is the place to say, Thank you! And i would like to thank so many people that have accompanied me and supported me on my scientific way. Looking back from where i stand now, a huge thank you goes to my supervisor Philipp Cimiano, who helped me and guided me on this way, especially in times when it was difficult to find the right path. He always had an open ear for new ideas and thereby managed to give me a last push into the right direction. I also want to thank Barbara Hammer and Katharina Rohlfing for their advice and for always giving me new perspectives.

I want to thank my colleagues at the Semantic Computing Group, Timo Reuter, Christina Unger, John McCrae, Sebastian Walter, Cord Wiljes and Judith Gaspers for making my journey being such an enjoyable time. A special thank you goes to Roman Klinger for the hours he spent on proofreading my thesis and providing me with most valuable advice. Furthermore, a huge thank you goes to Sascha Griffiths, who always had the right words in times when i needed motivation, and who helped me and supported me in many ways throughout the last years.

Of course, I have to thank all diploma/bachelor students which i had the pleasure to supervise and to benefit from their work: Fabian Vöhl and Carsten Poggemeier. A special thank you goes to Maximilian Panzner, who contributed to expanding the range of applications for the GCM framework and also helped me with the Conceptual Map visualizations.

I want to thank my parents and my brother Torsten for their support and their believe in me over the last years. I also want to thank my band members, Manuel, Christoph, Martin and Markus for their endless patience as i could not show up to so many rehearsals lately. Last but not least, a huge thank you goes to my family, Andrea, Oskar and Anton, as they gave me the strength to persistently pursue my dreams and gifted me their patience, especially in the countless number of hours that i had to contribute. Thank you!

Contents

List of Figures

Chapter 1

Introduction

"Tell me and I forget. Teach me and I remember. Involve me and I learn."
[*Benjamin Franklin*]

[1]

Imagine a future in which service robots assist us in most of our daily tasks. In order to be helpful, such robots are required to be equipped with motor skills as well as communicative abilities in order to act and react with us in our environments. Especially in domains such as healthcare, for example, in which they closely interact with humans, a high degree of reliability in the robots skills is crucial. Therefore, such robots need to be pre-trained in an appropriate way, so they can solve such tasks flawlessly.

However, it is impossible to pre-train them for every possible situation. Instead, it would be desirable that those robots are capable of adapting to new environments and new situations. The robot is confronted with a continuous data stream of information that it observes in the environment. While we as humans to some degree can provide a supervision by teaching the robot something about the relevant objects and mechanics of the unknown environment, the robot is mostly required to continuously learn without a dedicated supervision in a life-long learning process. Furthermore, in contrast to the fixed dataset used for pre-training of the robot, learned concepts constantly evolve and may change over time, which makes the learning task even more challenging.

[1] `http://letsmakerobots.com/node/35773` [All URL links have been accessed at 09/30/2013]

While the described service robots can not be expected to become reality in the very near future, the underlying principles of life-long learning already are highly relevant today. Nowadays, the processing of data streams is crucial in many domains, such as *web search engines* [BDH03], *cognitive robotics* [SJ06], *surveillance systems* [BRC+07], *driver assistant systems* [WLVT06], *ambient intelligence* [NBKL06], *etc.* Furthermore, we live in an age of information where the amount of data coming from domains such as news feeds, social media, or sensory networks, *etc.*, rapidly increases every day [GM05]. While approaches coming from data mining have been well established in recent years, most of them do not scale up to larger and more complex tasks, as they are designed to learn a function for a single isolated task without a broader context. In the following, the characteristics of life-long learning and the resulting challenges when transferring those to a technical system will be discussed in more detail.

1.1. Characteristics of life-long learning

As humans we are capable of learning throughout our entire life. While this ability is so natural to us, it is difficult to find a precise definition of what life-long learning means. Is it just the sum of single isolated learning experiences in our life time? How can we adapt to new tasks relying on our existing knowledge? Do the learned concepts relate to each other? In the area of educational philosophy, different perspectives to this topic have been discussed in the literature. In the following, some relevant aspects of those articles about life-long learning will be discussed:

- **"Constant reorganization or reconstructing of experience"** [DBB85]: For most of the concepts we learn during our life time we permanently see new examples every day, which help us to refine those concepts. Every time we see a "banana", for example, causes us to update our concept of "the banana". It helps us to sharpen concepts or even expand a concept, *e.g.* in case it is a "green banana" while we have only seen "yellow bananas" so far. The ability to update our concept knowledge on-the-fly is crucial, as it allows us to process this huge amount of information we perceive during our life time. This reorganization sometimes even causes us to merge or remove existing concepts. We reorganize our conceptualization of the world as long as we learn and thus as long as we live.
- **Growth and development** [DBB85, Wai87, Bag00]: It has been shown in psychological experiments that we as humans are able to acquire complex concepts, *i.e.* language concepts, based on a single training example [ABM92]. This is a good example demonstrating our ability to generalizing from existing knowledge to a novel concept. Especially infants experience new concepts daily and expand on their knowledge about the relevant properties of those concepts. It takes time until the infant

can identify a “ball” and distinguish it from a “book” or a “chair”. Furthermore, we are capable of transferring already learned properties from one to another concept. Having learned that the “banana” is “yellow” thus helps us to identify the color of a “lemon”, for example.

- **Non-linearity:** It is important to point out that we mostly do not learn concepts in a linear way, meaning concept after concept. Instead, our conceptualizations grow non-linearly in the sense that several concepts and their properties are learned at the same time. To distinguish between a “banana” and a “cherry” also improves our conceptualization of their properties “yellow” and “red”, and the other way around. This stands in contrast to most existing machine learning paradigms, as they are trained and furthermore process data in a linear way. An exception in that sense is given by *recurrent neural networks* [Jor86, Elm90] and algorithms coming from *reservoir computing* [Jae02]. They provide a non-linear signal flow by feeding a transformed input signal back into the network. However, those recurrent and reservoir computing-based algorithms are not equipped with mechanisms allowing them to learn in an incremental fashion and build upon existing knowledge. Nevertheless, non-linear architectures are desirable as a realistic life-long learning environment depends on non-linear dynamics.

1.2. Technical challenges

Following the intuition and characteristics of life-long learning, a number of challenges appear when transferring the concept of life-long learning to a technical system. In recent years, there has been a lot of work in the field of *incremental learning* and *online learning*, which are especially important in the context of large datasets. Those two areas build a foundation for a technical approach to life-long learning as they partially fulfill the characteristics that have been described in the previous section (Section 1.1). Incremental learning, on the one hand, deals with the incorporation of new knowledge into an existing model. Therefore, mechanisms are provided which adapt the existing model accordingly. Usually, the adaptation is done in a *batch mode*, which means that additional storage for training examples needs to be provided. However, algorithms coming from incremental learning are designed to theoretically process an infinite number of data points.

On the other hand, in online learning the model is updated stepwise and does not need to explicitly store training examples, as they are only seen once. However, most algorithms coming from online learning are not designed to handle an infinite data stream and thus do not allow the model to grow according to its task of classifying such data.

In order to follow the intuition of life-long learning as described before, the following challenges arise:

(i) **Learning with a continuous data stream**: In contrast to standard learning and data mining tasks, data streams demand the additional requirements of *one-pass data processing*, *online updates* of the learning model, and in many applications even a *fast response/prediction* on demand.

(ii) **Learning with weak supervision**: In many applications only a limited amount of supervision and expert knowledge is available, which requires the learning algorithm to *make use of labeled, indirectly labeled and unlabeled data in the classification task*. Furthermore, such system should be able to *generalize from categories for which labels are given at a later point in time*.

(iii) **Learning with non-stationary data**: In contrast to closed-world classification scenarios for which the task is to classify a fixed number of static categories, data streams require the learning model to *quickly adapt to changing data distributions that involve concept-drift*, *detecting new evolving categories on-the-fly*, while *regulating its model size* in order to still provide a fast prediction.

(iv) **Learning with multi-view data**: In many data stream applications, data is coming from multiple domains and is provided in different forms and formats. Each of those domains renders a unique view on the data and involves a unique set of subcategories. In order to structure and classify such data, it is required to *learn the relevant subcategories/properties for each category in each view*. Thereby, it is desirable to provide a *compact and uniform representation* in order to maintain the abilities of a fast prediction and memory efficient architecture.

We explain these challenges in more detail in what follows:

(i) Learning with a continuous data stream

Traditionally in machine learning, supervised learning is performed in order to solve a classification task. Thereby, it is assumed that the task is to learn a function $f : X \to Y$ by assigning a finite set of given categories Y to data points from some space X. Data labeled with these categories is used to learn a model for this function that minimizes the empirical risk of making an erroneous assignment. Algorithms based on statistical methods, Artificial Neural Networks (ANN) or Support Vector Machines (SVM) are very effective in that task. However, there are several requirements which uniquely apply to data streams:

- **One-pass data processing**: Standard learning architectures usually require the explicit storage of the complete training data, as data is typically processed in multiple passes. In the process of life-long learning, a theoretically infinite amount of data

and concepts are learned. Therefore, it is impossible to store the data, which leads to the requirement of having to process data in one-pass. This has been relevant in the context of many applications in which we find massive streams of data that cannot be stored on standard hardware anymore (see Gaber *et al.* [GM05]).

- **Online updates**: When processing such stream data in one-pass, updates of the model need to be performed online in order to improve the model gradually. Online clustering algorithms (see Barbakh *et al.* [BF08]) thus become especially relevant in the context of stream data mining, as data cannot be processed in batch mode or in several passes and the model needs to be updated on-the-fly instead. It even has been shown that online learning can render the training more efficient by using the model to generate new training examples which are closer to the desired solution, thus allowing a more efficient exploration of the parameter search space (see Rolf *et al.* [RG11]).

- **Fast response/prediction**: Furthermore, there are scenarios (*e.g.* Interactive Learning) and tracking applications in which not only batch learning is simply unsuitable (see Steil *et al.* [Ste07] and Mandic *et al.* [MG08]), but also a *fast response/prediction* of the model is crucial. Throughout an incremental learning process the model grows and the prediction time increases. Thus, mechanisms are required which reduce and optimize the structure of the model in order to provide fast predictions, while maintaining its classification performance.

(ii) Learning with weak supervision

With the tremendous growth of data in many areas, it is impossible to keep up with labelled information for all of these data. In many cases expert knowledge is necessary to provide a correct and detailed description of the data. While in machine learning one usually distinguishes between supervised and unsupervised learning approaches, techniques from Semi-supervised Learning (SSL) (see Chapelle *et al.* [Cha06]) have blurred the distinction between these two learning paradigms and have become especially interesting as more and more data becomes available of which, however, only a small fraction can be manually labeled due to the high cost incurred. On the one hand, semi-supervised learning has been shown to improve the performance of supervised classification approaches by factoring in unlabeled data (see Nigam *et al.* [Nig00]). On the other hand, semi-supervised learning has also been shown to improve clustering by factoring in labeled data that can be used as constraints to guide the search for an optimal clustering of the data (see Wagstaff *et al.* [WKCCRS01]). However, there are several requirements which uniquely apply to weak supervision in data streams:

- **Classification with labeled and unlabeled data**: Most of semi-supervised classification algorithms introduce an additional training phase to incorporate unlabeled data and thus process data in batch mode. Thereby, the initially trained model is used to label unlabeled data and retrain afterwards. This is unsuitable when it comes to stream data, as the processing needs to be performed in one-pass. Thus, labeled and unlabeled data should be used to update the classifier model in order to still provide a system learning in an incremental online fashion.
- **Generalization from unsupervised categories**: When learning with labeled and unlabeled data, often evolving structures in the model receive a delayed label after having seen many unlabeled examples of the same category before. A learning model is required to propagate those sparsely received labels over the whole model and thus generalize over the underlying categories appropriately.

(iii) Learning with non-stationary data

One of the key challenges in life-long learning is the *plasticity-stability dilemma*, which refers to the ability of acquiring new knowledge (plasticity) while retaining old memory (stability). This can be quite challenging as we live in a rapidly changing world in which information changes quickly. Streams of data nowadays emerge in a number of application domains. The data originating in those applications is often "non-stationary" in the sense that new concepts emerge while others disappear. In those cases, it is often desirable to predict new categories/trends in the data and, therefore, let the trained model follow the data distribution. In those scenarios the following requirements are crucial:

- **Fast adaptation towards changing data distributions**: Yet, machine learning approaches are usually designed for "stationary data" in which the concept distribution is fixed, due to the fact that they rely on prior knowledge about the number of concepts which cannot be assumed to be fixed and given in a life-long learning scenario. In order to process and predict concepts within non-stationary settings, the model needs to *quickly adapt to the new data distribution.*
- **Detection of new concepts on-the-fly**: Typically, in most machine learning tasks only a fixed number of categories are to be learned. However, this mostly does not apply to non-stationary data where new categories may appear while other categories disappear. The learning model is required to adapt to such new evolving categories appropriately.
- **Regulation of resource requirements**: As a theoretically infinite number of data points and categories are required to be processed, it is crucial to regulate the resources of the model in order to still be efficient when processing and predicting new (unseen) data points.

(iv) Learning with multi-view data

Another important aspect of many data stream applications, such as warehouses in e-business [HCD+05], is that the data is provided in several forms and formats. In many cases the desired categories manifest in several different formats, coming from different domains. The data naturally groups into multiple views according to those domains. When classifying the underlying categories, it is often required to consult several views in order to find the relevant properties that set the categories apart. If the task is to classify different kinds of fruits, for example, and the views of "color", "shape" and "taste" are given, then it would be insufficient to distinguish between a "banana" and a "lemon" only by their color. Instead, also their properties in the domain of "shape" or "taste" are crucial to discriminate the two categories. However, in order to tackle these challenges, the following requirements need to be fulfilled:

- **Learning of relevant properties**: In order to provide a sufficient description of such multi-view categories, it is required to learn properties in each of the relevant views. While usually such a classification task decomposes into a per-view classification, the compositional nature of the underlying categories requires to learn their properties/subcategories at the same time. Such properties thus emerge parallel in the classification model while being processed and therefore grow non-linearly.
- **Compact and uniform representation**: In order to provide a system that is capable of scaling up to a life-long learning task, it is desirable to represent information coming from several views in an uniform and compact way, *i.e.* in one model. The advantage of such a model lies in the fact that it allows to maintain the ability of a fast predictions, while providing a memory efficient classification model that represents categories coming from multiple views.

1.3. A basic model for life-long learning

In order to tackle the challenges of life-long learning, a basic learning architecture is required to build upon. This architecture needs to fulfill the properties of life-long learning and therefore be an incremental online learning algorithm. *Growing Neural Gas (GNG)*, as introduced by Fritzke [Fri95], is an algorithm which partially fulfills these requirements. It belongs to the family of topological feature maps, such as *Self-organizing Maps (SOM)* [Koh82], *Neural Gas (NG)* [Mar91] and *Learning Vector Quantization (LVQ)* [Koh95]. These feature maps are artificial neural networks and rely on the Hebbian learning rule, which renders them ideally for online learning tasks. Thereby, categories are prototypically represented by a number of neurons that allow a straightforward visualization and thus facilitates the interpretation of the model by humans. It is also the reason why topological

feature maps are also used in visualization tasks (see WebSOM [KT00]), besides clustering and classification. However, for most of these algorithms the network size is fixed and needs to be determined in advance. In contrast, GNG provides a growing structure and thereby includes mechanisms for inserting, merging and removing neurons and is thus idealy suited for incremental online learning. In particular, GNG's life-long learning properties can be listed as follows:

- The Network grows according to the data distribution.
- A data manifold is represented by a neuron distribution.
- Model updates are performed online.
- Synaptic links determine the relation between neurons.
- Neurons and links are inserted and removed according to their relevance.
- A storage of the training data is not necessary.

However, GNG is a clustering algorithm and thus cannot solve a classification task, as category labels are simply ignored. Therefore, this thesis focuses on the stepwise extension of GNG to meet the discussed requirements of life-long learning. In particular, the following extensions are required to apply GNG to a life-long classification task:

1. **Online classification**: GNG provides an online clustering architecture that allows the grouping of similar structures found in continuous data streams. However, in order to apply GNG to most life-long learning applications, it is required to also incorporate category label information and thus extend GNG to an online classifier.
2. **Semi-supervised Learning**: With the prior extension of GNG to an online classifier, unlabeled examples only contribute to the clustering process. In order to fully utilize all available data for the classification task, the labels for those examples need to be predicted.
3. **Non-stationary data**: Although GNG is capable of adapting to any data manifold, it is not designed to follow a rapidly changing data distribution. New mechanisms are required to let the network grow and adapt according to the changing data distribution and thereby solving the "plasticity-stability-dilemma" in the best possible way.
4. **Multi-view data**: A GNG-based classifier can represent many categories in a compact form. Representing multi-view data coming from multiple domains would usually require the use of one GNG per view and would increase the complexity of the architecture dramatically. In order to still provide a compact model in form of a single GNG network, strategies are required to incorporate the label information coming from those multiple views.

1.4. Contributions

In this thesis *Growing Conceptual Maps Framework* as framework for life-long learning is introduced (see Figure 1.1), providing solutions for the discussed challenges in the previous sections. In order to do so, the Growing Conceptual Maps framework is based on Growing Neural Gas and extends it accordingly to meet the requirements described earlier in this section. In particular, the following contributions are provided:

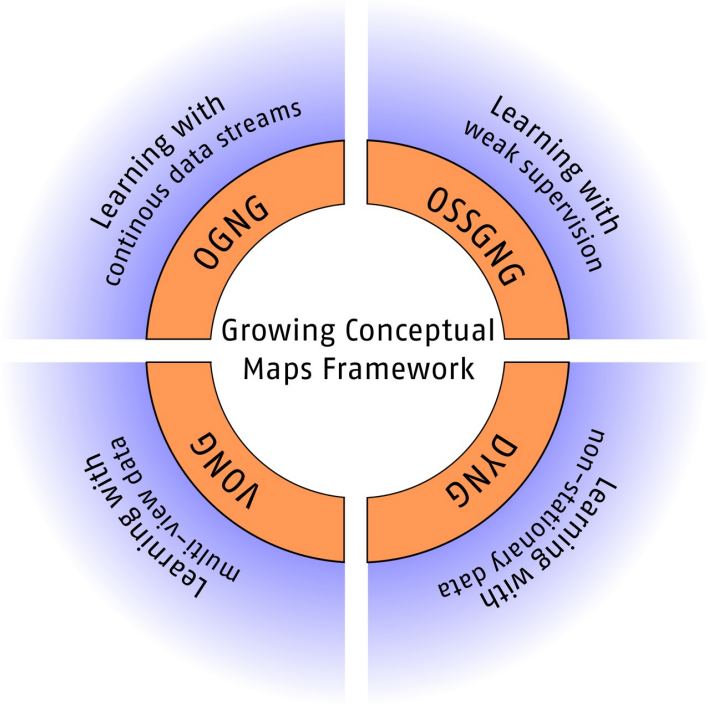


Figure 1.1.: Growing Conceptual Maps Framework.

- **Learning with a continuous data stream:** An algorithm is introduced that extends Growing Neural Gas to an online classifier (i.e., *Online Growing Neural Gas (OGNG)*) by an additional step that uses an appropriate labeling function to assign or re-compute the labels of neurons, as well as an appropriate prediction function that allows to assign labels to unseen data. It will be shown that these labeling and prediction functions do not deteriorate the classification performance when compared to corresponding offline methods.
- **Learning with weak supervision:** *Online Semi-supervised Growing Neural Gas (OSSGNG))* is introduced as extension of Online Growing Neural Gas. OSSGNG predicts labels for unlabeled examples and incorporates these data points into the model

on-the-fly. It will be shown that OSSGNG outperforms existing semi-supervised extensions of GNG, as well as OGNG on semi-supervised benchmark datasets. We will furthermore show that OSSGNG is competitive towards standard SSL classifiers.

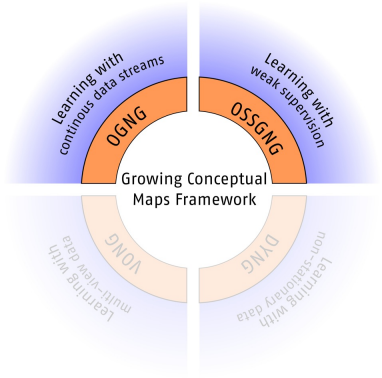
- **Learning with non-stationary data:** *Dynamic Online Growing Neural Gas (DYNG)* is introduced as extension of OGNG. DYNG utilizes category label information and introduces two novel neuron insertion strategies in order to grow with its task as long as its classification performance increases. It will be shown that DYNG better solves the plasticity-stability dilemma compared to an online classifier based on an existing GNG extension (*i.e.* Growing Neural Gas with Utility [Fri97]), in that it provides a better performance while retaining a relatively small network. Furthermore, we will show that DYNG outperforms OGNG and a linear multi-class SVM, when using a comparable amount of memory.

- **Learning with multi-view data:** *Multi-view Online Growing Neural Gas (VONG)* is introduced as multi-view extension of OGNG that is capable of learning with data coming from multiple domains and, furthermore, uniformly stores these data in a single compact network. We will show that VONG outperforms a naive multi-label extension of OGNG ($OGNG^{+}$), while being capable of predicting category labels for each view independently.

These extensions of GNG are not limited to GNG, but could also be applied to similar architectures fulfilling the mentioned GNG properties.

1.5. Components of the Growing Conceptual Maps Framework (Overview)

In this section, a listed overview of each component of the Growing Conceptual Maps framework is given, including its challenges, its contributions, the involved datasets and evaluations, and a list of publications in which it already has been published.



Learning with continuous data streams

Challenges

- one-pass data processing
- online model updates
- fast response/prediction

Online Growing Neural Gas (OGNG)

- extends GNG to incremental online classifier
- additional labeling and prediction strategies

Datasets

- SPIRAL1, ART (artifical datasets)
- SEG, ORL (benchmark sets)
- ReutersRCV1v2, TwitterLL (real stream datasets)

Evaluation

- Online vs. offline labeling strategies
- OGNG vs. (offline) GNG classifier vs. SVM
- OGNG vs. OGNG^{top}

Publications

- Online labeling strategies for Growing Neural Gas [BC11a, BC12]
- Human Activity Classification with OGNG [PBC13]

Learning with weak supervision

Challenges

- processing labeled and unlabeled data for classification
- generalization from unsupervised categories

Online Semi-supervised Growing Neural Gas (OSSGNG)

- extends OGNG to semi-supervised classifier
- utilizes unlabeled data for its classification model

Datasets

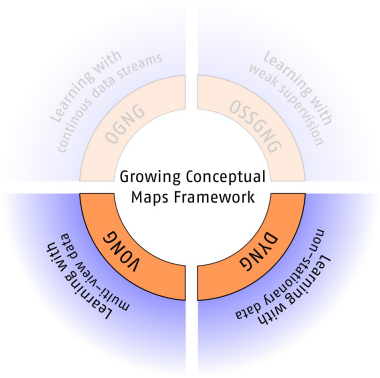
- SPIRAL1 (artifical datasets)
- g241c, g241d, Digit1 (artifical SSL benchmark sets)
- USPS, COIL, BCI (real SSL benchmark sets)

Evaluation

- OSSGNG vs. OGNG vs. SVM
- OSSGNG vs. standard SSL algorithms
- Late labeling of categories

Publications

- Online Semi-supervised Growing Neural Gas [BC11b, BC12]



Learning with non-stationary data

Challenges

- fast adaptation towards changing data distributions
- detection of new categories on-the-fly
- regulation of resource requirements

Dynamic OGNG (DYNG)

- extends OGNG to an online classifier for non-stationary data and concept drift
- two additional neuron insertion strategies
- performance tracking for categories in the network

Datasets

- SPIRAL1 (artifical dataset)
- ReutersRCV1v2, TwitterLL (real stream datasets)

Evaluation

- DYNG vs. OGNG vs. SVM vs. 1NN
- DYNG vs. DYNG^{-imp}

Publications

- DYNG: Dynamic Online Growing Neural Gas for Stream Data [BC13]

Learning with multi-view data

Challenges

- learning with multiple views
- grouping of categories by views
- compact uniform classification model

Multi-view OGNG (VONG)

- extends OGNG to multi-view classifier
- masking of relevant dimensions per view
- additional multi-view labeling/prediction strategy

Datasets

- SPIRAL2 (artifical multi-view dataset)
- OBJ (artificial object description dataset)

Evaluation

- VONG vs. OGNG^{+}
- VONG vs. VONG-M

1.6. Outline

This thesis is structured as follows: In Chapter 2, detailed information about Growing Neural Gas and its existing variations and extensions is given. Furthermore, an overview over existing work towards learning with data streams, semi-supervised learning, learning with non-stationary data and learning with multi-view data is given.

In Chapter 3, Online Growing Neural Gas (OGNG) is introduced as first extension of GNG. With OGNG a number of online labeling and prediction strategies are introduced and compared to existing offline labeling strategies. On four datasets, we will show that the online labeling strategies do not deteriorate the classification performance compared to offline labeling strategies, with the benefit of no additional required storage. Furthermore, we will show that OGNG even significantly outperforms a linear multi-class SVM on two of the datasets used for evaluation.

In Chapter 4, Online Semi-supervised Growing Neural Gas (OSSGNG) is introduced as semi-supervised extension of OGNG. OSSGNG is capable of using labelled and unlabeled data for classification by extending OGNG with an additional label prediction step. On an artificial dataset and six standard semi-supervised learning datasets we will show that OSSGNG outperforms existing semi-supervised GNG-based approaches, *i.e. Semi-supervised Growing Neural Gas (SSGNG)* (see Section 4.3.1) and furthermore delivers comparable results compared to state-of-the-art semi-supervised learning approaches.

In Chapter 5, Dynamic Online Growing Neural Gas (DYNG) as extension of OGNG is introduced. The advantage of DYNG lies in its ability to quickly adapt to non-stationary data that also involves a changing concept distribution. Experiments with non-stationary stream datasets show that DYNG outperforms OGNG, especially in highly non-stationary phases when a large number of new categories is introduced. Furthermore, our evaluation will show that DYNG better solves the plasticity-stability dilemma compared to OGNG-U, an online classifier based on GNG-U, by providing a better classification performance while retaining a compact model.

In Chapter 6, Multi-view Online Growing Neural Gas (VONG) as extension of OGNG is introduced. VONG is capable of solving a multi-view classification problem by introducing a novel multi-view labeling and multi-view prediction strategy and thereby maintaining a compact and uniform representation. On two artificial multi-view datasets, it will be shown that VONG clearly outperforms a naive OGNG-based multi-view classifier (OGNG$^+$. We will furthermore show that, for the labeling as well as for the prediction strategy, VONG's classification performance benefits from a weighted masking of the relevant dimensions per view.

In Chapter 7, the Growing Conceptual Maps framework is applied to three datasets from the domain of text document stream data classification and human activity classification.

In particular, we will evaluate the classification performance of DYNG on the document stream data and show that DYNG even outperforms a linear multi-class SVM under comparable memory usage. Furthermore, a new GCM-based online human activity classifier will be introduced and evaluated on a standard human activity dataset. We will experimentally show that the classification performance of this novel approach is comparable to those of an existing state-of-the-art (offline) human activity classifier. In addition, a visualization schema for GCM-based networks will be introduced and we will demonstrate its utility be applying it to the networks used in the applications of this chapter.

In Chapter 8, we summarize our results and conclude. In an outlook we will furthermore discuss possible GCM extensions, as well as possible new application areas for the GCM framework.

Chapter 2

Related Work

In this chapter we will give an overview of the research fields that are highly related to artificial cognitive systems which focus on methodologies for life-long learning. Thereby, we will highlight established solutions in each of these areas and also explain how our work builds upon those existing approaches and how it sets itself apart. The purpose of this chapter is to give the reader a broad overview of existing relevant research, while algorithms that are closely related to each part of our framework will be discussed in each chapter. Furthermore, a detailed description of the approaches we compare with, *i.e.* Semi-supervised Growing Neural Gas (SSGNG) (see Section 4.3.1) and Growing Neural Gas with utility (GNG-U) (see Section 5.2.1) will be available in their related chapters.

In this chapter, at first, topological feature maps will be explained, as our framework builds upon this family of algorithms due to their applicability in life-long learning scenarios. In particular, we will discuss Growing Neural Gas (GNG) [Fri95], its growing design and its extensions that have been proposed over recent years. We then discuss the challenges of learning with continuous data streams, including the most known solutions in the area of stream data clustering and stream data classification. After that, we will move onto the area of semi-supervised learning methods and describe their three main classes of algorithms, namely Generative models, Low-Density Separation and Graph-based Methods. Thereby, also data assumptions will be discussed which are necessary for semi-supervised learning in order to be effective. Then we discuss approaches that focus on the processing of data which involves non-stationarity and concept-drift, in the sense that the data as well as the category distribution changes over time. Finally, we discuss the problem of multi-view data, a scenario in which the challenge is to classify data originating from different domains.

2.1. Topological feature maps

Approaches based on topological feature maps, *e.g. Self-Organizing Maps (SOMs)* [Koh82], *Learning Vector Quantization (LVQ)* [Koh95], *Neural Gas (NG)* [Mar91], or *Growing Neural Gas (GNG)* [Fri95], have been successfully applied to clustering problems by representing a high-dimensional input space in a low-dimensional feature map. An important property of this family of algorithms is that they can be visualized straightforwardly by projecting the map into a two-dimensional space or, as shown in Chapter 7, by a simple interpretation of its topology. Growing Neural Gas, for example, when used with unlabeled data, will learn "natural categories" and thus the inherent topology of the data in an incremental fashion. GNG features the advantages of unsupervised approaches that can learn categories for which no labeled data is given. Approaches such as GNG are ideal in life-long learning settings where neither the categories can be assumed to be fixed a priori, nor labels can be assumed to be available for all categories. Topological maps also have been successfully applied to a number of *visualization tasks* in several domains [Hyo96, KT00, FV08, HF08] as the neurons, which represent prototypes, are easy to understand and interpret. Like SOM and NG, GNG is a *Competitive Learning* approach based on the *winner-takes-it-all (WTA)* principle. Therefore, in every iteration step the algorithm determines the neuron which is closest to the presented input stimulus and adapts it accordingly. In Figure 2.1, a two-dimensional visualization of GNG is shown as an example of such topological feature maps. The image shows the evolution and adaptation of a GNG network towards an input distribution, which is indicated by the blue ring. The network starts with two connected neurons and gradually increases its number of neurons after seen 0-4000 stimuli.

2.1.1. Growing Neural Gas (GNG)

Growing Neural Gas is an incremental self-organizing approach which is capable of representing a high-dimensional input space in a low dimensional feature map. Although the main idea behind SOM, NG and GNG is similar, there are some important differences which set GNG apart. First of all, Growing Neural Gas combines the ideas of *Growing Cell Structures (GCS)* [Fri94] and *Competitive Hebbian Learning (CHL)* [Mar93]. It shares the growing character of GCS in the sense that, starting from a small network, neurons are successively inserted into the network and can also be removed if they are identified as being superfluous. This is an advantage compared to SOM and NG, as there is no need of fixing the network size in advance. Inspired by CHL, GNG also integrates temporal *synaptic links* between neurons, which are introduced between a winner neuron and a second winner neuron. These links are temporal in the sense that they are subject to

[2] `http://www.demogng.de/JavaPaper/node19.html`

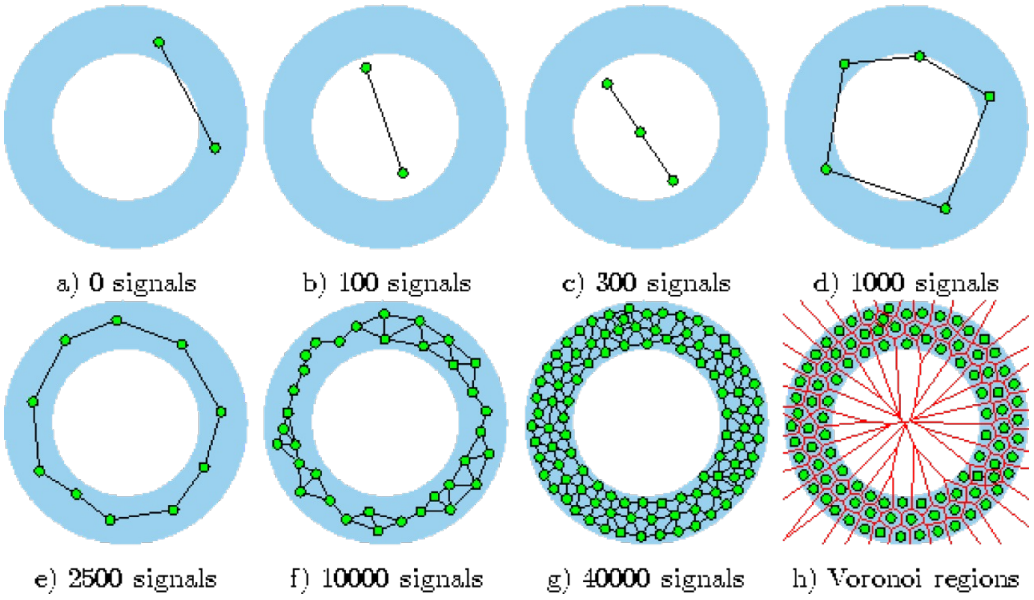


Figure 2.1.: Two-dimensional representation of a GNG learning sequence.[2]

aging during the iteration steps of the algorithm and are removed when they get "too old". The main difference compared to SOM and NG is the fact that the adaptation strength of the network is constant over time and fixed by the two parameters e_b and e_n, *i.e.* the adaptation strength for the winner neuron and its neighbors, respectively. Furthermore, only the best-matching neuron and its topological neighbors are adapted, such that there is no global optimization of the network.

In the following, we will briefly describe the single steps of the GNG algorithm as proposed by Fritzke [Fri95]. The algorithm is depicted in Algorithm 1.

1. In the first step, the algorithm starts with two neurons, randomly placed in the feature space.

2. The first stimulus $x \in R^n$ of the input space (first training example) is presented to the network.

3. The two neurons n_1 and n_2 which minimize the Euclidean distance to x are identified as first and second winner, respectively.

4. The age of all edges that connect n_1 to other neurons is increased by 1.

5. The local error variable $\Delta error(n_1)$ of n_1 is updated. This error variable will be used later in order to identify the position for a newly inserted neuron.

Algorithm 1 Growing Neural Gas (GNG)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ from the input set or according to input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: Increment the age of all edges emanating from n_1.
5: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

6: Move n_1 and all its topological neighbors (*i.e.* all the nodes connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1})$$

$$\Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
7: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
8: Remove edges with their age larger than a_{max}. If this results in nodes having no emanating edges, remove them as well.
9: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r as follows:
Determine unit n_q with the largest error.
Among the neighbors of n_q, find node n_f with the largest error.
Insert a new node n_r halfway between n_q and n_f as follows:

$$\omega_{n_r} = \frac{\omega_{n_q} + \omega_{n_f}}{2}$$

Create edges between n_r and n_q, and n_r and n_f. Remove the edge between n_q and n_f.
Decrease the error variable of n_q and n_f by multiplying them with a constant α. Set the error n_r with the new error variable of n_q.
10: Decrease all error variables of all nodes n_i by a factor β.
11: If the stopping criterion is not met, go back to step (2).

6. In this step, n_1 and its topological neighbors are adapted towards x by fractions e_b and e_n, respectively.

7. According to this step, a new connection between n_1 and n_2 is created and the age of the edge is set to 0.

8. All edges with an age greater than a_{max} as well as all neurons without any connecting edge are removed.

9. In this step, depending on the iteration and the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.

10. The error variables of all nodes are decreased by a factor β.

11. In the last step, the algorithm stops if the stopping criterion is met, *i.e.*, the maximal network size or some other performance measure has been reached.

2.1.2. Extensions of GNG

In recent years, a number of variations of Growing Neural Gas have been published, improving its ability to deal with noisy data (*Robust Growing Neural Gas (RGNG)* [QS04]) and by a hierarchy on top of GNG (*TreeGNG* [DAD05]). Growing Neural Gas has also been extended towards specific types of data, *i.e.* non-stationary data (*Growing Neural Gas with Utility (GNG-U)* [Fri97]), labelled data (*Supervised Growing Neural Gas (SGNG)* [JA07]) and weakly labelled data (*Semi-supervised Growing Neural Gas (SSGNG)* [Zak08], *Incremental Growing Neural Gas (IGNG)* [PE05]). Although these extensions address relevant aspects, they are unsuitable for life-long learning scenarios as they do not meet its requirements. In the following, we will describe each of these GNG extensions briefly:

Robust Growing Neural Gas (RONG)

This algorithm extends the standard GNG with a number of a strategies to improve its robustness towards noisy data. As first extension, the GNG update rule is modified in order to be less sensitive towards noisy data. As part of the modified update rule an iteration-dependent maximal adaptation strength is introduced to limit the force of outliers in the adaption phase of GNG. We use a similar method in Section 3.4.1 to limit the adaptation range for the labeling of neurons. A second modification is introduced to increase the robustness of GNG by its adaptive learning rates and repulsion scheme. In RONG, the learning rates of neurons decrease incrementally to force the convergence of older neurons in the network. Therefore, an individual adaptation rate for each neuron is introduced, which depends on their "age" in the network as well as on the overall number of neurons and a predefined threshold. The adaptive learning rate tends to result in the stagnation of neurons close to each other, which means that those neurons adapt towards and stabilize at the same position in feature space. In order to work against this side effect, an additional repulsing scheme is used to decrease the adaptation amount for the winners' neighboring neurons to an extent proportional to their closeness to this winner. Finally, RONG introduces an optimization step that determines the optimal number of neurons by minimizing the description length of the network. A more detailed explanation can be found in Quin and Suganthan [QS04].

Although RONG has been proven to improve the classification performance of GNG, it renders itself unsuitable for life-long learning due to the requirements of each of those additions. In the first two additions, a fixed maximal network size is required which cannot be assumed in a life-long learning process, as the network should grow and evolve alongside with the continuous data it processes. Also the optimization of the network size is difficult to be presumed, as the data is only partially available. In general, it is legitimate

to claim that the GNG extensions and modifications of RONG are specifically designed for fixed datasets and even work counter against a continuous learning scenario.

TreeGNG

Doherty *et al.* [DAD05] introduce a tree structure, which follows the development of GNG and represents a history of the evolving subclusters inside of its network. The main idea of TreeGNG is to provide a tree, similar to a *Hierarchical Agglomerative Clustering (HAC)* [ZMRA13] approach that represents the underlying hierarchy of the input data and is generated on-the-fly, as it builds upon GNG. In contrast to HAC, TreeGNG is build top-down in the sense that it starts with a single root node and then successively introduces new tree nodes and branches while following GNGs' topology. The tree is updated regularly after a certain number of iterations and relies on a growth and pruning mechanism to follow GNGs' evolution. The growth mechanism is triggered when a subcluster of GNG is divided into two, due the removal of the last connecting link between both subclusters and their neurons, respectively. A second mechanism, the pruning mechanism, is applied to the tree in case of a newly established link between neurons of subclusters which have not been connected before. The shape of the tree, as produced by the TreeGNG algorithm, depends on the specified tree update frequency and the maximal allowed age of an edge of GNG.

However, as TreeGNG only reflects the clustering history of GNG without incorporating external information about category labels, it cannot be assured that the tree really reflects the underlying semantic hierarchy of the data. It is thus only of limited use in a classification task.

Growing Neural Gas with Utility (GNG-U)

Growing Neural Gas with Utility (GNG-U) is designed to process and follow non-stationary data. Non-stationary data is a phenomena which often appears in the context of data streams and basically involves the (often unpredictable) change of the data distribution. It thus is highly relevant for life-long learning. GNG-U introduces a utility factor for each neuron that indicates how important each neuron is for its subcluster in the network. The utility factor is updated for a winner neuron n_1 by accumulating the difference between its local error and the local error of the second winner neuron n_2. The utility thus implicitly reflects in how far the local error of the second winner n_2 would increase after removing the first winner neuron n_1. However, a newly inserted neuron starts with a utility factor of $u = 1$. In contrast to GNG, GNG-U only removes neurons if their utility falls under a predefined fraction of the worst local error at the given stage of the network development.

In Section 5.2.1 of Chapter 5, we will explain GNG-U in more detail. In Chapter 5, we will introduce Dynamic Online Growing Neural Gas (DYNG) and experimentally show that DYNG better solves the plasticity-stability dilemma (see Section 2.4) as it quickly adapts to a changing distribution, providing a better classification performance while retaining a compact model.

Supervised Growing Neural (SGNG)

Jirayusakul and Auwatanamongkol introduced a GNG extensions called *Supervised Growing Neural Gas (SGNG)*, which incorporates label information to guide the clustering. The main difference between GNG and SGNG lies in SGNG's novel update rule that adopts the LVQ2 technique [Koh90], as well as integrating the already discussed adaptive learning rates and repulsing schema from RONG. Following this technique, SGNG adapts the winner neuron towards the stimulus, in case of matching labels of neuron and stimulus, and adapts them away from the stimulus otherwise. Initially, the SGNG network basically starts with a number of neurons equal to the number of categories in the training set, with each neuron being uniquely labeled with one of the category labels. SGNG then gradually inserts neurons, similar to GNG, until a threshold is met which determines the maximal allowed number of neurons. The authors of SGNG also propose an optimization of the clustering which depends on the *validity index*. This index sets compactness and impurity of the clustering in relation to the cluster separation to achieve an optimal balance between them.

Besides the inclusion of RONG extensions, which we already discussed as being unsuitable for life-long learning, SGNG has a strong assumption concerning the relation between data distribution and category distribution. It assumes the *cluster assumption* (see Section 2.3) to fulfilled. This assumption, which will be described in more detail later in this chapter, requires the data manifold and category manifold to match, so that basically a single cluster represents a single category. This is not necessarily the case, especially when real datasets are involved. In such scenarios, categories can possibly be *non-convex* and thus be represented by several not neighboring clusters in the feature space, for which SGNG would most probably perform poorly in a classification task. In Chapter 3, we will experimentally show that letting the category labels guide the clustering can significantly deteriorate the classification performance of a GNG-based classifier. Another disadvantage is the fact that SGNG assumes all category labels to be known in advance, which is mostly not the case in a life-long learning scenario in which categories my change over time.

Semi-supervised Growing Neural Gas (SSGNG)

The incorporation of labeled and unlabeled data in a semi-supervised fashion has been introduced by Zaki *et al.* with *Semi-supervised Growing Neural Gas (SSGNG)* [Zak08]. SSGNG is inspired by the *Expectation-Maximization (EM)* approach [Har58] and thus iteratively processes the data in two separate phases. The algorithm first performs a clustering, which essentially is identical to the GNG clustering, only using the labeled examples of the training set and then labels all neurons a posteriori. In a second step, it then predicts the labels of all unlabeled training examples and retrains the classifier based on the new dataset. The algorithm iterates over these two steps until the labels for the initially unlabeled training set stabilize, *i.e.* remain unchanged. A more complete description of SSGNG and its algorithm is given in Chapter 4.

The main disadvantage of SSGNG is the fact that a retraining of the model requires the training data to be stored. This contradicts the online nature of GNG and, furthermore, is unsuitable in a life-long learning task in which a theoretically infinite number of training data is required to be processed. In Chapter 4, we will introduce Online Semi-supervised Growing Neural Gas (OSSGNG) and furthermore compare SSGNG to OSSGNG, showing that OSSGNG outperforms SSGNG on the used datasets, while providing the benefit of utilizing labeled and unlabeled data on-the-fly without the need of storing any training data.

2.2. Learning with continuous data streams

In recent years, advances in hardware technology allow us to continuously collect a huge amount of data. No matter if we browse through the internet, use our mobile phones or build up our digital photo and music library, the data that is required to be stored increases daily. However, these data is only of use to us if it is organized in a way that allows us to quickly retrieve the relevant information task-dependently. While the field of data-mining has been well established, algorithms coming from this field mostly do not scale up to solve the challenges of clustering and classifying stream data. In this section, the challenges of stream data processing or *stream mining* are discussed. We will give an overview of the relevant methodologies that have been developed in the context of stream data.

2.2.1. Challenges & Methodologies

The models and algorithms of stream data processing can be roughly divided into *stream data clustering* and *stream data classification*. However, there are several algorithms com-

ing from both areas that focus on optimizing the arising challenges. In this section we will thus at first discuss the challenges of stream data classification that arise in the context of continuous data streams, and then give an overview of the popular solutions in that area. For each of these approaches we will discuss in how far they have addressed the introduced challenges.

Challenges of stream data classification

1. **One-pass processing constraint:** The continuous growth of data in many applications leads to an extensive storage of these data on large volumes. Due to the size of these volumes, it is impossible to allow the data mining to process data points multiple times, instead, a one-pass processing architecture is required. The one-pass processing constraint is already challenging for most machine learning approaches, as an optimal classification model in most cases cannot be provided by an analytical solution of the prediction function $f : X \rightarrow Y$. Instead, usually optimization algorithms such as *gradient descent* are required to adapt the model stepwise to a more optimal solution through multiple passes of the data. However, there are algorithms, such as GNG, that are capable of processing data in one-pass and thus are ideal to build upon in a stream data scenario. In Chapter 3, we will introduce Online Growing Neural Gas (OGNG) as a solution to this problem.

2. **Evolving data:** While in standard learning scenarios a fixed dataset can be assumed, stream data often is *non-stationary* in the sense that it changes its distribution over time and therefore its evolution is difficult to be predicted. The phenomena of non-stationary data mostly appears as result of the *temporal locality* of the data, which means that only small parts of the data are available at a time. Furthermore, also the underlying categories cannot be assumed of be fixed in those scenarios. Such *concept-drift* requires the classification model to quickly adapt to the new emerging categories and their data distribution. We will discuss the different methodologies about this topic in Section 2.4. Furthermore, in Chapter 5, we will introduce Dynamic Online Growing Neural Gas (DYNG) to approach concept-drift and evolving data distributions in non-stationary data.

3. **Limited time:** When processing stream data, only a small fraction of the theoretically infinite corpus is available at a certain point in time. This not only requires the classification model to process each data point in one-pass, as already explained, but also needs the whole adaptation of the model and its prediction to be fast. In particular, adaptation and prediction should meet the speed in which new data points arrive, in order to handle the data stream in the long term. In Chapter 3-6, we will show that the extensions of our framework are linear in the number of data items (neurons).

4. **Limited memory:** Processing a theoretically infinite data stream also requires to efficiently and exclusively store necessary information, and furthermore avoid the explicit storage of data points from the data stream. However, it is also desirable to refine and optimize the classification model in a way that complex categories are modeled in more detail, compared to more simple categories. As a side effect, a more compact model also helps to speed up its adaptation and prediction time. In Chapter 3, we will introduce online labeling and prediction strategies and experimentally show that in our architecture the least memory demanding strategy does not significantly deteriorate the classification performance of the model. Furthermore, in Chapter 5, we develop a neuron insertion strategy that lets the classification model grow as long as its performance increases.

5. **Noisy data:** Due to the nature of stream data, it is impossible to provide a supervision through expert knowledge about the involved categories for every data point in the stream. Mostly, in those scenarios only a weak supervision is suitable and given by a heuristic approach rather than a human expert, which potentially is noisy. In those cases, partial supervision is given in many different forms, such as inter-categorial constraints, semi-supervision with a relatively small amount of available category labels, delayed labeling, or even a constant supervision for only a few of the involved categories. In Chapter 4, we will introduce the semi-supervised approach Online Semi-supervised Growing Neural Gas (OSSGNG) that utilizes both labeled and unlabeled data for its classification model, and furthermore quickly adapts to categories for which the labels are given in delay. In Section 2.3 of this chapter, we will discuss approaches that focus on these topics in more detail.

6. **Visualization:** In many applications the aspect of detecting trends in data streams and visualizing their development is even more important than the classification task itself. In order to do so, the classification model is required to be very uniformly structured so that it can quickly be visualized on demand. In Chapter 7, we will demonstrate that one of the main advantages of our framework is that its underlying *Conceptual Map* can be visualized straightforwardly, and is useful in order to track a data stream and to show the relation of the involved categories in a simple two-dimensional map.

In what follows, we will give a brief overview of algorithms and architectures that have been proposed in the field and partially solve the listed challenges of data stream clustering and data stream classification. We will also discuss in how far the algorithms address the challenges of stream data classification.

Stream data clustering

Clustering is a widely studied field in data mining. However, adapting arbitrary clustering algorithms to meet the stream data clustering challenges is difficult, as the one-pass data processing restriction contradicts their learning architecture. Stream clustering decomposes in two classes of algorithms: *flat stream clustering* and *hierarchical stream clustering*. In each of those classes, algorithms have been developed trying to tackle the challenges of life-long learning. In the following, the principles of these two classes as well as their advantages and limitations will be discussed.

- **Flat stream clustering**: Approaches such as STREAM [GMMO00] build upon the principle of k-means. For a given set of data points and a given number of k centroids, it partitions the data into k clusters. As the complete data set is not available when processing stream data, STREAM is capable of solving the k-means clustering problem incrementally, following the principle of divide-and-conquer in order to partition the data into n small data subsets. The size of those subsets depends on the available memory and can be extended at any time. STREAM then uses the $n \times k$ clusters as input for the next clustering iteration as shown in Figure 2.2 (left). The advantages of STREAM lie in its ability of processing an infinite stream of data in one-pass while generating a compact model that consists of k prototypes which represent the current clusters distribution of the data. STREAM thus contributes to the Challenge 1, Challenge 3 and Challenge 4 according to our list of challenges. However, its still limited due to its requirement of a predefined k which cannot be given in a life-long learning setting.

- **Hierarchical stream clustering**: Hierarchical stream clustering approaches, such as BIRCH [ZRL96], provide a non-flat representation of the underlying data. The advantage of such a structure lies in its independence of a predefined fixed number of clusters. BIRCH provides an architecture that it is based on *Cluster Features (CF)* [ZRL96] which consist of tuples $CF_i = (N_i, LS_i^x, SS_i^x)$, with N_i being the number, LS_i^x being the linear sum and SS_i^x being the square sum of all processed data points at iteration i. Cluster features are usually organized in a height-balanced *CF tree* as illustrated in Figure 2.2 (right), and depend on a branching parameter B and a threshold T. Thereby, T defines the maximal radius allowed for each CF-based subcluster. The most interesting aspect of CF, besides its compact representation, is its *additive property*. Assuming $CF_1 = (N_1, LS_1^x, SS_1^x)$ and $C_2 = (N_2, LS_2^x, SS_2^x)$ being initial leaf nodes. According to the additive property, a cluster CF_3, which includes CF_1 and CF_2, is then defined as $CF_3 = (N_1 + N_2, LS_1^x + LS_2^x, SS_1^x + SS_2^x)$. The BIRCH algorithm makes use of cluster features and starts its procedure with building a CF tree using the available memory, in a first step. It then scans the leaf nodes and removes outliers, in order to provide a more memory efficient model,

in a second step. In a third step, an existing clustering approach (often HAC) is applied to the CF nodes in order to not be effected by the order of which the data is presented. Other closely related approaches such as *DenStream* [CEQZ06] and *SDStream* [RM09] make use of an extended version of CF called *Micro-Cluster (MC)*, which also include the sum of timestamps LS^t and the sum of squared timestamps SS^t into its vector to $MC_i = \{N_i, LS_i^x, SS_i^x, LS_i^t, SS_I^t\}$. These additions allow the creation and recall of temporal snapshots in which the actual data distribution gets stored, relying on the additive and *subtractive property*.

However, all of those algorithms share the advantage of cluster features/micro-clusters, in particular, its compact representation and its additive/subtractive property that allows to merge, extend and recall clusters by simple addition or subtraction of the CF/MC. The algorithms thereby avoid the need of multiple scans of the actual data and thus scan the data in an online phase and process CF/MC in an offline phase. BIRCH and the CF/MC related approaches thus focus on Challenge 1, Challenge 3, and Challenge 4. Due to its additional clustering in the third step, BIRCH also contributes to Challenge 5 as it increases the insensitivity of BIRCH towards noisy data.

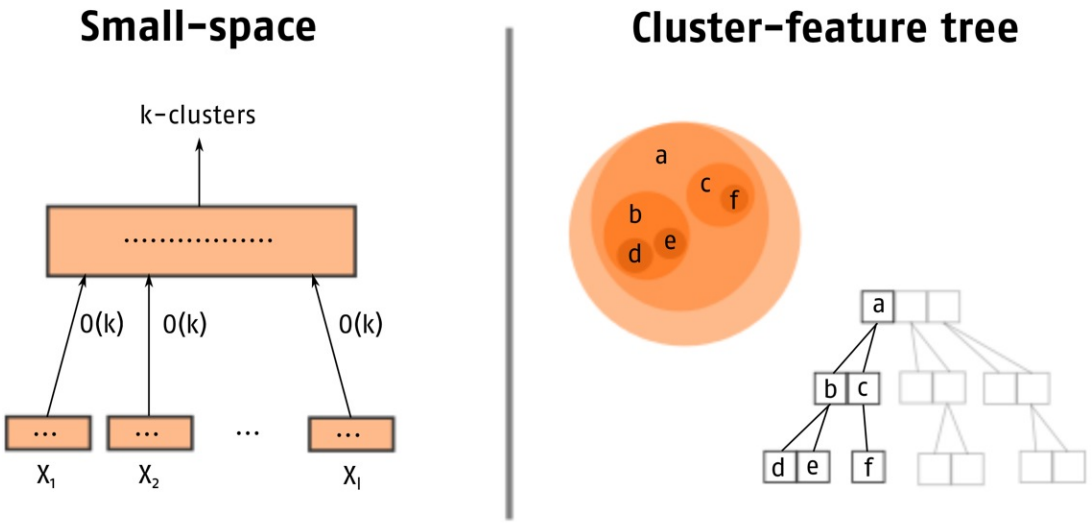


Figure 2.2.: Principles of stream clustering algorithms (left: STREAM, right: CF tree).

Stream data classification

The problem of classification is probably the most widely studied topic in data mining and stream data mining. Also in this field the adaptation of standard classification algorithms

is difficult, especially due to the one-pass restriction. However, there are some extensions of existing classification approaches, as well as extensions of clustering approaches, which have been successfully applied to stream data classification problems. Some of those algorithms will be discussed in what follows. Another important challenge for stream classification can be found in non-stationary data streams. As mentioned before, we will discuss this topic in Section 2.4 of this chapter. In this section, we distinguish between three common groups of data stream classification algorithms as follows:

- **Approaches based on existing classification models:** To some extend existing classification approaches have been modified in order to serve as a classification model for stream data. Very fast decision trees (VFDT) [YFSW12] are a good example of such approaches, as they incrementally build a decision tree by adding and splitting tree nodes depending on a small amount of the incoming stream data. The algorithm builds upon the principle of a *Hoeffding tree (HT)* and thus grows by holding the so called *Hoeffding bound*, which stands for the information gain of each attribute. The most interesting aspect of the algorithm is the fact that it performs a training and prediction process simultaneously. While traversing through the tree in order to determine the most similar leaf node, it splits nodes depending on accumulated statistics. The main disadvantage of this approach lies in the fact that the Hoeffding bound is required to be estimated from a given dataset which thus needs to be stored. According to our list of challenges, VFDT solves Challenge 3 and Challenge 4, but cannot fulfill Challenge 1 according to the determination of the Hoeffding bound.

- **Approaches based on stream clustering:** Also stream clustering algorithms have been extended to classification tasks. One commonly used stream classification approach is called *On-demand-classification* [AHWY04]. This architecture is based on the concept of micro-clusters, which have been explained previously in this section. Besides their additive property that helps to quickly merge clusters, they also ensure a *subtractive property*. This property allows to quickly retrieve micro-clusters from a requested time window/snapshot. Then the nearest neighbor determines the predictive cluster. This means that in a classification task one process constantly clusters the available data into a tree of micro-clusters, while a second process can make a prediction for unseen data points using the subtractive property. While on-demand-classification is very fast, its performance is highly dependent on the horizon of the micro-cluster tree. In case of non-convex categories, a high granularity of the category representation for those categories is required. The horizon of the micro-cluster tree then highly effects the classification performance.

- **Novel Approaches:** Another widely used algorithm is *LWClass* [CKMS03], which is based on the *Algorithm Output Granularity*. The interesting fact about this algorithm is that it uses the complete available memory in order to build a matrix that summarizes the original data. Each row holds a category, a contribution weight and

A_1	A_2	...	A_n	Category	Weight
$Value(A_1)$	$Value(A_2)$	...	$Value(A_n)$	Category label	X = #items contributing
...	...	...	...	...	

Table 2.1.: Schema of a LWClass Matrix.

the averaged attributes of the input data, as shown in Table 2.1. A predefined threshold is responsible for the clustering, as all data points are assigned to the category to which their Euclidean distance is below the threshold. Thereby, the contribution weight is increased or decreased depending on a matching or mismatching label. Unseen data points are then predicted by the weighted majority label of the k nearest neighbors. Thereby, the contribution weight marks the relevance of the specific row and thus influences its vote. The main disadvantage of this architecture lies in the predefined clustering threshold and the parameter k. While in principle the adaptive weights in each row of the matrix are similar to the weights for each neuron, the clustering requires a quantization threshold which is difficult to be provided in an evolving stream data setting. Also the determination of a fixed k for the prediction seems impossible in a changing environment.

2.2.2. Domain-specific solutions

In recent years, there have been approaches in statistics, machine learning and data mining, which focus on the online classification of data streams. Especially, in the e-business sector they help organizations to perform market analysis, monitoring, predict future trends/behaviors and make pro-active knowledge driven decisions. Tools such as the Social Media Matrix [KKK12] and the CLOUDwerker [KKK12] have been developed in order to coordinate and support social media activities of business companies. Conventional *Online Analytical Processing (OLAP)* [CD97] and data mining models have been extended in order to be capable to process large data streams, with the goal of capturing trends and patterns in the data and furthermore visualizing them. However, a lot of expert knowledge is necessary in order to use those systems, besides the fact that they need to be adjusted for every task individually. A system would be desirable that fulfills the requirements of stream data classification and does not depend on such expert knowledge.

2.3. Learning with weak supervision

In recent years, there has been substantial work in the area of semi-supervised learning, both in the context of classification and clustering tasks. Those approaches have been successfully applied to a number of applications such as *text classification*, *pattern recognition* and *medical diagnosis* [Joa, Zak08, CBV11]. In the following, we will give an overview of the different classes of algorithms coming from semi-supervised learning, as well as describing the data assumptions that are necessary in order for semi-supervised learning to be successfully. In the end of this section, we will furthermore discuss the general scope of weak supervision and late labeling as one special case of it.

2.3.1. Categories of SSL Methods

One can distinguish between three main classes of semi-supervised learning approaches: *Generative Models, Low-Density Separation* and *Graph-based Methods*. In what follows, we will describe those classes and, in particular, describe popular approaches belonging to them.

- **Generative models**: Generative models involve the estimation of the conditional density $p(x|y)$, with $p(x)$ being the density of the input space and $p(y)$ being the density of the category label space. Typically, the EM algorithm is applied to estimate the parameters of the Gaussian distribution for each category. Existing semi-supervised extensions of GNG [Zak08], in fact, build on the EM principle and thus process labeled and unlabeled data in two separate steps. Approaches such as SSGNG and *Co-Training* proposed by Blum *et al.* [Blu98] belong to this class of semi-supervised learning approaches. In contrast, our approach relies on a single (online) step that predicts labels for new examples and incorporates them into the existing model on-the-fly.
- **Low-Density Separation**: Approaches based on a Low-Density Separation such as *Transductive SVM (TSVM)* [Joa] make use of the unlabeled data to iteratively maximize the margin using labeled and unlabeled data points. Therefore, the TSVM is initially trained with only labeled examples and increases the amount of unlabeled data points iteratively.
- **Graph-based Methods**: The third class of semi-supervised learning algorithms are Graph-based Methods (see Belkin *et al.* [Bel04]). These approaches organize labeled and unlabeled data points as nodes in a graph. The edges in the graph represent the similarity between the single nodes and are thus labeled with the distance between them. Missing distances are typically approximated by the minimal aggregated path over all paths connecting those two nodes.

2.3.2. Data assumptions

In this subsection we will describe the data assumptions according to Chapelle *et al.* [Cha06], which are required to be fulfilled for semi-supervised learning to be effective. These assumptions are in particular the *semi-supervised smoothness assumption*, the *cluster assumption* and the *manifold assumption*.

Semi-supervised smoothness assumption

- *Semi-supervised Learning assumption:* If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.

This assumption addresses the relation of the data points x_1, x_2 in feature space and their classes y_1, y_2 in the category space. It implies that category space and feature space form a similar manifold.

Cluster assumption

- *Cluster assumption:* If points are in the same cluster, they are likely to be of the same class.

Similar to the semi-supervised learning assumption, this assumption demands the similarity of the two manifolds. However, this assumption even goes further as it demands the decision boundary to lie in a low-density region. It should be mentioned that this does not imply that each category is only represented by one cluster.

Manifold assumption

- *Manifold assumption:* The (high-dimensional) data lie (roughly) on a low-dimensional manifold.

The last assumption demands the data to not be effected by the "curse of dimensionality" [Bel56]. In this case, the curse of dimensionality refers to a phenomena that data originating from a high-dimensional feature space cannot to be reasonable represented in a low-dimensional manifold. As a consequence, the increase of dimensions causes a (often exponential) growth of the number of representatives in the model.

2.3.3. Weak supervision and late labeling

Learning with weak supervision has recently become a topic of interest in the machine learning community. A lot of frameworks are emerging since, including *Semi-supervised Learning* [Zak08, ZSH13], *Constrained-based Clustering* [BBD00], *Multi-instance and Multi-label Learning* [ZHM+08], *Transfer Learning* [JTC11] and *Multi-task Learning* [EP07], just to name a few. As the discussion of all of those frameworks is out of scope for this thesis, we focus on semi-supervised learning and *late label learning* as two scenarios within the area of weak supervision.

In many applications, such as stream data classification, only a small amount of supervision in form of category labels is available. The learning algorithm therefore is required to mostly learn without such labels during its learning process. In those cases, it is desirable that the learning algorithm can utilize the already learned patterns of the category, instead of relearning an already learned category from scratch after encountering the first label of it which is given in delay. We consider late label learning as a scenario in which categories are presented to the learner without an explicit label for the involved category. Those labels are given at a later point in time during the learning process. The challenge is to make use of those delayed given labels while assigning them to the correct category, and furthermore utilizing the already learned structures of the category in order to quickly generalize over them.

2.4. Learning with non-stationary data

In this section, we will focus on the topic of non-stationary data and concept-drift. Although this topic is very important for the domain of stream data and life-long learning in general, it is not addressed by many stream data classification algorithms. Especially the algorithms described in the previous section (Section 2.3) are not capable of following a changing data and category distribution. At first we will describe the challenges and then discuss some of the most relevant approaches in this field. Furthermore, we will discuss approaches which have been inspired by biological learning models and solve the problem of non-stationary data by providing a two-level memory-layer, including a short-term memory (STM) and a long-term memory (LTM).

2.4.1. Challenges & Methodologies

In what follows, we highlight some challenges which partially have been addressed in the previous section, but this time with an emphasis on the aspect of concept-drift.

Challenges of non-stationary data

1. **Concept-drift:** Many existing classifiers are not capable of handling the phenomenon of concept-drift, in which basically the relevant set of relevant categories changes over time. A changing category-distribution typically is connected to a change in the underlying data distribution as well. While some algorithms are potentially capable of adapting to non-stationary data, many algorithms such as STREAM assume the number of categories to be fix and given, which usually is not possible in such data.
2. **Efficency:** The creation and update of a classification model for non-stationary data can be complex and complicated due to the concept-drift. The classification model needs to remain simple in the sense that it can quickly be adapted and evaluated while following non-stationary data, in order to keep up with the data stream in a life-long learning process.
3. **Plasticity and stability:** When quickly adapting to non-stationary data and concept-drift, the classification model is required to retain its stability towards generalization over the already learned data and to avoid overfitting. A balance between the learning flexibility of the model and its robustness is required.

In Chapter 5, we will introduce Dynamic Online Growing Neural Gas (DYNG) as a solution to these problems, as DYNG is capable of processing non-stationary data for which the concepts drift. We will furthermore show that DYNG provides a better solution to the plasticity-stability dilemma than existing GNG-based approaches for non-stationary data.

Classification of non-stationary data

The first classifier framework we will have a look at is *Ensemble-Based Classification* [WFYH03]. Ensemble-based approaches have become quiet popular in machine learning, as they can be applied to a wide variety of applications. The main idea of the framework is to combine a number of classifiers $Class_1, Class_2, ...Class_n$ in a weighted linear combination $EClass(x_i) = \omega_1 Class_1(x_i) + \omega_2 Class_2(x_i) + ... + \omega_n Class_n(x_i)$, so that each classifier $Class_j$ contributes to the overall output when evaluating the input x_i. As proposed by Wang *et al.* [WFYH03], each of the weights is dynamically adapted according to the individual performance of the single classifier. They thus change over time and lead to a higher robustness of the classifier. Although the approaches partially address Challenge 1 and Challenge 3, they are limited in the sense that their efficiency is highly dependent on the used classifiers. Furthermore, the selection of classifiers and their number is highly task-dependent. It is desirable to provide a more uniform and generic structure which bypasses the requirement of a domain-specific selection of classifiers.

The second approach which we will discuss was proposed by Last and is called *Online Information Network (OLIN)* [Las02]. This online classifier consists of a tree-based *Info-Fuzzy Network (IFN)* classification model, and relies, in contrast to standard decision trees, on the *conditional mutual information* of the given data and their corresponding categories. However, the main idea of OLIN is to rebuild the classification model depending on a classification error rate. This error rate increases or decreases the window size of OLIN, depending on its stability. In other words this means that OLIN always uses the most recent data to rebuild its classification model. As soon as it gets more stable, the error rate decreases and the window size increases, for which the model is rebuilt less frequently. OLIN addresses Challenge 1, but there are several issues with Challenge 2 and Challenge 3. First of all, depending on the frequency for which a concept-drift occurs, OLIN will remain unstable. This sensitivity towards non-stationary data makes it less robust and it is valid to claim that OLIN is imbalanced in terms of the plasticity-stability dilemma. Another disadvantage of OLIN is the fact that the complete model is rebuilt frequently. This requires the explicit storage of data according to the window size, which if not regulated leads to inefficiency in memory and processing time. In Chapter 5, we will show that our approach DYNG gradually grows without the requirement of recreating the classifier, and furthermore does not require the explicit storage of input data.

2.4.2. Layering of memory

In this section, we will address another closely related concept of dealing with the plasticity-stability dilemma, which mostly can be found in the area of *Cognitive Robotics* [BDFS10, RP02, KWK05]. The algorithms coming from that research field share the two-layered memory architecture that consists of a *short-term memory (STM)* and a *long-term memory (LTM)*. Both of the memory structures are sequentially connected in the sense that the STM quickly adapts to new evolving categories and data then manifests in the LTM over time.

These approaches usually share the fact that the STM provides a rapid mapping between the sensory input and a categorial concept. This mechanism is also known as *fast mapping* in cognitive science. The LTM then builds upon the mappings coming from the STM in order to manifest categories in the model over time. In the work of Bellas *et al.* [BDFS10], an autonomous agent incrementally builds up three different memories, *i.e. World Model Memory, Internal Model Memory, Strategy Memory* in order to interact in a changing environment. The agent performs actions in the environment and perceives a feedback coming from the environment. Those action-perception pairs are stored in the STM and then further processed in the LTM, which basically is formed by the three memory types. Roy and Pentland [RP02], approach the topic of *Symbol Grounding* [Har90] in their work when building a system that anchors word meanings to audio-visual events relying on a

STM and LTM. Thereby, the STM stores the pairings of spoken utterances and audio-visual events. The LTM then builds upon those prototypical hypotheses and generates a dictionary of lexical items and corresponding audio-visual events, based on the mutual information of the pairings of the STM. Kirstein *et al.* [KWK05] introduce a hierarchical approach to learn objects from visual features. The hierarchical approach is based on shape and color features and extracts and stores template vectors out of the images in the STM, of which show the object in multiple views. The LTM then adapts object prototypes based on the vectors of the STM by using an *Incremental Learning Vector Quantization* approach [XFHZ09].

In Chapter 5, we will discuss the implicit two-layered memory architecture of DYNG. DYNG also features a STM and a LTM, with the difference that those memory structures are not explicit and manifest in the different neuron insertion and removal strategies. DYNG thus keeps its uniform architecture that is beneficial in life-long learning settings, as already discussed.

2.5. Learning with multi-view data

With the rapid growth of multimedia data, multi-view learning algorithms become more interesting. In traditional multi-view learning, the feature vector of a domain is composed by disjoint feature subsets (views) that are sufficient to learn a target category. Thereby, it is assumed that the views are conditionally independent, which basically means that given the label of any example its description in each view is independent of each other. As each view is assumed to be sufficient to learn the target category, the multi-view challenge can also bee seen as a semi-supervised learning task in which the labels are only partially given. Probably the most famous algorithm in that area is Co-Training as introduced by Blum & Mitchel [BM98]. Co-Training is a semi-supervised two-view classification which trains a classifier per view using labeled and unlabeled data, by combining the classifiers category hypothesis in order to improve each individual classification performance. Therefore, a (weak) classifier is initially trained with the labeled data in each view and then applied to the unlabeled data. The most confident predictions of a view are then added as newly labeled examples to the other view to iteratively improve each classifier. A prediction is then performed by a combined category voting coming from both classifiers. Another closely related approach is *Co-EM* proposed by Nigam & Ghani [NG00]. While it follows the same principles, in each view an EM algorithm is applied as classifier. The main difference of Co-EM compared to Co-Training is the fact the Co-EM does not commit to a label for unlabeled data points, but instead relies on probabilistic labels. In order to maintain the assumption of conditional view independence, Christoudias *et al.* [CUD12]

introduce a multi-view learning framework which detects view disagreements and filters the corresponding data points before applying a standard multi-view learning algorithm.

However, we extend the traditional multi-view learning challenge in that we allow each view to be part of a different domain, including multiple labels for each data instance. As a result, the conditional independence of each view cannot be assumed anymore. Instead, in our scenario each view can be seen as unique property describing the data instance. This also changes the learning task, as we do not predict a single label but a set of labels for each instance instead. In Chapter 6, we will introduce Multi-view Online Growing Neural Gas (VONG) as an online classifier which is capable of learning with multi-view data and predicts a label for an unseen data point per view. We will show that VONG outperforms similar multi-view approaches on a dataset for which the views are completely uncorrelated, as well as for a dataset in which they are partially correlated.

2.6. Summary

In this section, we have given an overview of the different challenges of life-long learning, as well as over topological feature maps. In particular, we discussed the Growing Neural Gas algorithm as part of the topological feature maps family, and also discussed its growing Hebbian learning-based architecture. We described existing extensions of GNG which aim on increasing its robustness, as well as extending it to be capable of processing labeled data, weakly labeled data and non-stationary data. Furthermore, we discussed the challenges of learning with continuous streams and provided an overview of the most commonly used stream clustering and stream classification algorithms. We also discussed some domain-specific solutions that have been established in e-business in recent years. We explained the different classes of semi-supervised learning algorithms and described data assumptions that are usually required for those algorithms to be effective. As a meta challenge to semi-supervised learning, we discussed weak supervision and introduced the special case of late label learning. We furthermore gave an overview of challenges and algorithms coming from the area of non-stationary data and involving concept-drift. Thereby, we explained the plasticity-stability dilemma and described biologically inspired algorithms that provide a short-term and long-term memory. In the last part of this section, we discussed the multi-view learning challenge and furthermore extended this problem.

Chapter 3

Classification of continuous data streams based on OGNG

In this chapter, we introduce the *Online Growing Neural Gas (OGNG)* algorithm, in order to extend Growing Neural Gas (GNG) [Fri95] to a classifier that is capable of learning from continuous data streams. GNG has been successfully applied to unsupervised learning problems. However, GNG-inspired approaches can also be applied to classification problems, provided they are extended with an appropriate labeling function. Most approaches along these lines have so far relied on strategies which label neurons a posteriori, after the training has been completed. As a consequence, such approaches require the training data to be stored until the labeling phase, which runs directly counter to the online nature of GNG. Thus, in order to restore the online property of classification approaches based on GNG, we present an approach in which the labeling is performed online. This online labeling strategy better matches the online nature of GNG where only neurons – but no explicit training examples – are stored.

To our knowledge, there has been no systematic investigation and comparison of different offline strategies so far, a gap we intended to fill. The question of how GNG can be extended to an online classification algorithm has also not been addressed previously. In most cases, offline strategies have been considered that perform the labeling after the training phase has ended and the network has stabilized to some extent as in the WEBSOM [KT00, LK99] and LabelSOM [Rau99] approaches. In both of these approaches, the label assignment is essentially determined by the distance of the labelled training datapoint to the neurons of the already trained network. Such offline labeling strategies contradict the online nature of GNG, whose interesting properties are that the network grows over time and only neurons, but no explicit examples, need to be stored in the network. As the main contribution, it will be shown that online labeling strategies do not deteriorate the performance compared to offline labeling strategies.

3.1. Challenges

In a classification task a model $\mathcal{M}$ is trained to provide a solution for the function $f : X \rightarrow C$, with X being the feature input space and the corresponding category space C. Thereby, $\mathcal{M}$ estimates the conditional density $p(X|C)$ with $p(X)$ being the density of the feature space and $p(C)$ being the density of the category space. The function f can be seen as prediction function for a stimulus $x_i \in X$ for which no corresponding category label $c_i \in C$ of the category space is given. In stream data classification tasks, however, X and C are only given by the subsets $X^t \subset X$ and $C^t \subset C$ at a certain point in time t during the learning progress. This causes the learning algorithm to incorporate X^t and C^t for $t = 1 \ldots n$, with n not being fixed. In the following we will focus on the challenges that arise out of this stream data classification scenario:

- **One-pass data processing**: Many incremental learning algorithms incorporate new data in a batch learning mode, in which every $x_i \in X^t$ is processed in multiple passes. This requires the explicit storage of one or more X^t (and their corresponding C^t) and is unsuitable for stream data classification. Instead, data should be processed in one pass in order to scale up to complex stream data classification tasks. While GNG is capable of processing data points $x_i \in X^t$ in one-pass, mechanisms are required to process the corresponding category information.

- **Online updates**: The immediate processing of X^t in one-pass causes the model to adapt stepwise as well. The adaptation of the GNG network is based on the hebbian learning rule and performed multiple times (for winner neurons n_1, n_2 and the topological neighborhood of n_1) in every iteration step. In order to update the classification model based on GNG, category labels coming from C^t should be assigned to the neurons stepwise, according to the models current estimate of $p(X|C)$.

- **Fast response/prediction**: Online updates of the classification model allow the classifier to predict unseen data points or *i.e.* $x_i \in X^t$ for which no corresponding $c_i \in C^t$ are given at any point in time. However, in order to provide a fast prediction, the model needs to be simple in the sense that it can be evaluated quickly and deliver a c_i for an unseen x_i. GNG provides an architecture in which the winner neuron n_1 to an input stimulus x_i can be determined in $\mathcal{O}(n)$ with n being the number of neurons included in the GNG network. In order to make use of this model architecture in classification terms, prediction strategies are required that can deliver a category label in $\mathcal{O}(n)$ as well.

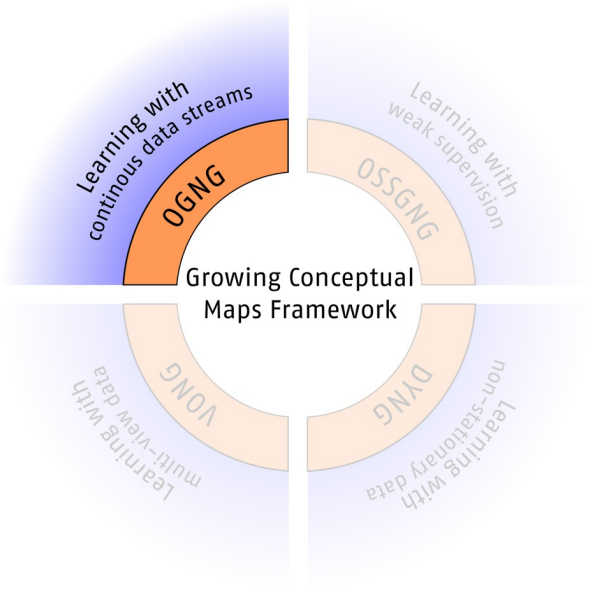


Figure 3.1.: Online Growing Neural Gas (OGNG) as component of the GCM framework.

3.2. Contributions

In this chapter, we present and evaluate several strategies allowing to perform the labeling on-the-fly, thus extending GNG to an online classification algorithm. We compare these strategies to several offline strategies that have been proposed in the literature and examine in particular whether an online labeling strategy can compete in terms of performance, *i.e.* classification accuracy, with an a posteriori labeling strategy. In fact, we show on four datasets (two artificial and two standard datasets) that an online labeling strategy does not perform significantly worse compared to an offline labeling strategy. We instead show that for one of the artificial datasets, *i.e.* SPIRAL1, we achieve better results than a linear multi-class SVM.

In particular, we offer the following contributions:

- **Offline labeling strategies**: We systematically evaluate different offline labeling strategies for GNG in a classification task.
- **Online labeling and prediction strategies**: We extend GNG by an on-the-fly labeling and prediction step that allows us to extend GNG to an online classification algorithm.

- **Comparison of online and offline labeling**: We present and systematically analyze a number of online labeling strategies and compare them to the offline labeling strategies, showing that they do not deteriorate the performance of a classification approach based on GNG.
- **OGNG and SVM comparison**: In order to compare our approach to a standard classifier, we compare OGNG with a linear multi-class SVM and show that OGNG delivers a comparable performance and even outperforms the SVM on two dataset.
- **Category guided topology**: We will compare OGNG to a OGNG variation (OGNG^{top}) which adapts its network topology according to the underlying category distribution, in order to show in how far the connection of neuron distribution and category distribution effects its classification performance.

This chapter includes parts which have already been presented and published at the *International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)* [BC11a] in 2011, as well as in the *International Journal of Neural Systems* [BC12] in 2012.

3.3. Classification with GNG

In order to apply self-organizing approaches to classification problems, two extensions are necessary: i) a function which assigns labels to neurons and ii) a function that performs the prediction on unseen datapoints. So far, mainly offline labeling strategies which require the explicit storage of labelled training data have been considered for the first case [Hyo96, Rau99, LK99, LK06, LG08]. They perform the assignment of labels to neurons a posteriori, after the training phase has been ended. The difference between an offline GNG-based architecture compared to an online GNG-based architecture is depicted in Figure 3.2. In the upper image, we see the offline architecture and that the data is required to be stored in a data buffer. The GNG network is then trained with this data buffer in multiple passes afterwards. In case of the online GNG-based architecture, no additional memory storage is required as each datapoint is processed immediately in one-pass, as shown in the lower image.

Another important requirement for a GNG classifier in general is the ability to assign labels to unseen data points and thereby predict the class of the new datapoint according to the learned network model. While this step does not differ in the online and offline classifier version, it is important to stress that only the online architecture is capable of predicting new data on-the-fly. This is crucial in many life-long learning applications, as it allows the classifier to make a prediction even during the processing of new data at any time. The offline architecture, instead, requires the model to finish its training phase before prediction.

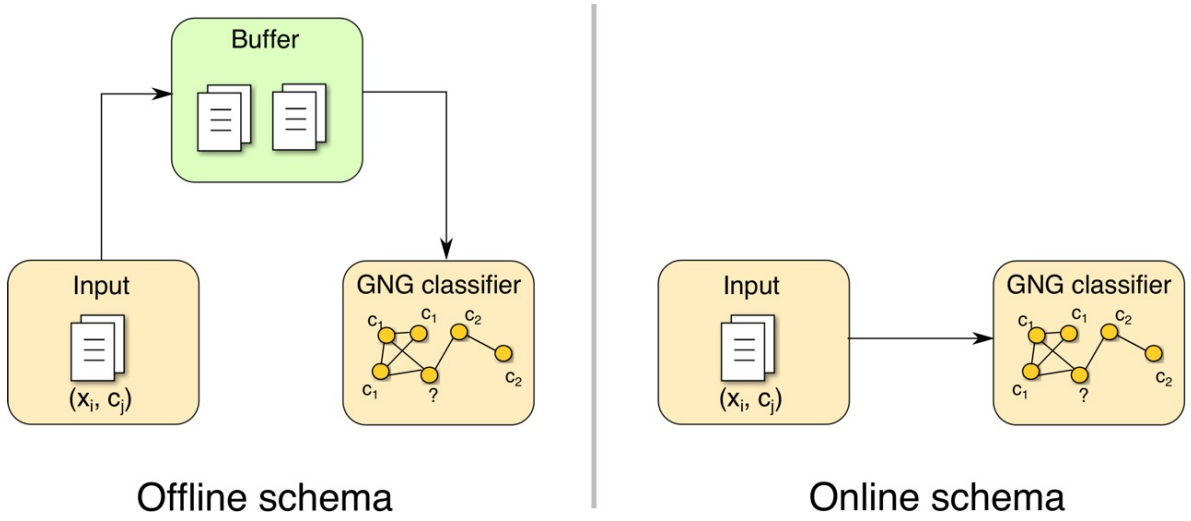


Figure 3.2.: Simplified offline vs. online GNG-based architecture.

In the following, standard offline labeling strategies for GNG are defined, which can be found in the literature. Additionally, we define prediction strategies that have been inspired by cluster analyses.

3.3.1. Offline labeling strategies

In order to apply GNG to a classification task, most approaches in the literature extend the algorithm by two functions. A neuron labeling function $l : N \to C$ where C is the set of category labels, and a prediction function $f : X \to C$ where X is the input space. We analyze the following offline neuron labeling functions as proposed by Lau *et al.* [LK06]. They are offline in the sense that they assume that the pairs (x, l_x) with $x \in X_{train} \subseteq X$ and $l_x \in C$ seen in the training phase are explicitly stored. The offline labeling strategies are depicted in Figure 3.3.

Minimal-distance method (min-dist)

In this first offline strategy, the label of the presented stimulus is simply assigned to the closest neuron in the network. It is the most commonly used labeling strategy for topological feature maps in general. According to this strategy, neuron n_i adopts the label l_x of the closest datapoint $x \in X_{train}$:

$$l_{min-dist}(n_i) = l_x = l(\arg \min_{x \in X_{train}} |n_i - x|^2)$$

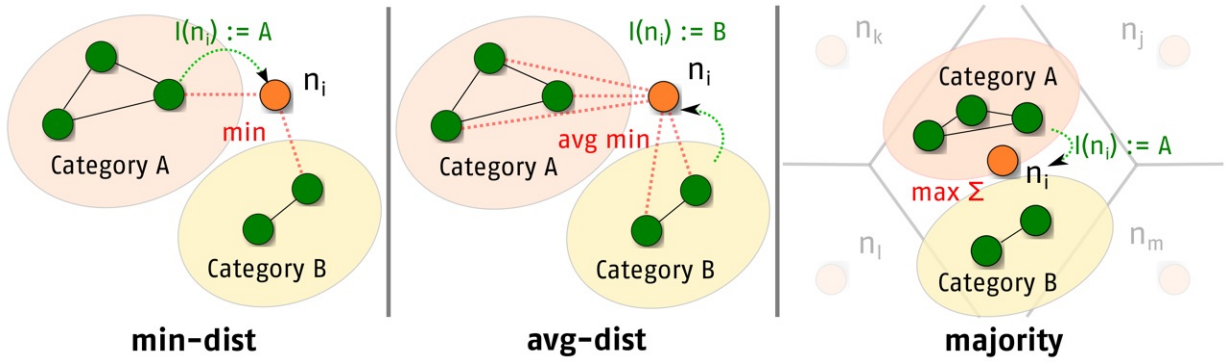


Figure 3.3.: Offline GNG-based labeling strategies.

Average-distance method (avg-dist)

The second strategy differs compared to *min-dist* in that the label of a neuron is determined by taking all data points of one category into account, instead of just the closest. Therefore, it would not be possible to transfer this strategy into an online version, as the complete training data is required. According to this strategy, we assign to neuron n_i the label of the category c that minimizes the average distance to all data points labelled with category c:

$$l_{avg-dist}(n_i) = \arg\min_c \sum_{k=1}^{|X(c)|} \frac{|n_i - x_k|^2}{|X(c)|}$$

where $X(c) = \{x \in X_{train} \mid l(x) = c\}$ is the set of all examples labelled with c.

Majority method (majority)

For the last offline labeling strategy, the neurons of the neural network divide the input space into a Voronoi tessellation. Thereby, each neuron represents the center of a Voronoi cell, as shown in Figure 3.3. According to this strategy, we label neuron n_i with that category c having the highest overlap (in terms of data points labelled with c) with the data points in the Voronoi cell for n_i. We denote the set of data points in the Voronoi cell for n_i as $v(n_i) = \{x \in X_{train} \mid \forall n_j, j \neq i : |n_j - x|^2 \geq |n_i - x|^2\}$ within the topological map, *i.e.*

$$l_{majority}(n_i) = \arg\max_c |D(c) \cap v(n_i)|$$

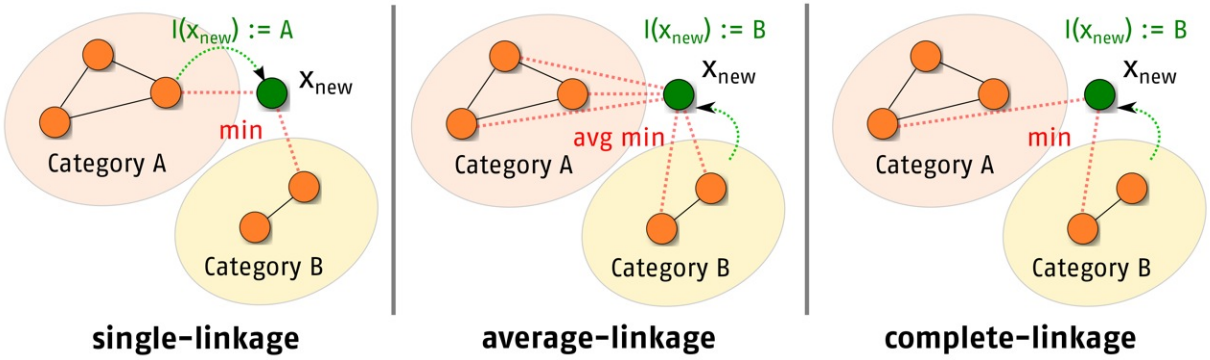


Figure 3.4.: Cluster analysis inspired GNG-based prediction strategies.

3.3.2. Prediction strategies for GNG

In addition to the neuron labeling strategy, we need to define prediction functions that assign labels to unseen examples. These prediction strategies are inspired by linkage strategies typically used in cluster analysis (see [Joh67, And73, Sne73]) and depicted in Figure 3.4.

Single-linkage

In this prediction strategy a new datapoint x_{new} is labelled with that category c of the neuron n that minimizes the distance to this new example:

$$pred_{single}(x_{new}) = \arg\min_{c}(\arg\min_{n \in N(c)} |n - x_{new}|^2)$$

where $N(c) = \{n \in N \mid l(n) = c\}$ is the set of all neurons labelled with category c according to one of the above mentioned neuron labeling function.

Average-linkage

In this strategy, example x_{new} adopts the label of category c to which neurons having the minimal average distance to the example:

$$pred_{average}(x_{new}) = \arg\min_{c}(\sum_{k=1}^{|N(c)|} \frac{|n_k - x_{new}|^2}{|N(c)|})$$

Complete-linkage

In this prediction strategy a new datapoint x_{new} is labelled with category c of the neuron n that minimizes the maximal distance to this new example:

$$pred_{complete}(x_{new}) = \arg\min_{c}(\arg\max_{n \in N(c)} |n - x_{new}|^2)$$

3.3.3. Limitations of batch learning

There are several disadvantages to these strategies. For example, offline labeling strategies assume that there is a definite end of the training phase after which the labeling can be performed. This is of course not the case in life-long learning scenarios in which training and prediction are interleaved. Using offline labeling strategies, we are not able to perform predictions for unseen examples in every iteration step, which is the crucial characteristic of online classification approaches. In the area of cognitive systems engineering, the online nature of learning processes is crucial in order to learn complex behaviors incrementally and continuously [SJ06]. Another disadvantage of offline labeling strategies is the fact that they directly run counter to the online nature of GNG, as training examples are stored explicitly.

3.4. Online Growing Neural Gas (OGNG)

In contrast to offline classification with GNG, as presented in the previous Section 3.3, we present Online Growing Neural Gas (OGNG) as online solution and implementation of the mentioned online architecture from Figure 3.2. OGNG performs a clustering on the input data and additionally assigns category labels to neurons, as shown in Figure 3.5. The resulting classification network is then capable of predicting category labels for new data points on demand.

3.4.1. Online labeling strategies for GNG

In order to extend GNG to an online classification algorithm, we extend the basic GNG by a step in which the label of the presented stimulus is assigned to the winner neuron during the learning process. We denote the winner neuron for datapoint x by $w(x)$. All prediction strategies are local in the sense that they do not consider any neighboring neurons besides the winner neuron $w(x)$. As the labeling is performed on-the-fly, the label assigned to a neuron can change over time, so that the labeling function is dependent on the number of

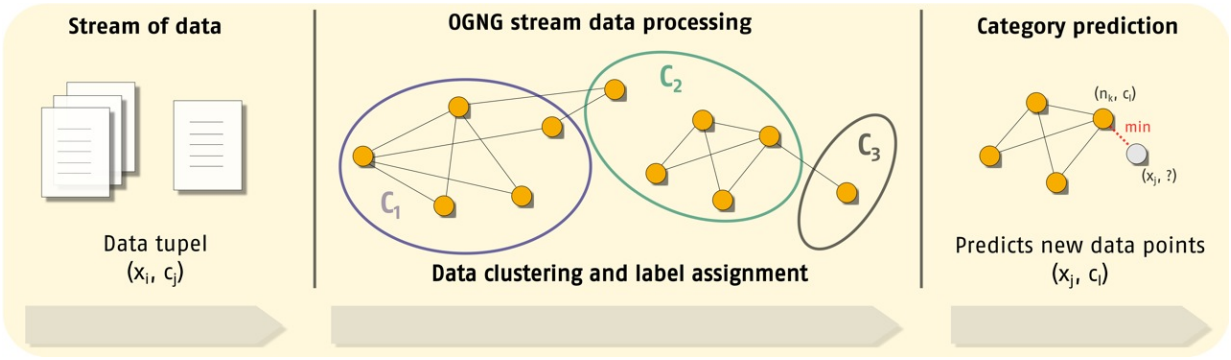


Figure 3.5.: Online Growing Neural Gas (OGNG) processing pipeline.

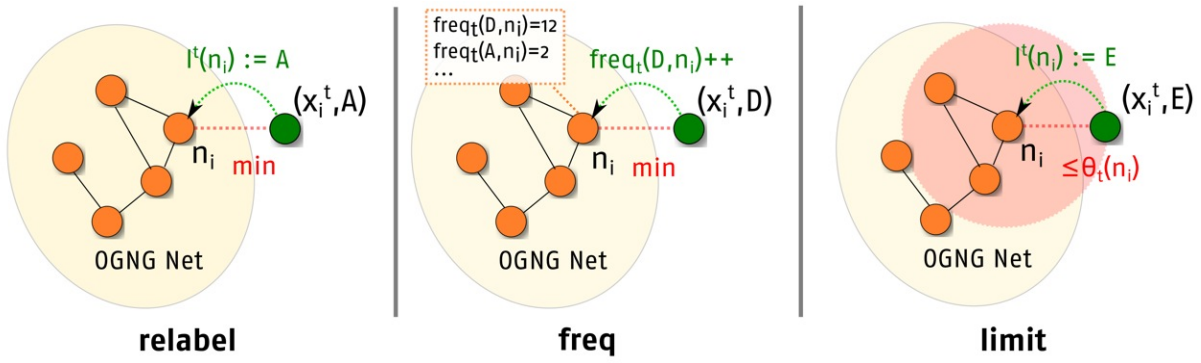


Figure 3.6.: Online GNG-based labeling strategies.

examples the network has seen and has the following form: $l : N \times T \rightarrow C$. We will simply write $l_t(n_i)$ to denote the label assigned to neuron n_i after having seen t data points. The online labeling strategies are depicted in Figure 3.6.

Relabeling method (relabel)

According to this very simple strategy, the winner neuron $w(x)$ adopts the label of x:

$$l^t_{relabel}(n_i) = l_x, \ where \ n_i = w(x)$$

Frequency-based method (freq)

We assume that each neuron stores information about how often a datapoint of a certain category has been assigned to n_i after t examples have been presented to the network as

$freq_t(c, n_i)$. A neuron is labelled by the category which maximizes this frequency, i.e.

$$l^t_{freq}(n_i) = \arg\max_c freq_t(c, n_i)$$

Limited-distance method (limit)

According to this strategy, the winner neuron $n_i = w(x)$ adopts the category label l_x of the datapoint x if the distance between them is lower than the adaptive distance $\theta_t(n_i)$ of the neuron n_i. In case of a smaller distance, $\theta_t(n_i)$ will be updated with the new distance.

$$l^t_{limit}(n_i) = \begin{cases} l_x, & if\ |n_i - x|^2 \leq \theta_t(n_i) \\ l^{t-1}_{limit}(n_i), & else \end{cases}$$

As these labeling strategies do not guarantee that every neuron in the network is actually labelled, we need to extend the prediction strategy to handle unlabelled neurons. For the presented prediction strategies we simply ignore unlabelled neurons during the prediction state.

3.4.2. OGNG Algorithm

In the following we will describe the single steps of the OGNG algorithm. The complete algorithm is shown in Algorithm 3.4.2.

1. In the first step, the algorithm starts with two neurons, randomly placed in the feature space.
2. The first stimulus $x \in R^n$ of the input space (first training example) is presented to the network.
3. The two neurons n_1 and n_2 which minimize the Euclidean distance to x are identified as first and second winner.
4. **A label is assigned to n_1. This labeling depends on the labeling strategies discussed in Section 3.4.1**
5. The age of all edges that connect n_1 to other neurons is increased by 1.
6. In this step, the local error variable $error(n_1)$ of n_1 is updated. This error variable will be used later in order to determine the location for a newly inserted node.

7. In this step, n_1 and its topological neighbors are adapted towards x by fractions e_b and e_n, respectively.
8. A new connection between n_1 and n_2 is created and the age of the edge is set to 0.
9. All edges with an age greater than a_{max} as well as all neurons without any connecting edge are removed.
10. Depending on the iteration and the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.
11. In this step, the error variables of all neurons are decreased by a factor β.
12. The algorithm stops if the stopping criterion is met, *i.e.*, the maximal network size or some other performance measure has been reached.

Algorithm 2 Online Growing Neural Gas (OGNG)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ from the input set or according to input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: Assign the label of x to n_1 according to the present labeling strategy.
5: Increment the age of all edges emanating from n_1.
6: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

7: Move n_1 and all its topological neighbors (*i.e.* all the neurons connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1})$$

$$\Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
8: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
9: Remove edges with their age larger than a_{max}. If this results in neurons having no emanating edges, remove them as well.
10: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r as follows:
Determine unit n_q with the largest error.
Among the neighbors of n_q, find node n_f with the largest error.
Insert a new node n_r halfway between n_q and n_f as follows:

$$\omega_{n_r} = \frac{\omega_{n_q} + \omega_{n_f}}{2}$$

Create edges between n_r and n_q, and n_r and n_f. Remove the edge between n_q and n_f.
Decrease the error variable of n_q and n_f by multiplying them with a constant α. Set the error n_r with the new error variable of n_q.
11: Decrease all error variables of all neurons n_i by a factor β.
12: If the stopping criterion is not met, go back to Step (2).

Complexity in time and space

Before we discuss the complexity of OGNG, it is required to discuss GNG as OGNG inherits its complexity in space in time. Furthermore, it is reasonable to consider alternative data structures for GNG in order to access data elements (neurons) more quickly. According to the GNG algorithm (see Algorithm 1), most operations such as the comparison of the feature vector of a stimulus and a neuron can be applied in linear time, under the assumption of a static feature vector. In case of a dynamically growing feature vector (see Online Human Activity Classifier in Section 7.2) it is difficult to estimate the time complexity which then highly depends on the factor of growth. However, a naive GNG implementation in which neurons are stored in a *linked list* would require the algorithm to consult the complete list of neurons when determining winner, second winner and neighboring neurons and would result in a time complexity of $\mathcal{O}(n)$, with n being the number of neurons. In contrast, a more efficient representation is given by treeGNG which we already discussed in Section 2.1.2. Utilizing the data structure of treeGNG architecture, which is based on a b-tree, would allow GNG to have a time complexity in average and worst case of $\mathcal{O}(log_2 n)$. As far as the complexity in space is concerned, GNG only requires the storage of all neurons and thus provides a space complexity of $\mathcal{O}(n)$.

OGNG adopts this complexity in space and time as its algorithm follows the same principles. OGNG introduces an additional neuron labeling step which for all online labeling strategies (relabel, freq, limit) lies in $\mathcal{O}(1)$, as in all three cases the assignment is based on the state of the neuron itself, in particular, its category label, its frequency (in case of freq) and its θ_t value (in case of limit). For the online prediction strategies the closest neuron to the unlabeled data point is required to be determined and thus leads to a time complexity of $\mathcal{O}(log_2 n)$. As an overall result, OGNG has a time complexity in average and worst case of $\mathcal{O}(log_2 n)$, as well as s space complexity in average and worst case of $\mathcal{O}(n)$.

3.4.3. A uniform two-level architecture

In the following we want to stress an important aspect of OGNG - its two-levels. It should be mentioned that these levels are not explicitly implemented, but OGNG uniformly combines both levels. This subsection should thus be seen as an interpretation of the underlying principles of OGNG and not of a algorithmic architecture. However, as depicted in Figure 3.7, OGNG implicitly consists of a *feature level* and a *category level*. The feature level, and thus the topological structure of the OGNG network, only depends on the distribution of the input data in feature space. The category level instead, is the result of combining the category labels with neurons of the feature level. It is thus the result of our labeling strategies. In a classification task, prediction strategies are applied on the category level, as it holds both category and location information.

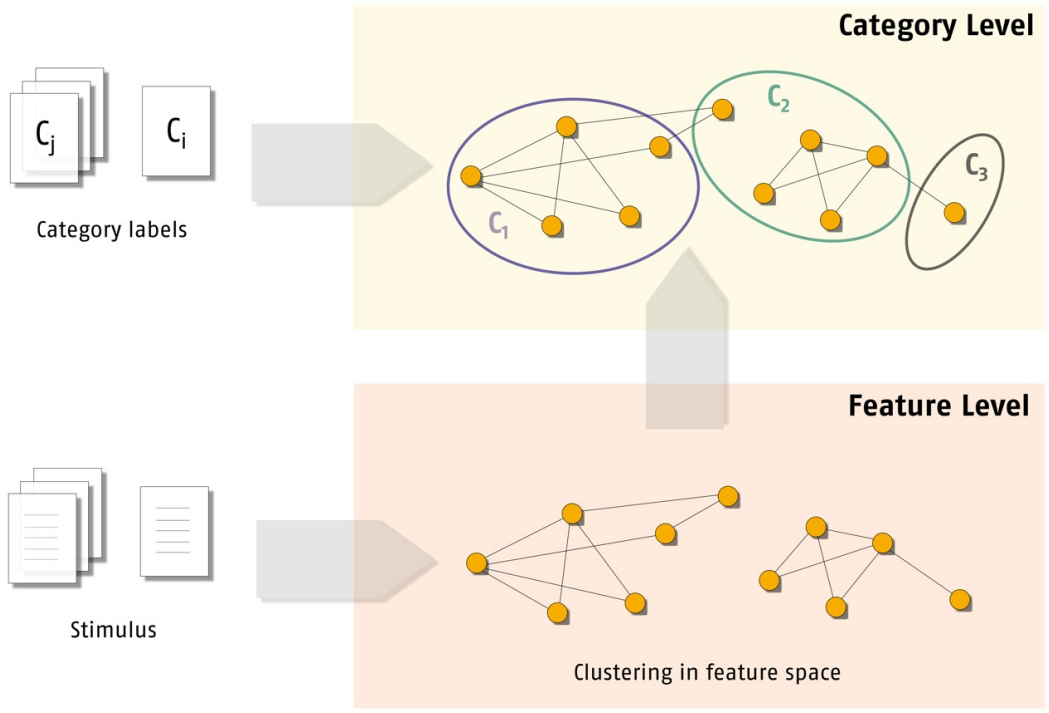


Figure 3.7.: The implicit two-level architecture of OGNG.

This architecture reveals one of the main advantages of OGNG compared to existing GNG extensions that incorporate category label information, *i.e.* SGNG. The independence of the feature level allows OGNG to process labeled and unlabeled data points, in contrast to approaches such as SGNG. In fact, when trained with unlabeled data in a clustering task, OGNG will provide an identical clustering compared to GNG. This is an advantage, as in many applications labels are only sparsely available and sometimes are given in delay. In Section 3.6 we will also compare OGNG to OGNG^{top} a modified version of OGNG in which the category level influences the topological structure of the OGNG network. Thereby, the classification performance of both will be compared to show in how far the performance is effected.

In Chapter 4, we will further investigate the incorporation of unlabeled data points for the classification task and will also evaluate scenarios of late labeling, in which the category label is given after the category clusters already have manifested in the feature level. We will show that the semi-supervised approach quickly adapts to delayed given category labels and that if furthermore benefits from a priori learned structures of such late label categories.

3.5. Datasets

In this section we describe the datasets used for our evaluation with OGNG. There are two artificial datasets (SPIRAL1, ART) and two real datasets (ORL, SEG). While ORL and SEG are standard machine learning benchmark sets, we use the dataset SPIRAL1 (and SPIRAL2) throughout this thesis in order to provide a small controlled environment which we can adapt to the individual challenges we address with the GCM framework. In this chapter we only evaluate OGNG on non-stream dataset to highlight and compare the different labeling and prediction strategies in a small controlled environment. In Chapter 7 we will show the advantage of OGNG and DYNG (see Chapter 5) in a stream data scenario.

SPIRAL1

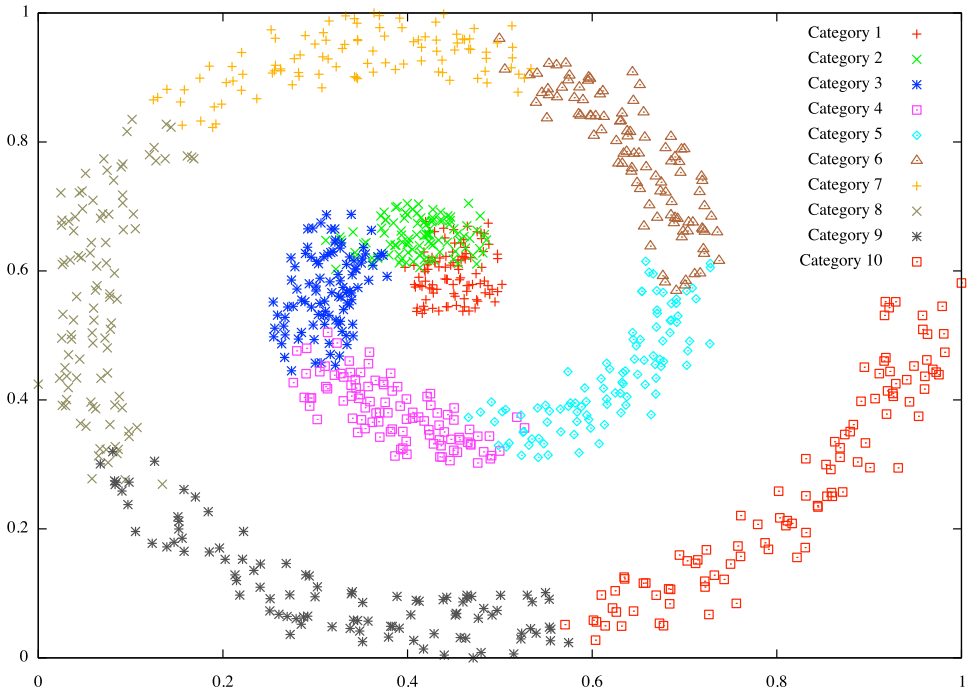


Figure 3.8.: Visualization of the SPIRAL1 dataset.

For this dataset we generated a two dimensional spiral as depicted in Figure 3.8. Spirals are standard benchmark data distributions for clustering tasks and can be found in many evaluations of clustering algorithms. SPIRAL1 is an Archimedean spiral $r(\phi) = a\phi$ with ϕ and $r(\phi)$ being the polar coordinates and $a = 1$ being constant. The spiral was generated with an angle between $[0 : 720]$ degree, uniformly divided into $|C| = 10$ categories (72

degree for each category) and then normalized and translated to provide values between $[0:1]$. For each category $c \in C$ and each angle ϕ we randomly sampled data points within the range $(x-k, y-k)-(x+k, y+k)$ and $k = 0.047$. This also causes the categories to overlap with $k = 0.047$.

ART

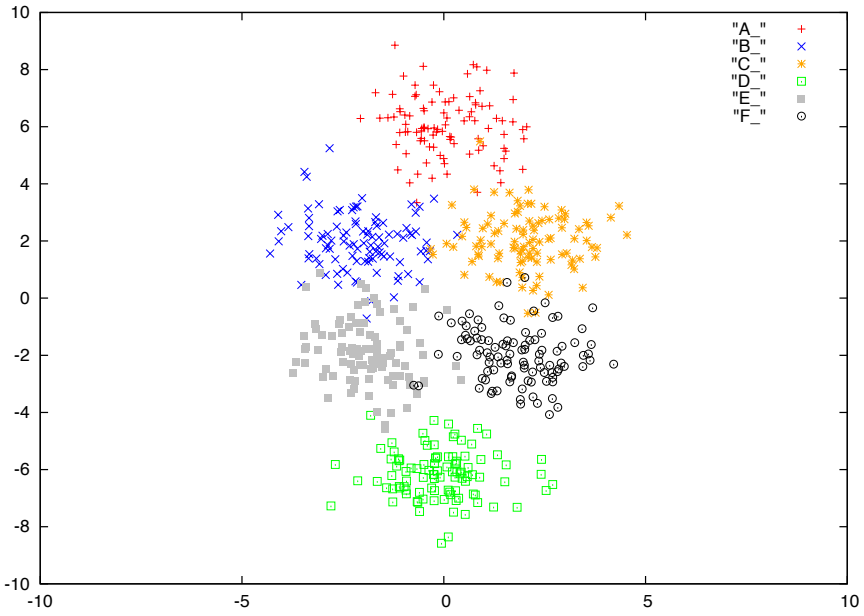


Figure 3.9.: Visualization of the ART dataset.

The second artificial dataset is a two dimensional Gaussian mixture distribution with 6 categories as depicted in Figure 3.9. The centers of each Gaussian distribution are located at $A = [0, 6], B = [-2, 2], C = [2, 2], D = [0, -6], E = [-2, -2], F = [2, -2]$. The data points of each category are Gaussian distributed $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$ with a standard derivation of $\sigma = 1$. The figure shows that each category overlap with its direct topological neighbors as in the SPIRAL1 dataset.

Figure 3.10.: Example images of The ORL Database of Faces.[3]

The ORL Database of Faces (ORL)

The first real dataset is The ORL Database of Faces [Sam94] base which contains 400 frontal images of humans performing different gestures. The dataset consists of 40 categories that consist of 40 distinctive subjects performing 10 facial expressions each, including facial gestures such as opened and closed eyes, and smiling and not smiling. The images were taken at different times with varying lightning and varying facial conditions (glasses/no glasses) conditions. The size of each image is 92×122 with 256 grey scale levels per pixel. In order to reduce the high dimensional feature space with $92 \times 122 = 11224$ dimensions, we downscaled each image from 92×112 to 46×56 and thereby reduced the number of dimensions to 2576. After that we applied a *Principal Component Analysis (PCA)* [Pea01] and again reduced the number of dimensions from 2576 to 60, corresponding to 86.65% of the total variance. Examples of the ORL database are shown in Figure 3.10.

Image Segmentation dataset (SEG)

The last real dataset is the Image Segmentation Dataset [Bla98]. It is a standard machine learning dataset from the UCI Machine Learning Repository, including 2310 instances

[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facesataglance.html

of 7 categories with 19 attributes. The features have been extracted from 7 outdoor images, showing the textures 'brickface', 'sky', 'foliage', 'cement', 'window', 'path' and 'grass'. Each instance corresponds to a 3×3 pixel region with one of the 7 images. For those pixel regions 19 attributes have been derived, including various RGB-based mean values (raw color channel) as well as saturation, hue, intensity means and horizontal and vertical contrast of adjacent pixels. The images were hand-segmented in order to prohibit additional noise.

3.6. Evaluation

In this section we evaluate all introduced labeling methods and prediction strategies to analyze and compare the different possible online and offline GNG-classifiers. Furthermore, we will compare the best offline and online version of the GNG-classifier to a linear multi-class SVM, in order to show how its classification performance scales up against a long time established classifier. We also compare OGNG to an alternative version of OGNG (OGNG^{top}), in which the category labels influence the networks' topology.

3.6.1. Parameters & OGNG^{top}

In order to compare the different labeling strategies to each other, we chose a fixed set of parameters for GNG and used this set in all of our experiments. This set was empirically determined on a trial-and-error basis through preliminary experiments. The GNG parameters are set as follows: insertion parameter $\lambda = 300$; maximum age $a_{max} = 100$; adaptation parameter for winner $e_b = 0.1$; adaptation parameter for neighborhood $e_n = 0.0006$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.0005$ and allowed a maximum number of neurons $n_{max} = 120$. Throughout all of our experiments in this thesis with the different components of GCM, we found the GNG parameters to be very robust and therefore only adjusted λ, a_{max} and n_{max}, while leaving the rest of the parameters fix.

For the SVM we chose a multi-class SVM with a linear kernel. As the SVM is a binary classifier, a multi-class classification problem involves the use of an SVM ensemble in order to set the classes apart of each other. We trained the SVM in a one-vs-all manner, which means that one SVM per class is created and trained with data of that category as positive example and data from any other category as negative example. The multi-class SVM then predicts the class having the highest confidence. It should be mentioned that we relied on the common SVM library libSVM [CL11] and did not implement the SVM ourselves.

In order to experimentally evaluate in how far a connection of the two layers, as described in Section 3.4.3, influences the classification performance of OGNG, we developed

OGNG^{top}. The algorithm is a variation of OGNG in which we included an additional Step 10b (compare to Algorithm 2). In Step 10, as shown in Algorithm 3, we simply connect neurons $n_i, n_j \in N$ which share the same label and disconnect those which are labeled differently.

Algorithm 3 OGNG^{top} (additional Step 10b)

1: **for all** $n_i, n_j \in N$ with $n_i \neq n_j$ **do**
2: **if** $l^t(n_i) = l^t(n_j)$ **then**
3: connect n_i, n_j with an edge (if they are not connected yet).
4: **else**
5: remove the edge that connects n_i and n_j (in case they have been connected before).
6: **end if**
7: **end for**

3.6.2. Experiments & Results

For the experiments with OGNG we evaluated the different labeling and prediction strategies on all of our datasets (SPIRAL1, ART, ORL, SEG). We additionally evaluated the SVM and OGNG^{top} on SPIRAL1.

For SPIRAL1 we performed a 10-fold cross validation and evaluated the accuracy of each algorithm. For ART, ORL, SEG we randomly sampled 10 training/test sets out of our data and averaged the accuracy performance, instead. For these datasets we only included 4 labeled examples in the training set. The reason for this is that the classification problems under consideration are so simple that by using more examples any strategy yields nearly perfect results, thus rendering a comparison meaningless.

Our results are shown in Table 3.1 and Table 3.2. Table 3.1 shows classification accuracy results for various configurations of labeling methods (*min-dist, avg-dist, majority, relabel, freq, limit*) and prediction strategies (*single-linkage, average-linkage, complete-linkage*). For each dataset two tables are shown, representing the offline versions (upper table), and the online version (lower table) of the GNG-based classifier. Thereby, each row of the table represents the classification results of one of the prediction strategies, while each column stands for a different (online/offline) labeling strategy. The cells in each table show the classification accuracy averaged over the 10-folds. Additionally we also provided average results for every labeling strategy (at the end of each column) and every prediction strategy (at the end of each row). The highlighted cells reflect the best averaged result. The second table (Table 3.2) depicts the results of our comparison between OGNG, OGNG^{top} and SVM. For these experiments OGNG and OGNG^{top} make use of the relabel online labeling method and the single-linkage prediction strategy.

	SPIRAL1			
OFFLINE	Min-dist	Avg-dist	Majority	*Average*
Single	92.6	91.9	94.3	***92.9***
Average	87.0	82.1	85.3	*84.8*
Complete	68.4	61.5	65.4	*65.1*
Average	***82.6***	*78.5*	*81.7*	

ONLINE	Relabel	Freq	Limit	*Average*
Single	92.8	92.9	91.6	***92.4***
Average	85.5	85.7	86.6	*85.9*
Complete	68.5	67.7	68.3	*68.2*
Average	***82.3***	*82.1*	*82.2*	

	ART			
OFFLINE	Min-dist	Avg-dist	Majority	*Average*
Single	89.5	89.4	89.5	*89.5*
Average	91.9	91.9	92.9	***92.2***
Complete	90.5	91.2	90.4	*90.7*
Average	*90.6*	*90.8*	***90.9***	

ONLINE	Relabel	Freq	Limit	*Average*
Single	89.5	89.5	89.5	*89.5*
Average	92.9	92.9	92.9	***92.9***
Complete	90.4	90.4	90.4	*90.4*
Average	*90.9*	*90.9*	*90.9*	

	ORL			
OFFLINE	Min-dist	Avg-dist	Majority	*Average*
Single	83.3	82.9	87.0	***84.4***
Average	77.5	77.5	77.9	*77.6*
Complete	62.5	62.0	63.3	*62.6*
Average	*74.4*	*74.1*	***76.0***	

ONLINE	Relabel	Freq	Limit	*Average*
Single	86.7	86.7	87.0	***86.8***
Average	80.9	82.9	82.9	*82.2*
Complete	63.3	62.9	63.3	*63.2*
Average	*77.0*	*77.5*	***77.7***	

	SEG			
OFFLINE	Min-dist	Avg-dist	Majority	*Average*
Single	72.9	64.4	73.6	***70.3***
Average	62.0	59.7	67.6	*63.1*
Complete	56.2	49.5	56.0	*53.9*
Average	*63.7*	*57.9*	***65.8***	

ONLINE	Relabel	Freq	Limit	*Average*
Single	73.6	73.6	73.6	***73.6***
Average	67.6	67.6	67.6	*67.6*
Complete	56.0	56.0	56.0	*56.0*
Average	*65.7*	*65.7*	*65.7*	

Table 3.1.: Classification accuracy results offline vs. online GNG-based classifier.

Results

The results license the following observations:

- **Comparison of offline labeling strategies:** According to Table 3.1, there is no labeling method which significantly outperforms the others. Comparing the accuracy results averaged over all prediction strategies, the *majority method* is the most effective labeling method for the datasets (ART, ORL, SEG) as it provides the highest accuracy with 90.9%, 76.0%, 65.8%, followed by the *min-dist method* with 90.6%, 74.4%, 63.7% and the *avg-dist method* with 90.8%, 74.1%, 57.9%. For the SPIRAL1 dataset *min-dist* achieves slightly but not significantly better results than *majority* with 82.6% compared to an averaged accuracy of 81.7%. Concerning the prediction strategies, the *single-linkage prediction* strategy shows best results averaged over all methods for the datasets (SPIRAL1, ORL, SEG) with 92.9%, 84.4%, 70.3%, followed by the *average-linkage prediction* strategy with an accuracy of 84.8%, 62.6%, 63.1%. The *complete-linkage* yields the worst results with an averaged accuracy of 65.1%, 62.6%, 53.9%. In case of the ART dataset, *average-linkage* yields the best

results with 92.2% compared to 90.7% for *complete-linkage* and 89.5% for *single-linkage*. However, we applied a *one-tailed t-test* and could not prove those results to be significantly different.

- **Comparison of online labeling strategies:** According to Table 3.1, all three online labeling strategies are also almost equal in their classification performance. The *limit method* performs slightly better than the other two methods on the datasets (ART, ORL, SEG) and achieves an accuracy of 90.9%, 77.7%, 65.7%, followed by the *freq method* with an accuracy of 90.9%, 77.5%, 65.7% and the *relabel method* with an accuracy of 90.9%, 77.0%, 65.7%. Again, SPIRAL1 forms an exception as for this dataset *relabel* performs slightly better with an accuracy of 82.3% compared to 82.2% of *limit*. As for the prediction strategies, here it is also the case that the *single-linkage prediction* is the best choice for the datasets (SPIRAL1, ORL, SEG) with an accuracy of 92.4%, 86.8%, 73.6%, followed by the *average-linkage prediction* with an accuracy of 85.9%, 82.2%, 67.6% and the *complete-linkage prediction* with an accuracy of 68.2%, 63.2%, 56.0%. For the ART dataset, *average-linkage* performs best with 92.9%, followed by *complete-linkage* with 90.7% and *single-linkage* with 89.5%.
- **Comparison of online and offline labeling strategies:** Comparing the averaged accuracy of all labeling methods of Table 3.1, the results show that there is no significant difference between them in terms of performance. In fact, we performed t-tests comparing the best online methods (*limit*/*relabel*) with the best offline methods (*majority*/*min-dist*) , using *single-linkage*, and could not prove any significant difference in their achieved performance. We thus consider those results as being comparable. The online labeling methods even provide a slightly higher accuracy on the datasets ART, ORL and SEG.
- **Comparison of OGNG and SVM:** As depicted in Table 3.2, OGNG delivers an averaged accuracy which is comparable to the performance achieved by the SVM classifier. For the datasets (SPIRAL1, ART), OGNG provides a better performance with 92.8%, 89.5% compared to 88.6%, 88.8% of the SVM. For the other datasets (ORL, SEG) the SVM renders itself as best choice with an averaged accuracy of 90.4%, 74.5% compared to OGNG with an accuracy of 86.7%, 73.6%. We applied a t-test on the results of both classifiers (OGNG and SVM) for all folds of all datasets and could prove that OGNG significantly outperforms the SVM on SPIRAL1 while the SVM significantly outperforms OGNG on ORL, with a p-value < 0.05. For the other two datasets (ART, SEG) there is no significant difference in terms of performs at a p-level of $p > 0.05$.
- **Comparison of OGNG and OGNG^{top}:** In Table 3.2 we can see that OGNG clearly outperforms OGNG^{top} on all datasets (SPIRAL1, ART, ORL, SEG) with an accuracy of 92.8%, 89.5%, 86.7%, 73.6% compared to 89.8%, 81.0%, 80.9%, 62.6%.

We also applied t-test for the results of OGNG and OGNG^{top} and could prove that OGNG significantly outperforms OGNG^{top} on the datasets (ART, ORL, SEG). It should be mentioned that we experienced a higher computation time for OGNG^{top} compared to OGNG.

- **Impact of memory:** Strategies relying on some sort of memory (e.g. storing the frequency of seen labels as in the *freq method*), do not perform significantly better than a simple context-free (or memory-less) method (*relabel method*) performing decisions on the basis of new data points only. This shows that the implementation of a label memory does not enhance the classifiers performance.

	OGNG	OGNG^{top}	SVM
SPIRAL1	92.8	89.8	88.6
ART	89.5	81.0	88.8
ORL	86.7	80.9	90.4
SEG	73.6	62.6	74.5
Average	***85.7***	*78.6*	*85.6*

Table 3.2.: Classification accuracy results OGNG, OGNG^{top}, SVM.

3.6.3. Discussion

The results of our experiments show that using online labeling strategies does not significantly deteriorate the performance of a classifier based on GNG in comparison to using offline labeling strategies. An open question is in fact in how far the labels of the neurons actually differ from each other when using online vs. offline labeling strategies. If the labels overlap to a high degree, this would explain why the accuracy of both approaches is comparable. In order to shed light on this issue, we compared the labels assigned to neurons using the online labeling strategies with those assigned by offline labeling strategies at the end of the training phase, quantifying the percentage of neurons for which both methods agree on the label. We carried out this analysis using the single-linkage prediction strategy (averaged over the three datasets ART, ORL, SEG) as it was the best performing strategy in most of our experiments described above. The results are summarized in Table 3.3. We can see that in general there is a very high agreement in the label assigned between the different labeling strategies, *i.e.* the labels are the same for over 85% of the neurons independent of the methods compared. This shows that the online and the offline labeling strategies ultimately assign almost the same labels to the neurons and thus explains the closeness of the results in terms of classification performance.

Percentage of label agreement	Relabel method	Freq method	Limit method
Min-dist method	91.7%	87.6%	89.9%
Avg-dist method	86.4%	85.8%	85.2%
Majority method	98.3%	97.2%	98.9%

Table 3.3.: Percentage of label agreement for different online and offline labeling strategies.

Our experiments show that OGNG is competitive to standard approaches such as the SVM. The results of SPIRAL1 and ART show that OGNG is even capable of providing a better data representation compared to the linear multi-class SVM. For these two datasets the results are plausible, as in both cases categories overlap to some degree and thus cause the categories to become non-convex. In such a scenario, OGNG benefits from its generative character compared to the discriminative architecture of the SVM. While OGNG represents the data and generalizes from a relatively small number of neurons, the support vectors of the SVM model the boundaries of each category and thus try to solve a harder task for these datasets.

Another conclusion we can draw from our experiments is that connecting the *category level* to the *feature level* (see Section 3.4.3) causes the performance of the classifier to fall off in quality. This indicates that a compromising manifold structure which iteratively integrates information coming from the category space and the feature space is not possible in our architecture. Furthermore, it seems that the feature level, which is build upon the manifold of the data in feature space, is more complex in the sense that it can represent both manifolds (feature space and category space), while trying to match it to the data manifold in category space only produces worse results.

The experiments also reveal the fact that in this architecture the use of additional memory to store a label history for every neuron is not necessary. According to the *relabel method*, the labeling method with the smallest label memory, neurons potentially change their labels several times before the model converges. This happens to be as the network and thereby each neuron is in flux during its growth and adaptation, until it's grown enough to represent the processed data on a reasonable level. However, our results lead to the assumption that the network stabilizes very quickly and therefore the neurons are changing their labels less frequently than expected. In order to investigate this further we visualize the development of OGNG in Figure 3.11 while processing the data of SPIRAL1 at a network size of 20, 50, 100 and 120 neurons. The figure shows that OGNG almost uniformly spreads across the data distribution at every stage of the processing. From the images and our experimental results we can conclude that neurons stabilize their location at an very early learning stage and thus do not change their labels often. We will furthermore observe this property of

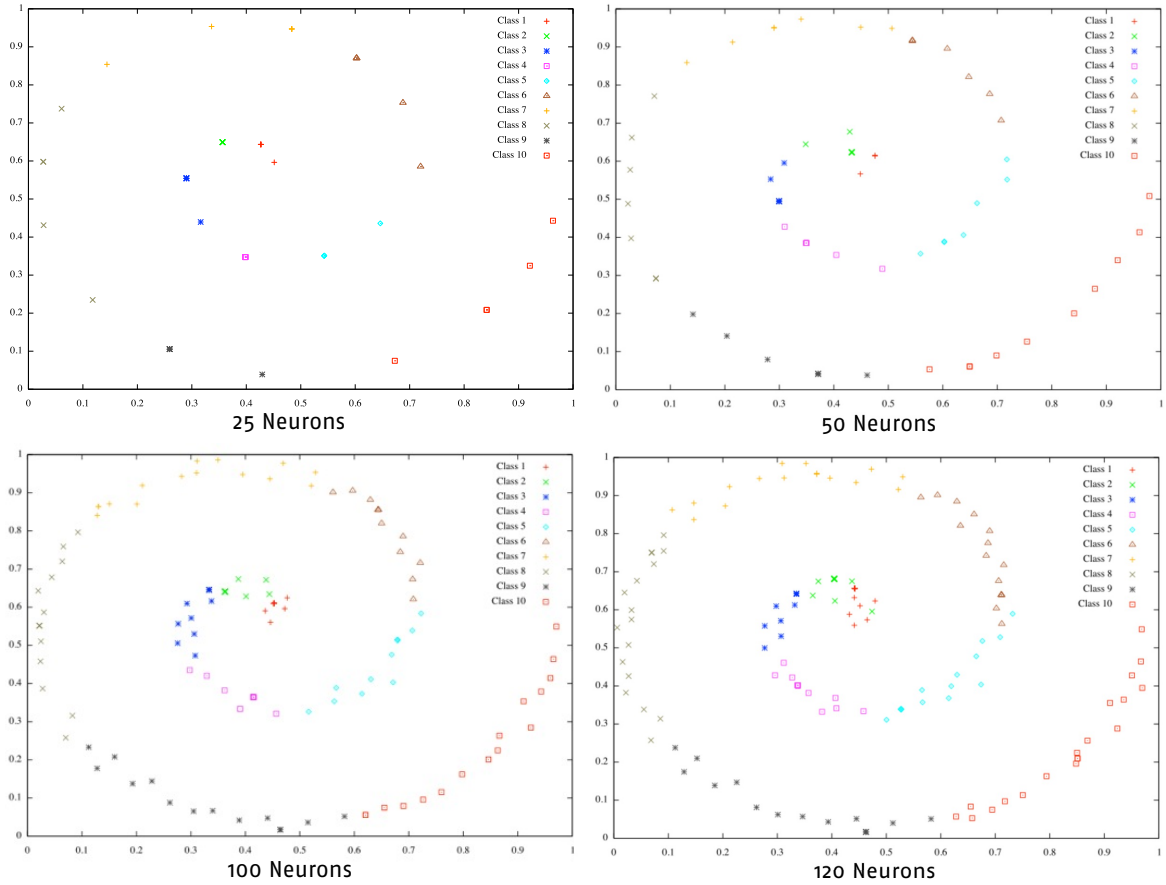


Figure 3.11.: Development of OGNG during the processing of SPIRAL1 with 25, 50, 100, 120 neurons.

a uniform category representation of OGNG in Chapter 7, in which we use OGNG in a human activity setting.

3.7. Summary

In this chapter we have presented, analyzed and compared different online labeling strategies in order to extend the growing neural gas (GNG) algorithm to an online classification approach. While GNG is essentially an unsupervised algorithm, previous approaches have presented extensions of GNG for classification tasks. Such extensions typically rely on a suitable labeling function that assigns labels to neurons as well as a prediction function that assigns labels to unseen examples. In line with this, we have experimentally compared different offline and online labeling strategies inspired by previous research. In this sense, an important question we have addressed in this chapter is whether GNG can

be extended to a classification algorithm without affecting its online nature or degrading performance considerably. Our research has shown that this is indeed possible. Online labeling functions where the label of a neuron can change over time and is computed when a new example is assigned to the neuron in question do not perform worse than offline labeling strategies. We also described the implicit two-level architecture of OGNG, including the feature level, which is responsible for the clustering of the input data, and the category level that builds upon the feature level and thereby exploiting the available label information.

We have furthermore shown that the online classifier OGNG can compete with standard classifiers, *i.e.* a linear multi-class SVM, and that OGNG even achieves a better classification performance for some of our datasets. We have also experimentally shown that the integration of category information into the clustering process of the underlying GNG (OGNG^{top}), and thereby changing the topology of the network, deteriorates the accuracy of the classifier model. A final conclusion we drawn from our experiments with OGNG is that the integration of an additional memory to hold the label history for every neuron is unnecessary, as OGNG converges quickly and almost uniformly to the data distribution.

In the next chapter, we will build upon OGNG and introduce a semi-supervised classifier that is capable of utilizing labeled and unlabeled data in a classification task.

Chapter 4

Learning with weak supervision based on OSSGNG

In this chapter we introduce *Online Semi-supervised Growing Neural Gas (OSSGNG)*, an extension of OGNG which is capable of learning under weak supervision. In standard supervised learning it is assumed that manually assigned category labels are given for all examples of the training dataset during the whole training. In weak supervised learning, instead, training datasets are automatically and heuristically generated and thus can not assumed to be complete and available at any time. OSSGNG, therefore, is capable of utilizing labeled and unlabeled examples for its classification, expending its known categories over the trained model in a semi-supervised learning fashion. Furthermore, OSSGNG can quickly redefine its classification model based on labels that are given at a delayed point in time after some examples of the corresponding class have been already seen. We refer to the process of labels given after their corresponding data points already have been processed as late label learning.

Semi-supervised learning exploits both labelled and unlabeled data and has been successfully applied to many clustering and classification tasks. Existing semi-supervised approaches for GNG process the labelled and unlabeled training data in two separate phases in order to perform a classification. In particular, Semi-supervised Growing Neural Gas (SSGNG) [Zak08] builds upon the Expectation Maximization (EM) principle and thus only trains with the labelled examples at first, before labeling the unlabeled data afterwards and ultimately retrain the classification model. Such approaches can be considered as being offline in the sense that each neuron of the network gets labelled and relabeled (in the second phase) after the GNG training ended and therefore it is necessary to store the complete training data. We present an approach that is able to utilize labelled and unlabeled examples of the training data and processes them on-the-fly, using online labeling and prediction strategies. OSSGNG thus and in contrast to existing semi-supervised

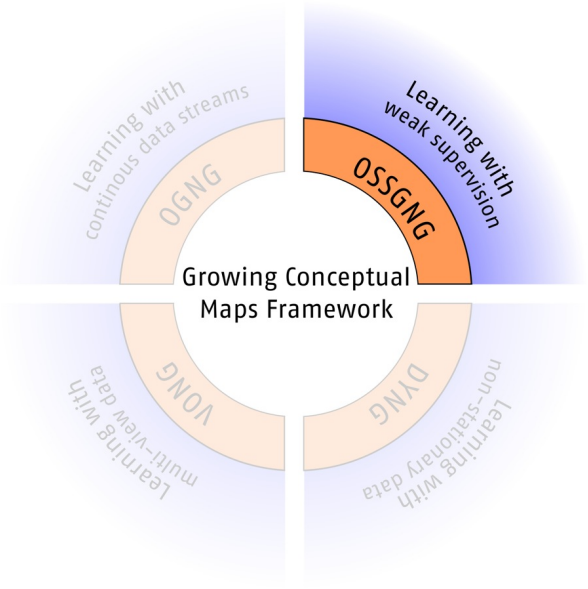


Figure 4.1.: Online Semi-supervised Growing Neural Gas (OSSGNG) as component of the GCM framework.

learning approaches based on GNG circumvents explicit storage of any training examples, processing all data online. As main contribution we show that our online approach does perform as good as previous semi-supervised learning extensions of growing neural gas.

Another closely related approach, *i.e. Semi-supervised Self-Organizing Incremental Neural Network (S-SOINN)* proposed by Shen *et al.* [SF10], also provide a growing structure and are capable of processing labeled and unlabeled data. S-SOINN successively inserts neurons into the network during the learning process, while separating the network into subclusters, depending on the neuron density in the network, and merging those sub clusters into bigger clusters. In contrast, our approach relies on simpler yet effective machinery that does not require any additional heuristics.

In this thesis we have not compared our approach to S-SOINN as its architecture is more complicated than SSGNG while its does not significantly outperform SSGNG.

4.1. Challenges

When using techniques coming from semi-supervised learning in a classification task, the set of training data usually divides into $X_L \cup X_U = X$. Thereby, X_L represents the part of the data for which category labels $c_j \in C$ for each $x_i \in X_L$ are given. Typically, a model $\mathcal{M}$

is then trained with X_L only in the first phase. All labels of X_U are then estimated given the current state of $\mathcal{M}$. In a second phase $\mathcal{M}$ is retrained with X_L and $X_{L'}$, the subset X for which the classifier is most confident. Sometimes, as for SSGNG, this training is performed iteratively in which $|X_U|$ constantly decreases. However, in a life-long learning scenario this training mode is unsuitable due to the fact that X cannot be assumed to be given at any point in time, for which a retraining of the model is not possible. Training in this batch-mode fashion requires the explicit storage of X. When transferring semi-supervised learning to a life-long learning scenario, X and C are only partially given as $X^t \subset X$ and $C^t \subset C$ at a certain point in time t. Therefore, the following challenges emerge out of this setting:

- **Classification with labeled and unlabeled data**: The described offline semi-supervised learning methods require the storage of X^t in order to retrain the classification model, as $\mathcal{M}$ possibly predicts the labels for X_U^t at a later point in time $t+k$ while already processing new labelled data. However, this asynchronous interleaved process would require even more storage. To provide an efficient synchronous processing of labelled and unlabeled data, it is required to predict the label $c_j \in C$ for a data point $x_i^t \in X_U^t$ at iteration t.

- **Generalization from unsupervised categories**: In a life-learning process which utilizes weak supervision, the category labels $c_j^t \in C^t$ for the data $x_i^t \in X^t$ are usually automatically determined by some external algorithm that is based on heuristics rather than human expert knowledge. This means that it cannot be guaranteed that the labels c_j^t are available at iteration t. Furthermore, it is possible that the category labels c_j^t are given at iteration $t+k$ with $k = 1, 2, ...\infty$. In these cases it is desirable to let the model quickly adopt the delayed labels for categories that have been learned in an unsupervised way and for which the corresponding label is provided at a later time point.

4.2. Contributions

In this chapter a semi-supervised extension of OGNG is presented that relies on an online labeling functions to label unlabeled examples and incorporate them into the model on-the-fly. As an important result, we show that OSSGNG outperforms previous semi-supervised extensions of GNG, *i.e.* SSGNG (see Section 4.3.1), which rely on offline labeling strategies. We also show that OSSGNG compares favorably to other state-of-the-art semi-supervised learning approaches on standard benchmarking datasets. Furthermore, it will be shown that OSSGNG is capable of utilizing delayed labels and benefits from previously learned unlabeled categories.

In particular, we offer the following contributions:

- **OSSGNG as extension of OGNG**: We extend the OGNG algorithm by an on-the-fly labeling step in order to assign labels to unlabeled data on-the-fly while processing them.
- **Comparison of OSSGNG and SSGNG**: We compare OSSGNG with SSGNG [Zak08] as baseline on a classification task and show that the online character of OSSGNG does not deteriorate the classification performance compared to SSGNG, but even outperforms SSGNG in 75% of our experiments.
- **Comparison of OSSGNG and OGNG**: We compare OSSGNG to OGNG on several datasets to show the effect of the novel labeling strategy for unlabeled data points.
- **Comparison of OSSGNG and Standard SSL approaches**: We show that OSSGNG is competitive with respect to other established semi-supervised classification approaches and demonstrate its performance on SSL benchmark datasets.
- **Late labeling with OSSGNG**: We also investigate the capability of OSSGNG to deal with late labeling and therefore let OSSGNG learn categories unsupervised at first and then tracking its classification performance for those categories while assigning their labels at a later point in time.

Parts of this chapter have already been published in the proceedings of the *Workshop on New Challenges in Neural Computation (NC2)* [BC11b] in 2011 and in the *International Journal of Neural Systems* [BC12] in 2012.

4.3. Offline semi-supervised learning with GNG

In this section Semi-supervised Growing Neural Gas (SSGNG) will be described in detail as introduced by Zaki *et al.* [Zak08]. SSGNG is a semi-supervised GNG-based classifier that utilizes labeled and unlabeled data in a classification task. It operates offline in the sense that it iteratively assigns labels to unlabeled data through multiple passes. However, SSGNG renders itself ideal for a comparison with our approach as that it implicitly follows the two-level architecture as described in Section 3.4.3 of Chapter 3 and thus performs the GNG-based clustering independently of the label information. SSGNG will serve us as baseline in order to compare our approach against and to underline the advantages of OSSGNG.

4.3.1. Semi-supervised Growing Neural Gas (SSGNG)

In the approach of Zaki *et al.*, two steps are iterated until the labeling of the network stabilizes. Both steps are similar to those of the *Expectation-Maximization (EM)* approach [Har58]. In a first step, the network is trained using labeled examples only. Then, labels are assigned to neurons using an offline labeling approach. This step can be seen as *maximization-step (M-step)*, in which the network maximizes the likelihood $p(X_L|C_L)$ with C_L being the category labels of X_L. In the *expectation-step (E-step)*, unlabeled examples are classified into the network and labeled appropriately. Following the principle of the EM algorithm, these two steps are then iterated until the labeling converges.

4.3.2. SSGNG Algorithm

In the following, we will describe each step of the SSGNG algorithm in detail:

1. The algorithm starts with a given set of labeled and unlabeled training examples $X_L, X_U \subseteq X$, as well as an initially empty set of newly labeled data $X_{L'}$. The set $X_{L'}$ holds all examples from X_U that have been labeled during the last iteration step of the training process.

2. In the next step, the GNG network trains only with X_L. In this step, the clustering is only performed on the basis of the feature vectors of X_L, without taking the label information into account.

3. Labels are assigned to each neuron of the network after the GNG training has finished. As Zaki *et al.* do not explicitly describe their labeling strategy.

4. In this step, a data point $x_j \in X_U$ is presented to the SSGNG network and a winner neuron n is determined by the minimal distance to x_j.

5. The data point x_j adopts the label of n, gets removed from X_U and is added to the dataset of newly labeled data $X_{L'}$.

6. Depending on the amount of unlabeled data points left in X_U, a next iteration step is performed. The algorithm then goes back to Step 4 in order to present the next unlabeled data point. Otherwise, Step 7 is applied.

7. In this step, the SSGNG network retrains with $X_L + X_{L'}$.

8. The algorithm stops if the labels of X_U stabilize during the iterations in the sense that the labels do not change anymore.

It should be mentioned that in our implementation the SSGNG quickly stabilized in all of our experiments, which is understandable after a closer inspection of the algorithm. As initially mentioned, SSGNG (and OGNG) does not utilize label information to guide the clustering of the underlying GNG. This means that after labeling all unlabeled data (steps 4-6) and retraining SSGNG in Step 7, the clustering will remain stable and thus the labels will also stabilize after the next iterations. We see a design problem here as neurons are labeled after the GNG-based clustering. This means that it is not necessary to retrain the SSGNG after the second iteration in which it is trained with the complete available data X_L and $X_{L'}$, with $|X_{L'}| = |X_U|$.

Algorithm 4 Semi-supervised Growing Neural Gas (SSGNG)

1: Given X_L and X_U, let $X_{L'} = \{\emptyset\}$ represent an initial empty set of newly labeled data.
2: Present X_L to the GNG algorithm and train the network only with X_L.
3: Label all the neurons of the GNG network according to X_L.
4: Present an input x_j from X_U iteratively and compute the Euclidean distance between x_j and every neuron n of the GNG network:

$$Distance = ||\omega_n - x_j||^2$$

5: Label x_j according to the class label of the winner node. Remove x_j from the current unlabeled dataset, X_U, and add x_j into the newly labeled dataset $X_{L'}$.
6: If all unlabeled data has been labeled, go to 7, otherwise go back to 4.
7: Present X_L and $X_{L'}$ together to the GNG classifier and retrain the classifier with $X_L + X_{L'}$ and evaluate the new classification performance.
8: Check the labels of $X_{L'}$; if they become stable during successive iterations, stop. Otherwise go back to Step 4.

4.3.3. Limitations of SSGNG

The main disadvantage of the SSGNG is the fact that labels are assigned to each neuron a posteriori after the end of the training phase. Thus, the approach is not able to process a continuous stream of labeled and unlabeled training examples. Furthermore, labeled and unlabeled examples are processed in different phases and therefore need to be stored until the SSGNG training ends. Another disadvantage of SSGNG is that a minimal set of labeled examples for each class is crucial for the training. This even excludes SSGNG from being applicable to late labeling scenarios, as SSGNG requires all categories to be known at its initialization. Our online version of Semi-supervised Growing Neural Gas, presented in the next section, circumvents these problems.

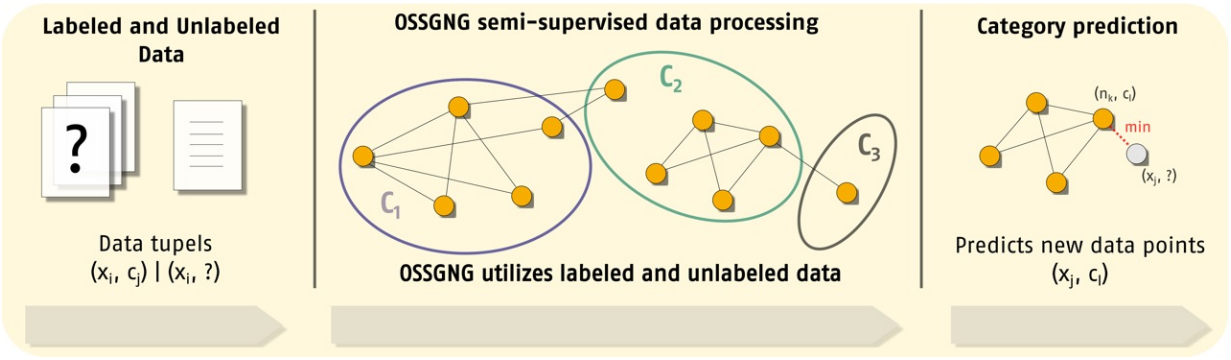


Figure 4.2.: Online Semi-supervised Growing Neural Gas (OSSGNG) processing pipeline.

4.4. Online Semi-supervised Growing Neural Gas (OSSGNG)

In this section we introduce OSSGNG as semi-supervised extension of OGNG. At first, we will explain the new additions that set OSSGNG and OGNG apart, before we then focus on the details of the algorithm. We will further discuss the dynamics of the OSSGNG and how it relates to the two-level architecture as described in Section 3.4.3. This also includes the concept of late label learning and how OSSGNG can adapt to it.

OSSGNG is a semi-supervised classifier that, in contrast to SSGNG, utilizes labeled and unlabeled training examples uniformly for its classification model in every iteration step. This means that OSSGNG is able to solve a semi-supervised classification task after each iteration step. As shown in Figure 4.2 from left to right, labeled and unlabeled data points are the input for our OSSGNG algorithm and its processing results in a OSSGNG classification model which then is capable of predicting previously unseen data points. The main advantage of OSSGNG lies in its ability to train in an online fashion without the need of storing training examples explicitly.

4.4.1. OSSGNG as extension of OGNG

OGNG is an incremental online classifier that is capable of classifying a continuous stream of data without the need to explicitly store any training data. Its implicit two-level architecture allows OGNG to process labeled and unlabeled data, as the data is clustered depending on its similarity in feature space, without any guidance of category labels. However, OGNG only makes use of unlabeled data to redefine the feature level, without integrating this information into the category level (see Section 3.4.3). Therefore, the classification model based on OGNG does not benefit from unlabeled data in a classification

task. In contrast, OSSGNG predicts the labels of unlabeled examples in order to optimize the model and thus change its classification behavior.

The differences of OSSGNG and OGNG are minor but effective, as we can show in our evaluation in Section 4.6 when comparing both OSSGNG and OGNG. OSSGNG introduces an additional prediction step (Step 4 of Algorithm 5) in which labels for unlabeled data are predicted. After applying the step it behaves equally to OGNG. As labeling strategy in Step 5 (see Algorithm 5) the relabel method (see Section 3.4.1) for the experiments described in this chapter, which has proven to be simple and effective providing the smallest label memory. As prediction strategy (also in Step 4) we chose single-linkage (see Section 3.3.1) as it performed best in previous OGNG experiments. In general it is possible to implement OSSGNG with any of the OGNG related online labeling and prediction strategies.

4.4.2. OSSGNG Algorithm

In the following we will describe the single steps of the OGNG algorithm. The complete algorithm is shown in Algorithm 5. We highlighted the novel step which sets OSSGNG and OGNG apart.

1. The classification model is initialized with two connected neurons that are randomly placed in feature space.
2. The first stimulus $x \in R^n$ of the input space (first training example) is presented to the network.
3. The two closest neurons n_1 and n_2 are determined as winner and second winner.
4. **A label for x is predicted in case there is no label available for this specific data point.**
5. According to the used labeling strategy, a label is assigned to n_1.
6. The age of all edges that connect n_1 to other neurons is increased by 1.
7. The local error variable $error(n_1)$ of n_1 is updated. This error variable will be used later in order to determine the location for a newly inserted node.
8. In this step, n_1 and its topological neighbors are adapted towards x by fractions e_b and e_n, respectively.
9. The neurons n_1 and n_2 get connected and the age of its edge is set to 0.
10. All edges with an age greater than a_{max} as well as all neurons without any connecting edge are removed.

11. If the iteration counter is a multiple of the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.
12. The error variables of all neurons are decreased by a factor β.
13. The algorithm stops if the stopping criterion is met, *i.e.*, the maximal network size or some other performance measure has been reached.

Algorithm 5 Online Semi-supervised Growing Neural Gas (OSSGNG)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ from the input set or according to input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: If the label of x is missing, assign a label to x according to the selected prediction strategy.
5: Assign the label of x to n_1 according to the present labeling strategy.
6: Increment the age of all edges emanating from n_1.
7: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

8: Move n_1 and all its topological neighbors (*i.e.* all the neurons connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1})$$

$$\Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
9: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
10: Remove edges with their age larger than a_{max}. If this results in neurons having no emanating edges, remove them as well.
11: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r as follows:
Determine unit n_q with the largest error.
Among the neighbors of n_q, find node n_f with the largest error.
Insert a new node n_r halfway between n_q and n_f as follows:

$$\omega_{n_r} = \frac{\omega_{n_q} + \omega_{n_f}}{2}$$

Create edges between n_r and n_q, and n_r and n_f. Remove the edge between n_q and n_f.
Decrease the error variable of n_q and n_f by multiplying them with a constant α. Set the error n_r with the new error variable of n_q.
12: Decrease all error variables of all neurons n_i by a factor β.
13: If the stopping criterion is not met, go back to Step (2).

Complexity in time and space

OSSGNG inherits OGNGs' complexity in time and space as it is based on the same architecture. As an addition, OSSGNG utilizes unlabeled data points for its model updates

and thus, depending on the average number of unlabeled data points, increases the computational complexity slightly. However, due to the time complexity of the with OGNG introduced online prediction strategies lies in average and worst case in $\mathcal{O}(log_2 n)$, OSSGNG yields for an overall time complexity in average and worst case of $\mathcal{O}(log_2 n)$ and a space complexity in average and worst case of $\mathcal{O}(n)$.

4.4.3. Dynamics of the OSSGNG labeling

OSSGNG builds upon OGNG and thus shares the same implicit two-level architecture, including a feature level which represents the data in feature space and a category level which bridges the feature level and the category label space that is defined when processing labeled data. However, in contrast to OGNG, OSSGNG labels unlabeled data and thus extends the amount of data that effects the category level, as the data point and its (assigned) label is further processed. The quality of the (new) data therefore relies on the quality of the prediction of the classification model at prediction time. This is a problem not particular limited to OSSGNG, but of semi-supervised learning in general which require certain data assumptions (see Section 2.3) to be fulfilled in order to be useful. Also drifting concepts are a huge problem for semi-supervised approaches. In Chapter 5 we introduce an extension of GNG that particularly addresses this challenge. However, in order to deeply understand the dynamics of OSSGNG, we will have a closer look at what we call labeling dynamics of the OSSGNG labeling.

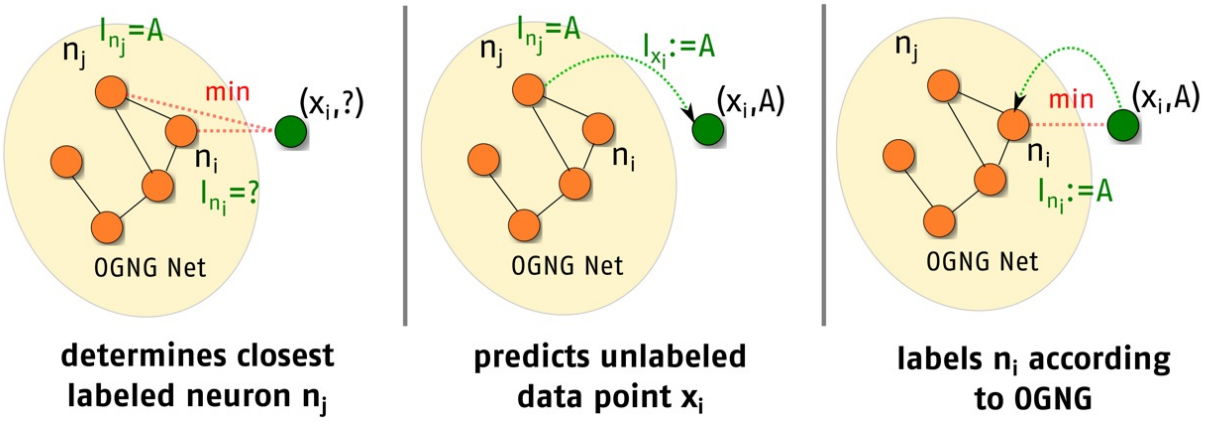


Figure 4.3.: Illustration of the labeling dynamics of OSSGNG.

Figure 4.3 illustrates these *labeling dynamics* of OSSGNG. The figure shows three steps of the labeling process, which characterizes OSSGNG. In the first image (left), a new unlabeled data point x_j is presented to the OSSGNG network. In this image the nearest

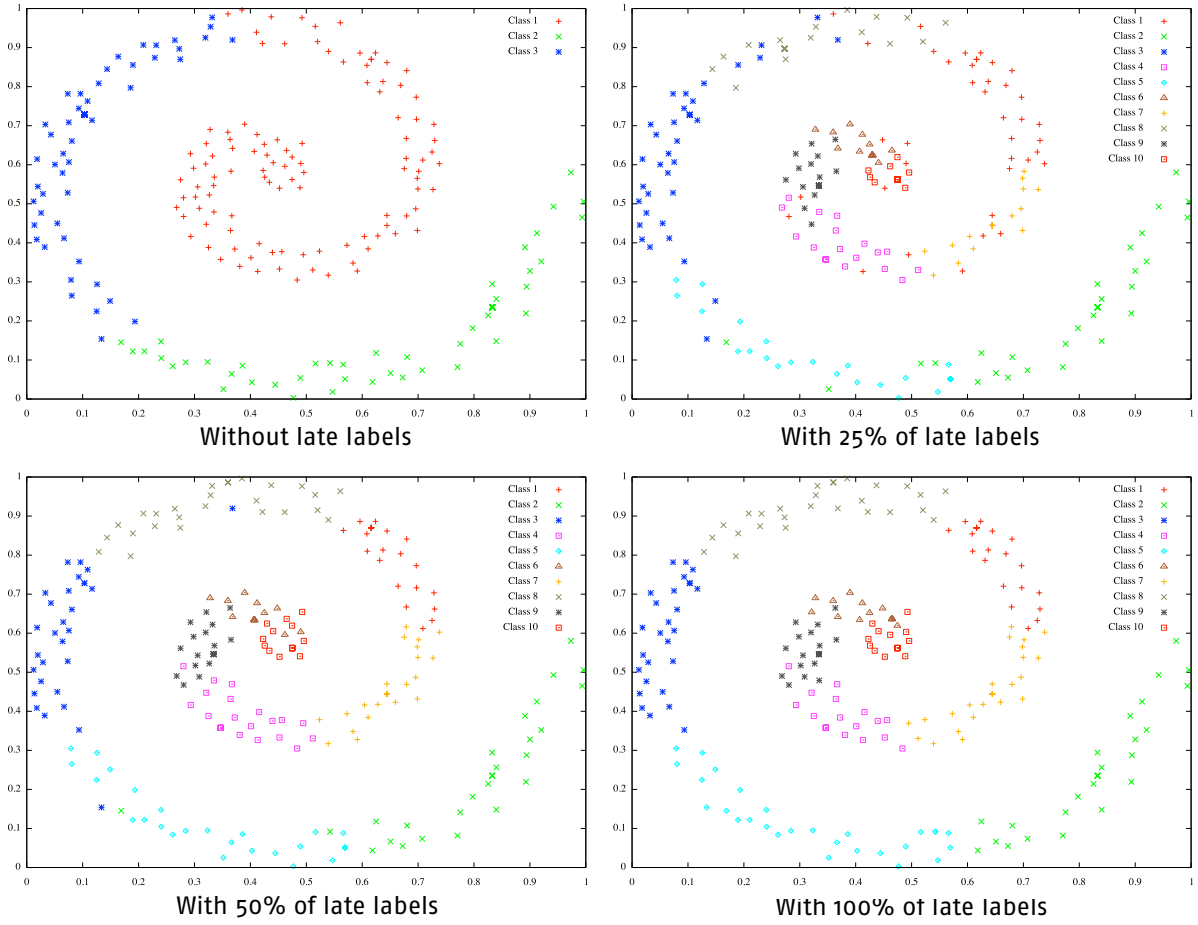


Figure 4.4.: 2D Visualization of OSSGNG during the processing of SPIRAL1 with 0%, 25%, 50%, 100% late labels.

neighbor n_1 does not provide a label, while the second nearest neighbor n_2 is labeled with category A. According to the new prediction strategy of OSSGNG x_j adopts the label of the closest neighbor in the network that provides a label. Therefore, x_j is labeled with category A according to n_2, as depicted in the second image (middle). However, the most interesting part of the labeling dynamics are shown in the third image (right). After a label prediction for an unlabeled data point, OSSGNG follows the behavior of OGNG and thus labels the winner neuron n_1 according to the label of the presented stimulus x_j, which is in our case the category label A. This demonstrates how OSSGNG propagates categories in the network and furthermore how it generalizes and expands the known categories.

OSSGNG is mainly designed to process labeled and unlabeled data in a semi-supervised classification scenario, but is also capable of adapting to scenarios in which a weak supervision is given in form of late label learning. As described earlier in this chapter, late

label learning involves category labels c_j^t that correspond to the data points $x_i^t \in X^t$ to be given at $t + k$ with $k > 0$. While OSSGNG expands and propagates its known categories over the network, due to its labeling dynamics of labeling it still quickly adapts to newly given labels according to the OGNG labeling strategy. In order to demonstrate this behavior, we visually and experimentally investigate the adaptation of OSSGNG towards late labeling scenarios. In Figure 4.4 the OSSGNG network is visualized in four images after being trained with the SPIRAL1 dataset (see Section 3.6) and various label configurations. For the visualization in the first image (top-left), OSSGNG was trained with all data points of SPIRAL1, while only labels for category 6, category 8 and category 10 have been provided. For the other images we simply provided the missing labels for all categories and let the category level of OSSGNG adapt to those labels, by providing 25% of the late labels (top-right), 50% of the late labels (bottom-left), and 100% of the late labels (bottom-right). After having seen 25% of the late labels, OSSGNG already has acquired all categories with a number of neurons. Although the category boundaries are blurred to some degree, especially at the category boundary on top of the spiral, the overall distribution of the 10 categories is already clearly visible. After having seen 50% of the late labels, the category boundaries got more precise while only a few wrongly labeled neurons are left inside the category distributions. After seen all of the late labels, all categories are well established in the OSSGNG network and no obviously wrongly labeled neurons are left. The visualization shows that OSSGNG quickly adapts the category topology (its category level) and thus is capable of adapting to a late label learning scenario. In Section 4.6, we will experimentally show that the prior knowledge about category structures can be beneficial in a late labeling classification task.

4.5. Datasets

We evaluate OSSGNG on overall six datasets, including three artificial sets (g241c, g241d, Digit1), and three real sets (USPS, COIL, BCI). These datasets have been proposed by Chapelle *et al.* [Cha06] as benchmarking for semi-supervised classification and are widely used in this field. The artificial datasets were specifically generated in order to fulfill the data assumptions we described in Section 2.3. We use SSGNG as baseline for our approach and evaluate the classification accuracy on test for the six datasets described below in more detail. Except for the BCI dataset, all SSL benchmarking datasets include 1500 data points of 241 dimensional feature vectors. In order to visualize the data, Figure 4.6 shows a two-dimensional representation of each dataset plotting the first two principal components after applying a Principal Component Analysis (PCA) [Pea01].

We describe these datasets in what follows:

g241c & g241d

The first artificial dataset (g241c) is a Gaussian mixture distribution that was generated by two unit-variance isotropic Gaussians with their centers having a distance of 2.5 from each other in a random direction. Additionally, all dimensions are standardized in the sense that they are shifted and rescaled to zero-mean and unit variance. The dataset is designed to fulfill the cluster assumption, but not the manifold assumption. This means that data points of the same clusters are likely to belong to the same category. In Figure 4.6 (top-left), a two-dimensional representation of both categories is shown according to the two most varying dimensions. Both Gaussian distributions are clearly visible in the image.

The second artificial dataset (g241d) is a similar Gaussian mixture distribution such as g241c. However, the two classes A, B were split into A_1,A_2 and B_1,B_2, as depicted in Figure 4.6 (top-right). The distance between the subclasses (A_1, B_1) and (A_2, B_2) was set to 2.5 in random direction, while the interclass distance between (A_1, A_2) and (B_1, B_2) is 6. The dataset was designed in such a way that these subclasses are not convex, thus not fulfilling the cluster assumption anymore.

Digit1

In the last artificial dataset, images of the digit '1' were generated. These 16×16 images are the result of transformations of the digit along five degrees of freedom: two for translations ([-0.13,0.13] each), one for rotation ([$-90°$,$90°$]), one for line thickness ([0.02,0.05]), and one for the small line at the bottom ([0,0.1]). The class labels were set according to the tilt angle, with the boundary corresponding to an upright digit. Additional noise was added in order to make the task a slightly more difficult. In this case, the dataset is generated to fulfill the manifold assumption, but does not provide a cluster structure. Its two-dimensional projection is shown in Figure 4.6 (mid-left).

USPS

This first real dataset is derived from the well known USPS handwriting dataset, which includes 16×16 images of 10 handwritten digits. The digits '2' and '5' were assigned to the first category, while the rest of the ten digits ('1', '3', '4', '6', '7', '8', '9', '10') were assigned to the second category. Thus, the dataset is imbalanced with the ratio of 1:4. The original dataset is also modified by additional introduced noise and masked dimensions in order to set this dataset apart. A projection of the dataset onto the two most variant dimensions is shown in Figure 4.6 (mid-right).

COIL

The second real dataset is derived from the Columbia object image library (COIL-100) [NNM96] and includes color images of 100 different objects taken from varying angles (in steps of 5 degrees) at a resolution of 128×128. The dataset was again modified in several ways to provide a unique semi-supervised learning benchmark set. Every image was downsampled to 16×16 images by averaging over 8×8 pixel blocks. Thereby, only the information of the red RGB channel was considered. The authors of the dataset randomly selected 24 out of the 100 objects and partitioned them into six categories with four objects each. Finally, they applied an algorithm to introduce noise and masking dimensions. The dataset is visualized in Figure 4.6 (bottom-left).

BCI

In the last real dataset, EEG (electroencephalography) measurements were recorded from 39 electrodes. As published by by Lal *et al.* [LSH+04], the experiments consist of recordings from 400 trials with subjects imagining a left hand movement (category -1) and a right hand movement (category +1). An autoregression model of the order 3 was applied to the resulting 39 time series in order to construct a 117 (39×3)-dimensional feature vector. A two-dimensional projection of BCI is depicted in Figure 4.6 (bottom-right).

4.6. Evaluation

In this section we evaluate OSSGNG as semi-supervised extension of OGNG of Chapter 3 in our experiments. We experimentally compare its classification performance to an existing semi-supervised learning approach, *i.e.* Semi-supervised Growing Neural Gas (SSGNG)[4], on standard semi-supervised classification benchmark datasets. We will show that OSSGNG outperforms SSGNG in most of the datasets. Furthermore, OSSGNG will be compared to standard semi-supervised classifiers described in the literature. Thereby, we also compare our results to a *1-Nearest Neighbor (1-NN)*, OGNG and linear multi-class Support Vector Machine (SVM) classifier, which are trained with labeled examples only. This allows us to quantify the benefit of semi-supervised learning for those datasets. We also investigate if OSSGNG benefits from prior clustered unlabeled data in a late labeling scenario.

[4]We use the min-dist method from Section 3.3.1 as labeling strategy for SSGNG.

4.6.1. Parameters

As we want both GNG variations, SSGNG and OSSGNG, to be comparable, we chose a fixed set of parameters. In contrast to the set we use in the rest of our experiments, we chose to use a set throughout our experiments that was proposed by Zaki *et al.*[Zak08] for SSGNG to provide a fair comparison here. However, we found out that the parameter set of OGNG and its extensions is insensitive towards parameter changes for most of the parameters. In our experiments in this chapter the parameters are thus set as follows: insertion parameter $\lambda = 300$; maximum age $a_{max} = 100$; adaptation parameter for winner $e_b = 0.2$; adaptation parameter for neighborhood $e_n = 0.006$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.995$. The algorithm stops when a network size of 200 neurons is reached. Our experiments are carried out using a 12-fold cross validation with 100 labeled examples per fold, respectively. This setup corresponds to the setup used in Chapelle *et al.* [CO05] where a number of state-of-the-art SSL algorithms were benchmarked.

4.6.2. Experiments & Results

We evaluate the accuracy of all compared algorithms on test, as shown in Table 4.1 and Table 4.2. Each row in the table represents the accuracy on test averaged over the 12 folds. The best accuracy is highlighted in each row. We also compare to a 1-NN classifier and a linear multi-class SVM in order to provide an overall baseline for all SSL approaches. Both classifiers were only trained with the labeled data points of our data sets.

For the late labeling scenario, we compare OSSGNG to a modified version OSSGNG-M, in oder to demonstrate the benefit of a priori clustering of categories which are labeled in delay. For these experiments we let OSSGNG process the complete COIL dataset while only presenting the labels for Category 6, Category 8 and category 10. We then stepwise provide the labels for the missing categories and measure the classification performance of OSSGNG for those late labeled categories. OSSGNG-M in contrast to OSSGNG also adapts to the late labels without processing the unlabeled data a priori, still providing the same network size. The accuracy curve of OSSGNG and OSSGNG-M in these experiments is depicted in Figure 4.5.

Results

The results license the following observations:

- **Comparison of OSSGNG and SSGNG:** According to Table 4.1, OSSGNG clearly outperforms SSGNG on five out of six datasets (g241c, g241d, COIL) with 52.98%, 62.66%, 81.45% for OSSGNG and 49.64%, 44.03%, 75.49% for SSGNG. Both have a comparable performance on the datasets (Digit1, USPS) with 96.77%, 93.07% for OSSGNG and 96.20%, 92.58% for SSGNG. For the BCI dataset SSGNG performs best with 80.51% compared to 79.47% for OSSGNG. On average, OSSGNG has also a higher accuracy with 77.73% compared to SSGNG with 73.08%. According to these results, it is valid to claim that OSSGNG outperforms SSGNG, while circumventing the need to explicitly store training examples and perform several passes over the data until reaching convergence.

- **Comparison of OSSGNG and OGNG:** The results in Table 4.1 show that extending OGNG with a semi-supervised component (OSSGNG) improves its classification performance for the datasets (g241c, g241d, COIL, BCI) by 6.55%, 7.68%, 4.34%, 2.14%. There are only two datasets (Digit1, USPS) for which OSSGNG yields worse results compared to OGNG and its performance decreases by 1.04%, 1.28%, albeit these differences are clearly minor. Interestingly, these are also the datasets for which a semi-supervised SVM classifier (TSVM) performs worse with 93.49%, 90.23% than a standard SVM with 94.47%, 90.25% (see Table 4.2). This shows that SSL is not effective for these particular datasets.

- **Comparison of OSSGNG and standard semi-supervised learning algorithms:** We additionally compared our results to the results of standard semi-supervised classification algorithms published by Chapelle *et al.* [CO05], namely *Transductive SVM (TSVM)* [Joa] (using a linear kernel), *Cluster-Kernel* [Nob03], *Data-dependancy regularization* [CA06] and *Low-Density Separation (LDS)* [CO05]. We did not reimplement these algorithms, but compared our results to the published results using the same data under same conditions. The results are summarized in Table 4.2. On two datasets (g241c, g241d), OSSGNG performs definitely worse compared to the other semi-supervised learning approaches with 52.98%, 62.66%. On the other four datasets (Digit1, USPS, COIL, BCI), the performance of OSSGNG is better than the ones of 1NN, SVM, TSVM and Cluster-Kernel with 96.77%, 93.07%, 81.45%, 79.47%, while the Cluster-Kernel algorithm performs best on average. Therefore, the results are comparable to those of other state-of-the-art semi-supervised learning approaches. On one dataset, *i.e.* BCI, it is even the case that OSSGNG outperforms all other approaches by far 79.47%. These results clearly license the conclusion that OSSGNG can compete with other semi-supervised learning approaches.

- **OSSGNG vs. OSSGNG-M**: According to Figure 4.5, OSSGNG clearly benefits from its a priori clustering of unlabeled data and, furthermore, quickly adapts to the new labels. OSSGNG achieves an accuracy of 88% after seen all labeled and

late labeled examples compared to 68.27% of OSSGNG-M. It is noticeable that the accuracy curve of OSSGNG is smoother than those of OSSGNG-M, which is most probable an indication for a more uniformly representation of the underlying categories by OSSGNG.

	OGNG (labeled data)	SSGNG	OSSGNG
g241c	46.43	49.64	**52.98**
g241d	54.98	44.03	**62.66**
Digit1	**97.81**	96.20	96.77
USPS	**94.35**	92.58	93.07
COIL	77.11	75.49	**81.45**
BCI	77.33	**80.51**	79.47
Average	*74.67*	*73.08*	***77.73***

Table 4.1.: Averaged classification accuracy of OGNG, SSGNG and OSSGNG performed on datasets several datasets.

4.6.3. Discussion

Our experiments show the benefit of the semi-supervised OGNG (OSSGNG). It clearly outperforms SSGNG and improves the classification performance of OGNG in four out of six datasets. The two datasets (Digit1, USPS) in which the semi-supervised approaches (OSSGNG, TSVM) yield worse results compared to their original algorithms (OGNG, SVM) seem to be very easy to classify as every compared algorithm achieves an accuracy over 90%. The results of the 1-NN approach also license this observation as it performs much better on those datasets than on the others. It seems that in these cases semi-supervised learning can not improve the classification performance further.

For four out of six datasets, OSSGNG and OGNG achieve better results compared to a standard SVM (with a linear kernel) while also being comparable to other standard SSL algorithms. It is striking that OSSGNG outperforms all other approaches by far on the BCI dataset. This dataset is characterized by the availability of only few data points (400 in total) as well as by low-dimensional feature vectors (117 dimensions). OSSGNG thus seems to generalize better on low numbers of examples.

OGNG and OSSGNG show worst results on the datasets g241c and g241d. In order to shed light on this observation, we performed a PCA to reduce the dimensionality of the data to the number of principal components that capture 90% of the variance. The results are shown in Table 4.3. The analysis shows that those two datasets have a much

	1-NN (labeled data only)	SVM	OGNG	TSVM	Cluster-Kernel	Data-Dep. Reg.	LDS	OSSGNG
g241c	59.72	76.89	46.43	81.54	**86.51**	79.69	81.96	52.98
g241d	62.51	75.36	54.98	77.58	**95.05**	67.18	76.26	62.66
Digit1	93.88	94.47	**97.81**	93.49	96.21	97.56	96.54	96.77
USPS	92.36	90.25	94.35	90.23	90.68	94.90	**95.04**	93.07
COIL	76.73	77.07	77.11	74.20	78.01	**88.54**	86.28	81.45
BCI	55.17	65.69	77.33	66.75	64.83	52.53	56.03	**79.47**
Average	*73.40*	*79.96*	*74.67*	*80.63*	***85.23***	*80.07*	*82.02*	*77.73*

Table 4.2.: Averaged classification accuracy of 1-NN, SVM, OGNG, OSSGNG and standard SSL approaches on benchmarking datasets.

higher complexity (with 192 and 193 components) compared to the rest, which seems be the reason for the weak results of both OGNG and OSSGNG. Their performance is even worse than the one of 1-NN, which hints at the fact that some parts of the data are underrepresented in the OGNG/OSSGNG. This could be due to too few neurons or due to a very sparsely labeled network. In fact, the OSSGNG algorithm does not guarantee that all neurons are actually labeled. This is understandable as both datasets do not fulfill the manifold assumption, thus suffering from the curse of dimensionality in the sense that the high number of dimensions requires a number of examples that is exponential in the number of examples in order to classify them. In case of Digit1, the manifold assumption is fulfilled, which means that the high dimensionality lies on a low dimensional manifold. This explains the better performance of OGNG and OSSGNG on this dataset, in spite of its high-dimensionality.

g241c	g241d	Digit1	USPS	COIL	BCI
193	192	147	63	42	15

Table 4.3.: Number of principal components that capture 90% of the data variance.

4.7. Summary

In this chapter we have presented an extension of OGNG to an online semi-supervised classifier which in addition to OGNG predicts labels for unlabeled data in order to utilize both labeled and unlabeled data for its classification model on-the-fly. We have shown that OSSGNG renders itself ideal for tasks in which only weak or partial supervision in the form of labels is available, semi-supervised classification and late labeling in particular.

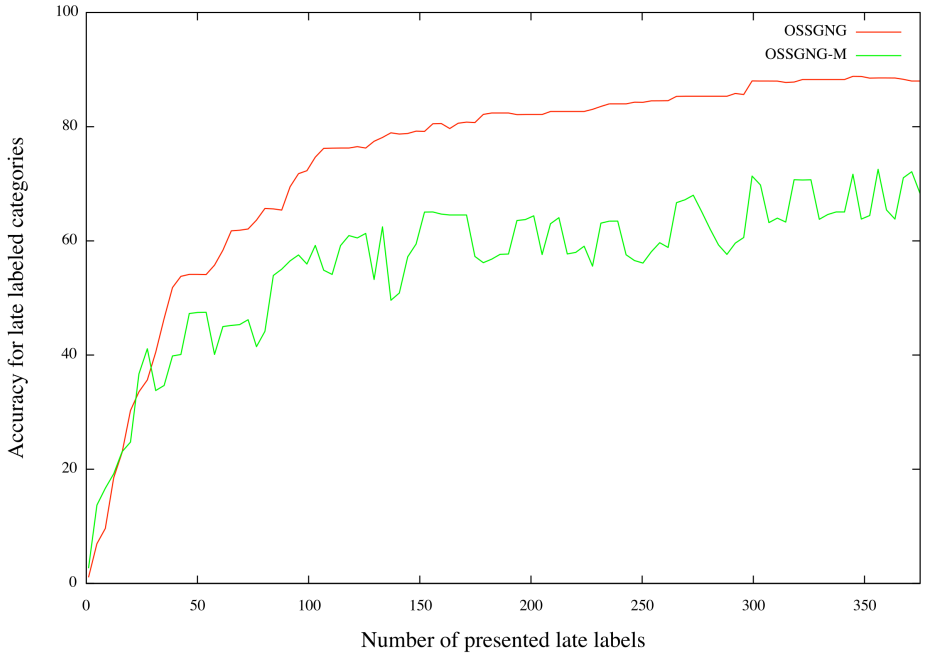


Figure 4.5.: Accuracy for late-labeled categories of OSSGNG and OSSGNG-M.

We experimentally compared OSSGNG to an existing semi-supervised learning approach based on GNG (SSGNG) which is offline in the sense that it requires to store the complete training data in order to iteratively predict unlabeled data and retrain its model afterwards. We evaluated both algorithms on benchmarking semi-supervised classification datasets and could show that OSSGNG outperforms SSGNG in 75% of our experiments while processing each data point in one pass.

We have also shown that OSSGNG improves the accuracy performance of OGNG on the datasets with up to 7.68%. On two dataset OSSGNG provides a lower performance compared to OGNG with an accuracy decrease of up to 1.28%. Those two datasets seem to be difficult for semi-supervised classification approaches, as also the transductive version of the SVM (TSVM) performs worse than the standard SVM.

Furthermore, we have shown that OSSGNG compares favorably to other standard semi-supervised classification approaches. The results of OSSGNG are better than the results of half of the four compared standard SSL approaches for four out of six datasets. On the other two datasets OSSGNG and OGNG perform worse, which most probably relates to the high complexity of those datasets.

We also investigated the adaptation ability of OSSGNG towards late labeling scenarios, in which category labels are given after the corresponding data points have already been

presented to the network. We visualized the fast adaption of OSSGNG in such scenarios and furthermore experimentally showed the benefit of an unsupervised category formation step prior to use the labels available later in time to learn these acquired categories.

In the next chapter we will introduce the second extension of OGNG, Dynamic Online Growing Neural Gas (DYNG), which is capable of quickly adapting to non-stationary stream data while solving the plasticity-stability dilemma better than existing GNG-based approaches.

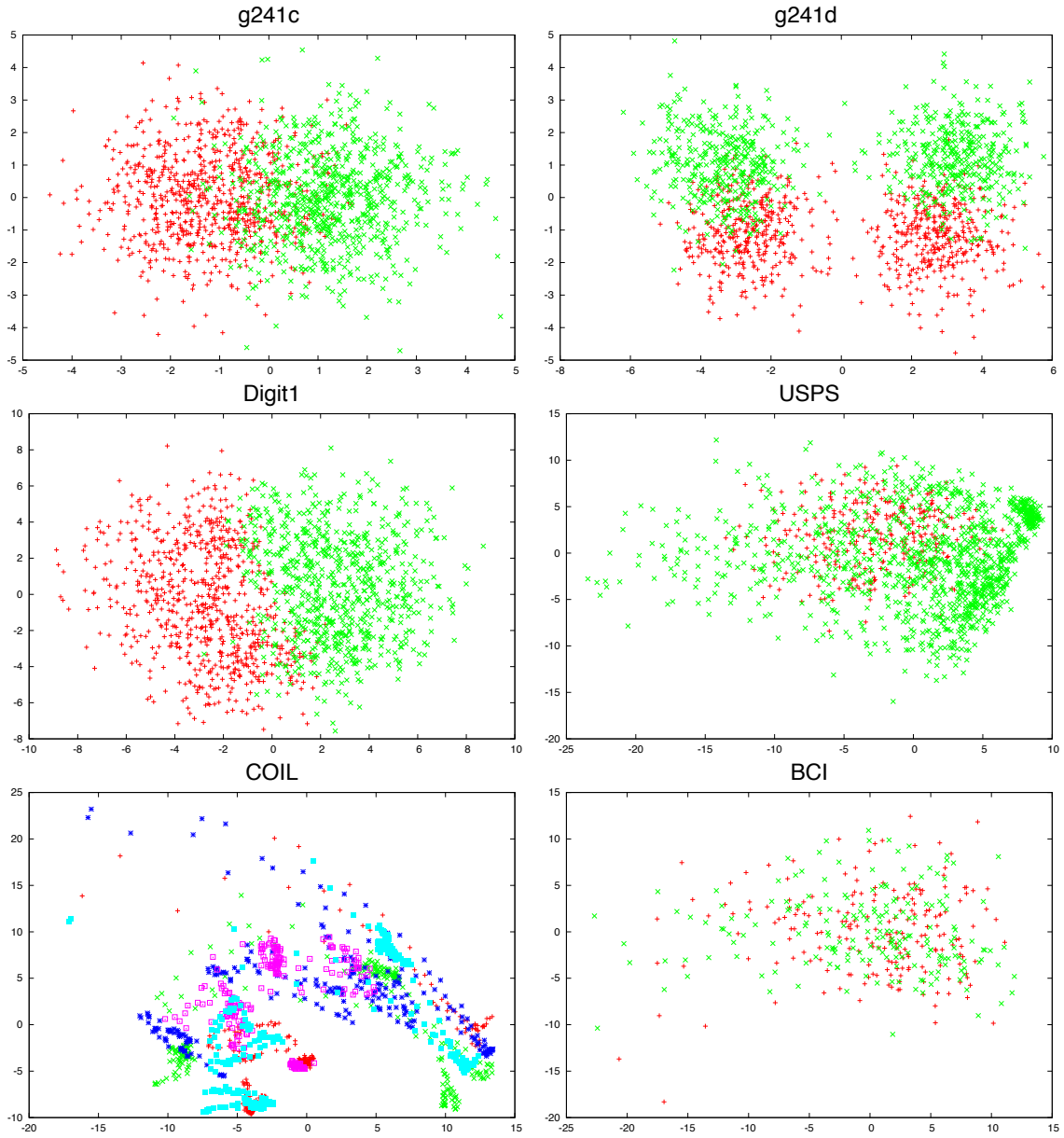


Figure 4.6.: 2D Visualizations of the SSL benchmark datasets.

Chapter 5

Learning with non-stationary data based on DYNG

In this chapter we introduce *Dynamic Online Growing Neural Gas (DYNG)*, a novel online stream data classification approach based on Online Growing Neural Gas (OGNG). DYNG exploits labelled data during processing to adapt the network structure as well as the speed of growth of the network to the requirements of the classification task. It thus speeds up learning for new categories/labels and dampens growth of the subnetwork representing the category once the classification performance converges. We show that this strategy is beneficial in life-long learning settings involving non-stationary data, giving DYNG an increased performance in highly non-stationary phases compared to OGNG. We will also compare DYNG with a novel classifier Online Growing Neural Gas with Utility (OGNG-U) based on existing GNG extensions for non-stationary data, *i.e. Growing Neural Gas with Utility (GNG-U)* (see Section 2.1.2), and furthermore show that DYNG outperforms OGNG-U in our experiments and thus provides a better solution to the plasticity-stability dilemma compared to OGNG-U.

Streams of data nowadays emerge in a number of application domains, in particular in those concerned with processing data originating from social media applications, sensor networks, news feeds, etc. [GM05]. In many scenarios, one wishes to classify the data items in these streams into a number of evolving categories. In order to scale to massive amounts of data, approaches to classify stream data should be able to work with non-stationary data, *i.e.* with changing data and category distributions and be able to detect new categories on-the-fly. In recent years, there have been a number of GNG-based algorithms that incorporate label information, such as *Incremental GNG (IGNG)* (see Section 2.1.2), *Enhanced Self-organized Incremental Neural Network (ESOINN)* [FOH07] and Semi-supervised Growing Neural Gas (SSGNG) (see Section 4.3.1). However, none

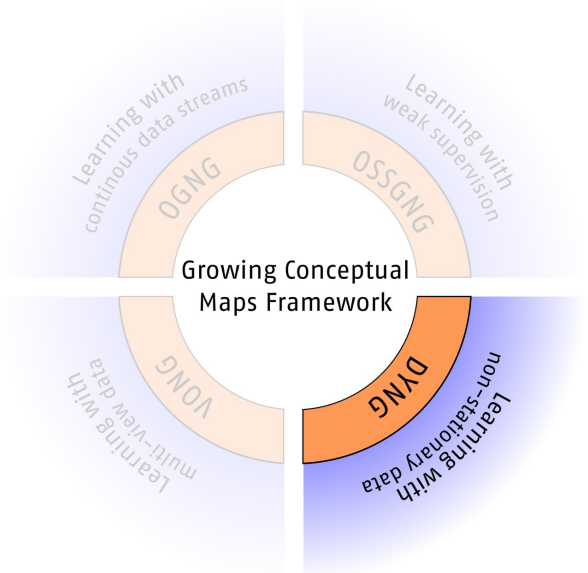


Figure 5.1.: Dynamic Online Growing Neural Gas (DYNG) as component of the GCM framework.

of these approaches are equipped with mechanisms to adapt to a changing data and category distribution as for those algorithms label information does not influence the behavior of the network, *i.e.* its growth. GNG-U, instead, is designed to follow a changing data distribution, but still needs to be extended to a classifier in order to be useful in a classification task. In this chapter, we thus present DYNG as a novel approach that utilizes label information in order to quickly adapt to non-stationary data in a classification task.

5.1. Challenges

In a classification task involving non-stationary data, a model $\mathcal{M}$ is trained to provide a solution for the function $f^t : X^t \rightarrow C^t$, with X^t being the feature input space and C^t being the corresponding category space at a certain time t. A property that uniquely applies to non-stationary data is given by the fact that the following equations can be fulfilled:

$$\exists i, j \in \mathbb{N}, i \neq j : X^{t+i} \setminus X^{t+j} \neq \emptyset \tag{5.1}$$

$$\exists k, l \in \mathbb{N}, k \neq l : C^{t+k} \setminus C^{t+l} \neq \emptyset \tag{5.2}$$

In non-stationary classification scenarios, neither data distribution, nor category distribution can be assumed as fixed and given at any point in time, which leads to the so called plasticity-stability dilemma. The challenge is to find a balance between learning flexibility and generalization. As discussed in Section 2.4, there are a number of approaches which make use of a two-layered memory structure, *i.e.* a short-term memory (STM) and a

long-term memory (LTM). However, it would be desirable to have a more uniformly classification model which implicitly incorporates similar STM and LTM mechanisms in order to be more efficient in memory usage and processing/prediction time.

For the rest of this section we will describe the challenges that arise within a classification scenario involving non-stationary data:

- **Fast adaptation towards changing data distributions**: The model $\mathcal{M}$ is required to quickly adapt to a changing data and category distribution. However, this can quickly lead to an overfitting situation in which the ability to generalize over the learned data decreases and thus the classification performance of $\mathcal{M}$ drops. It is important to find a reasonable balance between the learning flexibility and the robustness against overfitting.
- **Detection of new categories on-the-fly**: A concept drift involves the removal of categories c_j with $c_j \in C^{t-1} \wedge c_j \notin C^t$, as well as the introduction of new categories c_l with $c_l \in C^t \wedge c_l \notin C^{t-k}$ and $k = 1, 2, ..., n$. The model $\mathcal{M}$ is required to grow with the newly emerging categories and quickly adapt to them.
- **Regulation of resource requirements**: With a increasing amount of non-stationary data in a life-long learning scenario, $\mathcal{M}$ should grow accordingly following the evolving categories. However, it is necessary to retain an efficiency in processing time and memory usage, which thus requires mechanisms of regulation and optimization that balances growth on the one hand and classification performance on the other hand.

5.2. Contributions

In this chapter we are concerned with exploring how to improve the classification performance of OGNG by exploiting label information. In particular we would like OGNG to behave in the following fashion: i) it directly inserts a new neuron for an unseen label, ii) it grows dynamically as the task requires by inserting neurons as long as the error for a class decreases, dampening this growth once the classification error converges. Toward meeting these desiderata, in this chapter, we present an extension of Online Growing Neural Gas that we call Dynamic Online Growing Neural Gas (DYNG). DYNG uses label information of the presented stimulus and tracks the classification error for each class to insert neurons as long as the classification performance for this class grows. We evaluate DYNG on the task of classifying a textual stream of data on two datasets, showing that it outperforms OGNG and that it even outperforms a SVM classifier when both use a comparable amount of memory.

In particular, we offer the following contributions:

- **DYNG as extension of OGNG**: We extend OGNG by introducing two novel neuron insertion strategies in order to follow non-stationary data and a changing category distribution and thereby let the network grow according to its classification task.
- **Comparison of DYNG and OGNG-U**: We will extend an existing GNG-based approach for non-stationary data *i.e.* Growing Neural Gas with Utility (GNG-U) to the online classifier *Online Growing Neural Gas with Utility (OGNG-U)* based on labeling and prediction strategies of OGNG. We will experimentally show that DYNG clearly outperforms OGNG-U and OGNG in our experiments.
- **Comparison of DYNG and SVM**: We will compare our approach to a linear multi-class SVM and show that, although DYNG does not significantly outperform the SVM, it provides a more reliable prediction performance and is thus preferable in non-stationary data settings.

It should be mentioned that DYNG has already been published in the proceedings of the *European Symposium on Artificial Neural Networks* [BC13] in 2013.

5.2.1. Growing Neural Gas with Utility (GNG-U)

In this section we will describe *Growing Neural Gas with Utility (GNG-U)* as proposed by Fritzke [Fri97]. GNG-U is a clustering algorithm, such as GNG, with a focus on non-stationary data. The original GNG determines the utility of a neuron by its activity in a subcluster which is formed by a connected group of neurons inside the GNG network. If the subcluster is active, which means that one of the neurons of the sub cluster is winner to the presented stimulus, then the edges of all neighboring neurons age and get removed if they get older than the threshold a_{max} (see Section 2.1.2). In contrast, GNG-U introduces a utility attribute for each neuron that reflects the utility of a neuron depending on the density of its neighboring neurons, without taking the GNG links and their topology into account. GNG-U substitutes the already explained neuron removal strategy of GNG and removes neurons for which the utility decreases below a predefined fraction of the worst local error in the network. In the following, we will describe the GNG-U algorithm in detail.

5.2.2. GNG-U Algorithm

In the following, we will describe the GNG-U algorithm in detail. The algorithm is depicted in Algorithm 6. Steps which are novel compared to GNG are highlighted.

Algorithm 6 Growing Neural Gas with Utility (GNG-U)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ from the input set or according to input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: Increment the age of all edges emanating from n_1.
5: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

6: **Update the utility variable by adding the distance between $\Delta error(n_1)$ and $\Delta error(n_2)$.**

$$\Delta utility(n_1) = \Delta error(n_2) - \Delta error(n_1)$$

7: Move n_1 and all its topological neighbors (*i.e.* all the neurons connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1}), \Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
8: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
9: **Remove edges with their age larger than a_{max}. Find the neuron n_l having the smallest utility and the neuron n_m having the worst local error. Remove n_l, if the following equation is true:**

$$\frac{error(n_m)}{utility(n_l)} > k$$

10: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r according to OGNG (see Section 3.4.2).
11: **Decrease all error variables and utility variables of all neurons n_i by a factor β.**
12: If the stopping criterion is not met, go back to Step (2).

1. At first, the network is initialized with two randomly placed neurons.
2. Present the first stimulus $x \in R^n$ of the feature input space to the GNG-U network.
3. In this step, we find the winner and second winner n_1 and n_2 which are closest to x, according to the Euclidean distance.
4. The age of all edges that connect n_1 to other neurons is increased by 1.
5. In this step, the local error variable $error(n_1)$ is updated. This variable later helps to identify candidates for which a new neuron is inserted in between.
6. **According to this step, the utility variable $utility(n_1)$ of the winner neuron is updated by adding the distance between $\Delta error(n_1)$ and $\Delta error(n_2)$.**
7. An adaptation of n_1 and its topological neighbors towards x by fractions e_b and e_n, respectively, is performed.
8. A new connection between n_1 and n_2 is created and the age of the edge is set to 0.

9. **All edges with an age greater than a_{max} are removed. Additionally, the neuron n_m having the lowest utility gets removed in case the utility is smaller than the fraction k of the worst error in the network.**

10. In this step, depending on the iteration and the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.

11. **The error and utility variables of all neurons are decreased by a factor β.**

12. In the last step, the algorithm stops if the stopping criterion is met, *i.e.*, the maximal network size or some other performance measure has been reached.

5.2.3. Limitations of GNG-U

One of the disadvantages of GNG-U is its high sensitivity towards rapidly changing data distributions. Preliminary experiments have shown that the network completely moves to a new position in feature space in case the data distribution change involves a relatively high Euclidean distance between the old and new position of the presented data. This sensitivity is understandable, as such a high distance in Euclidean space leads to a high local error. As GNG-U removes neurons if their utility falls below a fraction of the worst local error, which is high in such scenario, the complete network will get deleted and built anew at the new position in feature space. Depending on the frequency of such dramatical change, the network would remain unstable unless the parameter k is set to a high value accordingly. Basically, k is responsible for the balance between plasticity and stability and this also is a problem, as it is required to carefully be set for each specific application domain due to the destructive nature of GNG-U. In contrast, our approach DYNG provides a non-destructive approach which does not influence the removal strategy of GNG and makes DYNG more insensitive towards extreme changes in the data distribution.

It should also be mentioned that GNG-U is not a classification algorithm and thus is required to be extended accordingly. In Section 5.5, we will briefly describe the extension of GNG-U to Online Growing Neural Gas with Utility (OGNG-U) by making use of the online labeling and prediction strategies introduced in Chapter 3.

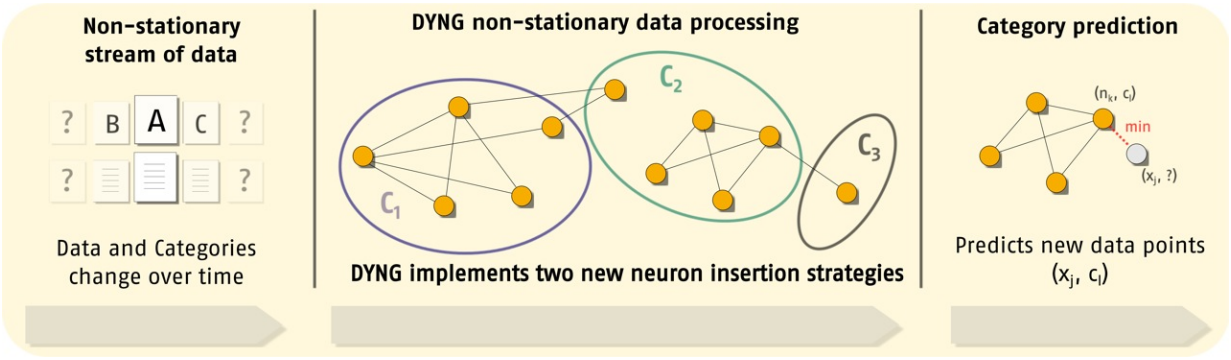


Figure 5.2.: Dynamic Online Growing Neural Gas (DYNG) processing pipeline.

5.3. Dynamic Online Growing Neural Gas (DYNG)

In this section we introduce *Dynamic Online Growing Neural Gas (DYNG)* as extension of OGNG with an emphasis on the processing of non-stationary data. At first, we will explain the additions that set DYNG and OGNG apart. Then we will explain the DYNG algorithm in detail, before we then discuss its implicit two-layered memory structure consisting of a short-term memory (STM) and a long-term memory (LTM).

DYNG is a classification algorithm that exploits label information in order to react to concept drifts by introducing two novel neuron insertion strategies. As shown in Figure 5.2 from left to right, the DYNG algorithm processes input given as data category pairings and results in a classification model which is capable of predicting unseen data points on-the-fly. Its main advantage lies in its two novel neuron insertion strategies that allow DYNG to follow non-stationary data and to quickly react to concept drifting categories.

5.3.1. DYNG as extension of OGNG

OGNG is an online classifier which is capable of processing and predicting data on-the-fly, without the need to store any training example explicitly. Although OGNG is capable of processing a continuous stream of data, it lacks the ability to react to changing distributions, as its neuron insertion strategy does not incorporate the label information of the data. In contrast, DYNG utilizes the label information in order to quickly react to changing data and category distributions.

DYNG introduces two novel neuron insertion strategies which depend on the category label information and allow DYNG to outperform OGNG when processing non-stationary data, as shown in Section 5.5. In particular, the two new neuron insertion strategies can be described as follows:

- **Fast-mapping insertion strategy**: The first insertion strategy is responsible for a fast adaptation towards novel categories. As soon as a new stimulus x_i with a novel category c_j is presented to the network, a new neuron n_{new} is inserted at the position of x_i with $l^t(x_{new}) := c_j$.
- **Distance-based insertion strategy**: The second novel neuron insertion strategy inserts new neurons for each category after every iteration, as long as its classification performance increases. The purpose of this strategy is to quickly establish a reasonable representation of each category and in particular of new categories. The strategy works as follows: In case of a mismatch between the label of n_1 and the label of stimulus x_i, DYNG determines the nearest neuron n_{lb} with $l^t(n_{lb}) = l^t(x_i)$ and inserts a new neuron if the distance $|\omega_{n_{lb}} - x_i|^2 \geq |\omega_{n_{lb}} - x_i|^2\tau + |\omega_{n_1} - x_i|^2$, i.e. if n_{lb} is too far away from x_i compared to n_1. In the other case n_{lb} is adapted towards x_i. Thereby, $|\omega_{n_{lb}} - x_i|^2\tau + |\omega_{n_1} - x_i|^2$ can be seen as a vigilance parameter as known from *Adaptive Resonance Theory (ART)* [Gro87] and thus controls the size of the additional network memory provided by DYNG. The distance-based insertion strategy is illustrated in Figure 5.3.

The Fast-mapping insertion strategy is applied in every iteration if required, while the distance-based insertion strategy is only applied if the classification performance of category $l^t(x_i) = c_j$ improved at iteration t compared to iteration $t - \lambda$. In order to track this performance, similar to the local error, we introduce a function $classError : C \times T \to \mathbb{R}$ with C being the set of known categories and T being an iteration number. We denote by $classError^t(c_i)$ the classification error of category c_i, which is defined as follows:

$$classError^t(c_i) = \sum_{s=1}^{t} \begin{cases} 1 - \frac{\Delta error^s(w_s(x_s))}{\max_{n \in N}(error^s(n))} & \text{if } c_i = l^s(w_s(x_s)) \\ 0 & \text{else} \end{cases}$$

In each iteration, we only apply this function to a category in case of mismatch between $l^t(w_s(n_s))$ and $l^t(x_s)$. For every update, a value between $[0 : 1]$ is added to the classification error of c_i, depending on the distance between winner neuron $w_s(x_s)$ and stimulus x_s. The intuition is that a misclassification of x_s with $w_s(x_s)$ and x_s being close, is worse than a misclassification with both being far away from each other. Although $classError^t(c_i)$ potentially holds a complete classification error history for each known category c_i until iteration t, we only store the last value and compare it to the most recent one after λ iterations have passed. This decreases the memory usage for each category to $\mathcal{O}(1)$.

5.3.2. DYNG Algorithm

In what follows, we will describe every step of the DYNG algorithm. The complete algorithm is depicted in Algorithm 7.

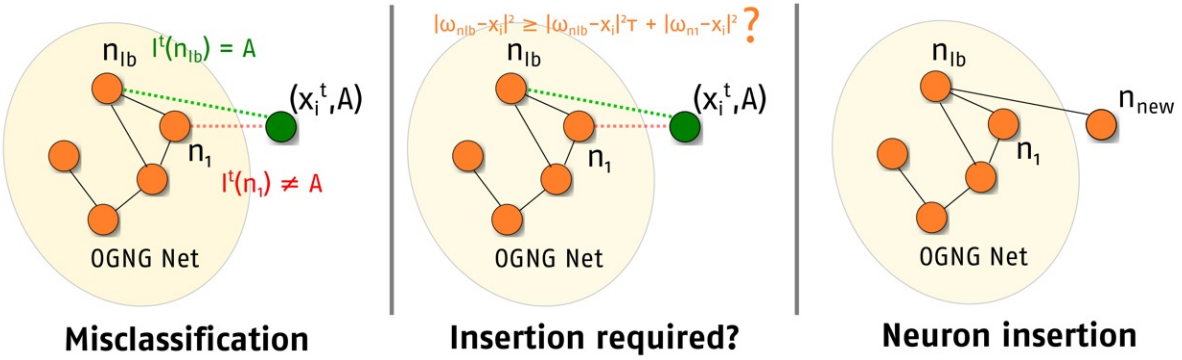


Figure 5.3.: Illustration of the Distance-based insertion strategy of DYNG.

- In the first step, the network is initialized with two randomly placed neurons n_i, n_j.
- In the second step, the iteration loop starts which runs as long as stream input is available, which theoretically could be infinity.
- In Step 3, the next stimulus x_i is presented to the network at iteration t.
- According to Step 4, the winner n_1 and second winner neuron n_2 are determined according to the Euclidean distance between neurons and stimulus.
- Steps 5-8 describe the fast-mapping insertion strategy. In these steps, a new neuron n_{new} is inserted into the network at the position of x_i in feature space. It adopts the label $l^t(x_i)$ of the stimulus x_i and gets connected to the previously determined winner neuron n_1.
- In step 9, the error variable of n_1 is updated according to OGNG (see Section 3.4.2) and the emanating edges of n_1 increment their age by one.
- In steps 10-12, the classification error variable of category $l^t(n_1) = c_1$ is updated with $\Delta classError^t(c_1) = 1 - \frac{\Delta error^t(n_1)}{\max_{n \in N}(error^t(n))}$.
- In Step 13, the winner neuron n_1 is labeled according to the used online labeling strategy. In our case we used the relabel method (see Section 3.4) throughout our experiments, which simply adopts the label of the stimulus.
- In Step 14, n_1 and its topological neighbors are adapted towards the stimulus by the fraction of e_b and e_n, respectively.
- Steps 15-23 form the distance-based insertion strategy, in which basically three conditions need to fulfilled in order to insert a new neuron at the position of the stimulus. The first two of them are the requirements that $l^t(n_1) \neq l^t(x_i)$ and $impl(l^t(x_i)) =$ "$true$", which means that the labels of neuron n_1 and stimulus x_i

differ from each other, as well as that the classification error of category $c_j = l^t(x_1)$ improved over the last $t - \lambda$ iterations. If both criteria are fulfilled, the algorithm determines the closest neuron n_{lb} to x that shares its label with $l^t(n_{lb}) = l^t(x)$. The last condition decides if the distance between n_{lb} and the stimulus is still close enough to adapt n_{lb} or to insert a new neuron instead. In case that a new neuron is inserted, it will get connected to n_{lb}.

- In Step 24, all edges and neurons are removed according to the criteria of OGNG.
- According to steps 25-34, the classification error of all known categories at iteration t is compared to those of iteration $t - \lambda$. If the classification error decreased then the boolean function $imp : C \rightarrow \{true, false\}$ is set to the value "true" in order to indicate that the classification performance for the particular category improved. Otherwise it is set to "false". Additionally we insert a new neuron according to the OGNG algorithm.
- In Step 35, the error variables as well as the classification errors for all neurons and all categories, respectively, are decreased by the factor β. After that we continue the loop cycle with the next stimulus and Step 2.

Complexity in time and space

The additions we introduce do not increase the overall complexity of DYNG compared to OGNG. As in the case of OGNG, we need to run through all neurons in order to determine the nearest neighbor, to adapt the neurons' positions, its local errors, as well as increase link ages. Assuming the whole network is stored as a b-tree using the in Section 2.1.2 introduced TreeGNG schema, then each search, insertion or removal operation has a complexity of $\mathcal{O}(log_2 n)$ with n being the number of neurons. As for OGNG, the complexity of the labeling and the prediction is in $\mathcal{O}(1)$ and in $\mathcal{O}(log_2 n)$, respectively, when the relabel method and the single-linkage prediction strategy is used. Our new additions , the fast-mapping strategy and the distance-based strategy have a complexity of $\mathcal{O}(1)$ and $\mathcal{O}(log_2 n)$, respectively. However, steps 26-32 have a complexity of $\mathcal{O}(log_2 k)$, with k being the number of known categories. Under the assumption that only categories remain stored that are present in the network, we have a worst case complexity of $\mathcal{O}(log_2 n)$. So overall, DYNG has a average and worst case time complexity of $\mathcal{O}(log_2 n)$.

The memory usage of DYNG is again similar to OGNG, as we only need to store the position of each neuron, its connection, its local error and its label. DYNG additionally stores the all the known categories and their current and last (of iteration $t-\lambda$) classification error, which in worst case is in space complexity of $\mathcal{O}(n)$.

Algorithm 7 Dynamic Online Growing Neural Gas (DYNG)

Start with two units n_i and n_j at random positions in the input space.
while *Stream.hasNext* **do**
  $x_i := Stream.next(t)$ with $x_i \in \mathbb{R}^n$.
  Find the nearest unit n_1 and the second nearest unit n_2.
  if $l^t(x_i)$ is unknown **then**
    Add a neuron n_{new} with $\omega_{n_{new}} = x_i$ and $l^t(n_{new}) = l^t(x_i)$.
    Create an edge between n_{new} and n_1.
  end if
  Update the local error variable of n_1 and increment the age of all edges emanating from n_1 according to OGNG (see Section 2).
  if $l^t(n_1) \neq l^t(x_i)$ **then**
    Update the class error: $\Delta classError^t(c_1) = 1 - \frac{\Delta error^t(n_1)}{\max_{n \in N}(error^t(n))}$, with $l^t(n_1) = c_1$.
  end if
  $l^t(n_1) := l^t(x_i)$ {OGNG relabel-method}
  Move n_1 and all its topological neighbors according to OGNG and connect n_1 and n_2.
  if $l^t(n_1) \neq l^t(x_i) \wedge imp(l^t(x_i)) =$ "$true$" **then**
    Find nearest unit n_{lb} with $l^t(n_{lb}) = l^t(x_i)$.
    if $|\omega_{n_{lb}} - x_i|^2 \geq |\omega_{n_{lb}} - x_i|^2\tau + |\omega_{n_1} - x_i|^2$ **then**
      Add a neuron n_{new} with $\omega_{n_{new}} = x_i$ and $l^t(n_{new}) = l^t(x_i)$.
      Create an edge between n_{new} and n_{lb}.
    else
      Move n_{lb} towards x_i by the fraction of e_b: $\Delta\omega_{n_{lb}} = e_b(x_i - \omega_{n_{lb}})$
    end if
  end if
  Remove edges and neurons according to OGNG.
  if $t \mod \lambda = 0$ **then**
    for all classes $c_i \in C$ **do**
      if $classError^{t-\lambda}(c_i) > classError^t(c_i)$ **then**
        $imp(c_i) =$ "$true$"
      else
        $imp(c_i) =$ "$false$"
      end if
    end for
    Insert a new neuron and update the network link structure according to OGNG.
  end if
  Decrease all local error variables of all neurons n_i and all class error variables of all classes c_i by a factor β.
  $t++$.
end while

So overall, DYNG has a average and worst case time complexity of $\mathcal{O}(log_2 n)$ and a complexity in space of $\mathcal{O}(n)$.

5.3.3. Memory Structures

In the following we want to highlight the two-layered memory structure. Similar to the two-level architecture, we explained in Section 3.4.3, DYNG provides a two-layered memory structure which is only to some degree explicitly implemented. One of the layers is explicitly implemented, while the other one is subject of our interpretation. Nevertheless, it is important to deepen the discussion of the memory architecture of DYNG in order to understand how DYNG works. As illustrated in Figure 5.4 and already mentioned before,

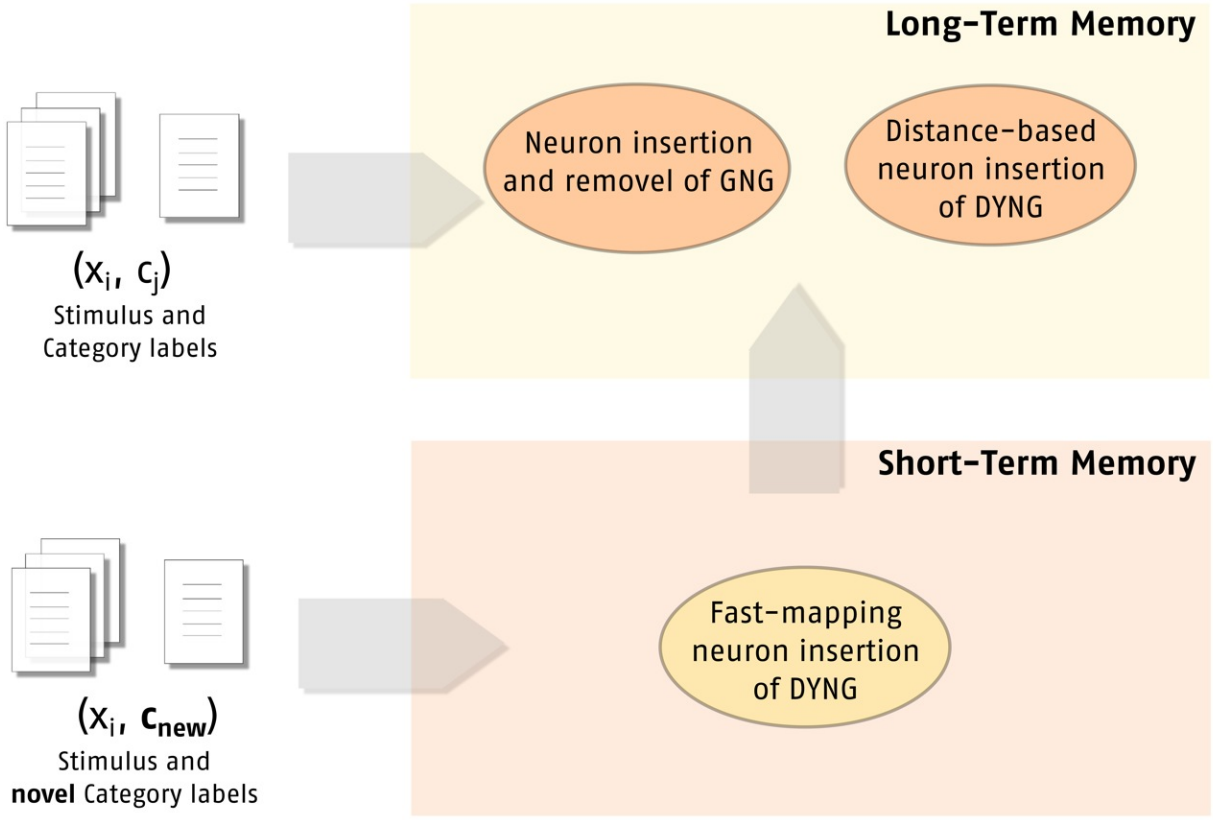


Figure 5.4.: Two-layered memory structure schema of DYNG.

DYNG includes a short-term memory (STM) and a long-term memory (LTM). The STM is explicitly implemented by the fast-mapping insertion strategy, as it causes novel categories to directly manifest in the DYNG network. However, those categories may or may not evolve in the network, depending on how frequently they appear in the data. Due to the implicit LTM based on the distance-based insertion strategy and the GNG-based neuron insertion and removal strategies, novel categories will expand or disappear.

The structure shows that DYNG, similar to other existing approaches, balances the plasticity-stability dilemma by providing a STM and LTM, which can be considered as one of the main advantages of DYNG compared to other GNG-based approaches, such as GNG-U. The STM allows the model to directly adapt to new unknown categories, while the LTM allows those categories to rapidly grow inside the network.

5.4. Dataset

In this section we describe the dataset we used in our experiments with DYNG. It should be mentioned that we evaluate DYNG on a small artificial dataset in this chapter, in order to show its performance compared to OGNG-U, OGNG and an linear multi-class SVM in a small controlled environment. In Chapter 7, we will further demonstrate the advantage of DYNG when applied to realistic stream datasets.

However, for our evaluation in this chapter we use the SPIRAL1 dataset, which we already introduced in Section 3.5 and used with OGNG. In contrast to the separation of the dataset according to the 10-fold cross validation, here we use a set up which is closer to a realistic stream dataset. In particular, there is no separation between different training sets as we only interrupt the processing in order to perform a prediction for unseen data as a test to quantify the classification accuracy of the network. The test set, which we use for the prediction, is formed by taking 50% of all examples for each category and excluding them from the set of data used for training. We refer to the current state of the data sets (train/test) as *snapshots*. Two of such snapshots are shown in Figure 5.5. The figure shows the available data for training (left side) and test (right side) at snapshot 4 and snapshot 8.

In the next section we will further describe how the data evolves over time and which data is available in each snapshot.

5.5. Evaluation

In this section we evaluate DYNG as extension of OGNG in a stream data scenario involving non-stationary data and concept drift. We will compare DYNG to Online Growing Neural Gas with Utility (OGNG-U), an online classification algorithm which is based on GNG-U while adopting online labeling and prediction strategies of OGNG. We use OGNG-U and OGNG as baselines for our experiments and will experimentally show that DYNG significantly outperforms OGNG-U and OGNG, without substituting the underlying neuron insertion and removal strategies of GNG as by OGNG-U. We also compare DYNG to the standard OGNG algorithm in order to show the improvement of our additions towards non-stationary data, as well as to a linear multi-class SVM.

5.5.1. Parameters & OGNG-U

In order to compare DYNG to our baselines, we chose a fixed set of GNG parameters. The set has been empirically determined by trial-and-error through preliminary experiments.

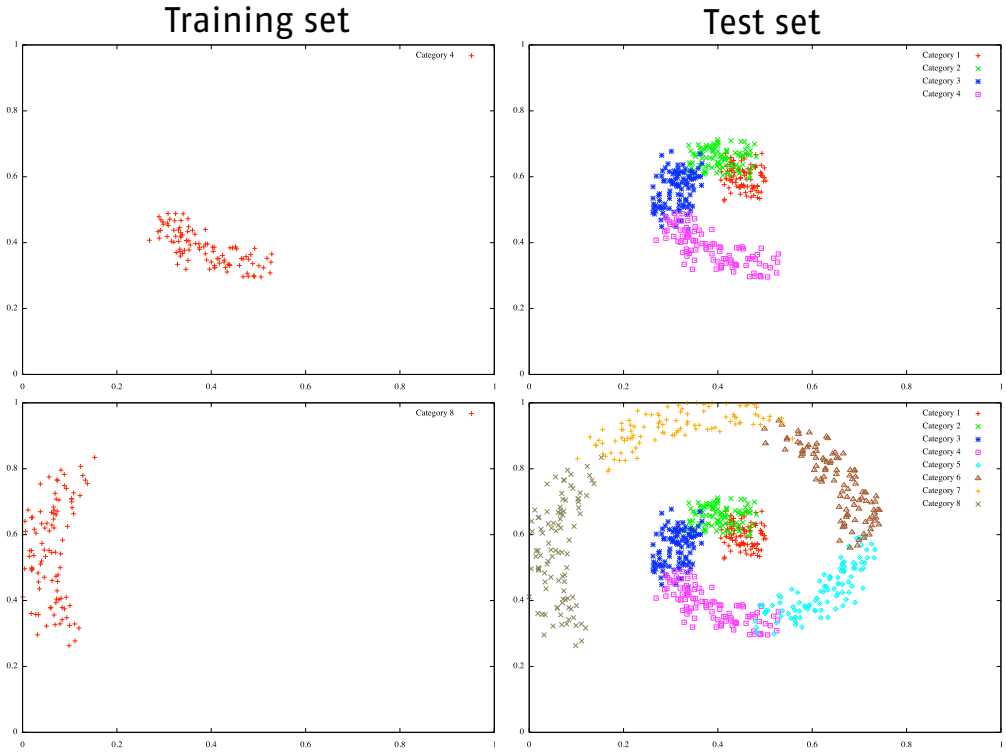


Figure 5.5.: Training and test set of SPIRAL2 at Snapshot 4 (upper) and Snapshot 8 (lower).

For our experiments we have set the GNG parameters for DYNG, OGNG-U and OGNG as follows: insertion parameter $\lambda = 30$; maximum age $a_{max} = 100$; adaptation parameter for winner $e_b = 0.1$; adaptation parameter for neighborhood $e_n = 0.0006$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.0005$ and allowed a maximum number of neurons $n_{max} = 120$. For OGNG-U we additionally set the GNG-U parameter $k = 3.0$, as we achieved best results with it. The factor allows GNG-U to have a higher memory, compared to $k = 1.0$, as the worst utility may be three times smaller than the worst local error of the network. For DYNG we set $\tau = 0.3$ as we achieved best results in all of our experiments, including the real stream data experiments of Chapter 7. For the SVM we chose a multi-class SVM with a linear kernel and trained it in a one-vs-all fashion and relied on the commonly used SVM library libSVM [CL11].

As GNG-U is a clustering algorithm that is not capable of processing category label information, we extended GNG-U to the online classifier *Online Growing Neural Gas with Utility (OGNG-U)*. OGNG-U simple adopts the online labeling method relabel and the prediction strategy single-linkage of OGNG as introduced in Section 3.4.1 and Section 3.3.1. OGNG-U furthermore demonstrates the general applicability of the mechanisms introduced in this thesis beyond GNG to other sorts of topological feature maps. The

novel strategies, in particular the ones introduced by DYNG, do not modify or substitute the underlying clustering schema, as they simple extend it.

5.5.2. Experiments & Results

For our experiments with DYNG we evaluated the classification performance of DYNG, OGNG-U, OGNG and the SVM on the SPIRAL1 dataset. Thereby, we separated the dataset so that 50% of all examples of each category will be shown through every snapshot. In particular, we only show one category at a time for each snapshot, while having 10 snapshots overall. This means that in Snapshot 2, for example, we exclusively show the examples for Category 2 while in Snapshot 3 we exclusively present the stimulus of Category 3, *etc.*. The test set, instead, gradually grows, as with every snapshot a new category is added to it. Therefore, a prediction is basically performed on all categories seen so far. An illustration of Snapshot 4 and Snapshot 8 is given by Figure 5.5. In contrast to previous experiments, the stopping criterium for OGNG (and OGNG-U) is that all data has been processed.

For the SVM we needed to adapt the scenario slightly, as the SVM is an offline classifier which is required to explicitly store the training examples and retrain in order to incorporate newly gained knowledge. In order to make a fair comparison, we allowed the SVM a storage of $n = 112$ training examples, with n being equal to the number of neurons of DYNG. While the data evolves during the different snapshots, the SVM is required to equally represent the relevant categories with the given amount of memory and thus is required to insert and remove examples from its training set. Those removal operations are based on the *first in - first out (FIFO)* principle.

Our results are shown in Table 5.1 and Figure 5.6. Table 5.1 shows the averaged classification accuracy including standard deviation of all compared algorithms in the first row. Additionally the table lists the number of vectors, which includes neurons and also training examples and support vectors for the SVM in the second row. This should indicate the memory demand of each algorithm throughout this evaluation. In Figure 5.6, we depicted the development of the accuracy curve for DYNG, OGNG-U and OGNG over the processing of the data set. Thereby, the x-axis stands for the number of snapshots and the y-axis reveals the corresponding accuracy.

Results

The results license the following observations:

	DYNG	OGNG-U	OGNG	SVM
Accuracy	**75.1 ± 2.9%**	58.9 ± 6.0%	58.8 ± 3.6%	73.8 ± 16.5%
#Vectors	112	100	102	224

Table 5.1.: Averaged classification performance of DYNG, OGNG-U, OGNG and SVM on SPIRAL2 and number of stored vectors.

- **Comparison of DYNG and OGNG-U**: According to the first row of Table 5.1, DYNG significantly outperforms OGNG-U with an average accuracy of $75.1 \pm 2.9\%$ compared to $58.9 \pm 6.0\%$. DYNG also provides a lower standard deviation which additionally is observable in Figure 5.6. The figure shows that especially in the beginning phase of the processing at Snapshot 2, DYNG outperforms OGNG-U by an increased performance of 27%. It also confirms the lower standard deviation of DYNG in that its accuracy is more stable compared to OGNG-U over the processing.

- **Comparison of DYNG and OGNG**: The first row of Table 5.1 also shows that DYNG significantly outperforms OGNG with an average accuracy of $75.1 \pm 2.9\%$ compared to $58.8 \pm 3.6\%$. Compared to OGNG, DYNG increases the performance by 27% and also shows a slightly better stability in performance with a standard deviation of 2.9% compared to OGNG with 3.6%.

- **Comparison of OGNG-U and OGNG**: The results in the first row of Table 5.1 show that OGNG-U provides a slightly better averaged accuracy compared to OGNG with a classification performance of $58.9 \pm 6.0\%$ compared to $58.8 \pm 3.6\%$. The standard deviation of OGNG-U is twice as much compared to OGNG with 6.0% compared to 3.6%. This difference becomes clear when inspecting both accuracy curves as depicted in Figure 5.6. The figure shows that OGNG-U starts with an accuracy that is 8.7% below the one of OGNG at Snapshot 2 and gradually increases until it outperforms OGNG with a difference of 3.4% at Snapshot 7, up to 3.6% at Snapshot 10.

- **Comparison of DYNG and SVM**: As shown in the first row of Table 5.1, DYNG performs slightly better compared to the SVM with a classification accuracy of $75.1 \pm 2.9\%$ compared to $73.8 \pm 16.5\%$. We applied a t-test and could not prove the results to be significantly different. An interesting observation, however, is that the standard deviation is almost 6 times as high as the standard deviation of DYNG, with 16.5% compared to 2.9% of DYNG.

- **Memory usage**: In the second row of Table 5.1 the used number of vectors are shown. In case of DYNG, OGNG-U and OGNG this reflects the number of neurons that the network has stored. In case of the SVM the number consists of the stored training examples and used support vectors. The SVM has the highest memory usage

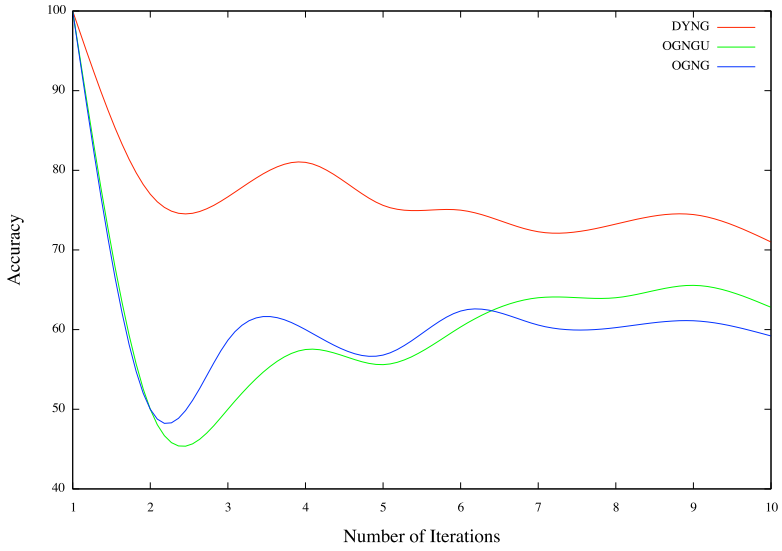


Figure 5.6.: Performance development of DYNG, OGNG-U and OGNG on the SPIRAL2 dataset.

with 224 vectors, followed by DYNG with 112 vectors, OGNG with 102 vectors and OGNG-U with 100 stored vectors.

5.5.3. Discussion

The results of our experiments show the advantage of DYNG in a stream data scenario which involves non-stationary data as well as concept drift. DYNG significantly outperforms all other compared GNG-based approaches. Furthermore, it even performs slightly better than the SVM, although we do not consider the difference as being significant. However, the GNG-based classifiers allow a more reliable prediction with an almost up to 6 times less standard deviation. This reliability is very important for stream data and lifelong learning applications, as a prediction is performed on demand and it is thus crucial for the usability of such a classification model. DYNG shows the best performance and highest reliability and thus renders itself as most ideal for stream data scenarios compared to OGNG-U, OGNG and the SVM. The high degree of instability of the SVM is understandable as the amount of training examples for each retraining of the classifier seems to be insufficient and causes some of the categories to be underrepresented while others are reasonable represented by the given set of support vectors.

Comparing the memory usage of all algorithms, the SVM demands the highest amount of memory. However, within the GNG-based classifiers, DYNG requires a 12% increased

memory amount compared to OGNG-U, which is due to its non-destructive nature. In relation to the performance improvement of DYNG with a 16.2% increased accuracy compared to OGNG-U, the slightly higher memory usage should be considered as reasonable.

5.6. Summary

In this chapter we have presented DYNG as extension of OGNG with emphasis on non-stationary data and concept drift. DYNG introduces two novel neuron insertion strategies which allow the model to quickly grow and adapt to new evolving categories and, furthermore, damping this growth when the classification performance for a category converges. We have also introduced OGNG-U as online classifier based on GNG-U and used it as baseline for our experiments. We discussed the short-term and long-term memory of DYNG and how the different neuron insertion strategies and neuron removal strategies realize the two memory structures. We have evaluated DYNG, OGNG-U, OGNG and a linear multi-class SVM in a stream data scenario which involves non-stationary data and concept drift. Thereby, we shown that DYNG significantly outperforms the GNG-based classifiers, especially in the beginning phase of the processing. Furthermore, DYNG achieves slightly better results than the SVM while providing a six times more reliable prediction accuracy. We also compared the memory usage of all algorithms and found the SVM as being most memory demanding. Out of the GNG-based algorithms, DYNG requires the most memory due to its non-destructive nature. The increase of memory usage, however, can be considered as reasonable when taking into account its increased performance.

In the next chapter, we will introduce the last extension of OGNG within the GCM Framework that is capable of processing data coming from multiple domains in a single classification network.

Chapter 6

Learning with multi-view data based on VONG

In this chapter, we introduce *Multi-view Online Growing Neural Gas (VONG)* as an extension of OGNG. VONG is an online classifier that is capable of processing data coming from multiple domains. As each of these domains renders a unique view on the data, VONG predicts category labels pre view. Therefore, it introduces a new multi-view labeling and multi-view prediction strategy that involve the selection of relevant feature space dimensions, masking them for each view independently. We additionally introduce a naive multi-label classification approach, *i.e.* OGNG$^+$, that is also based on OGNG and serves us as baseline in our evaluation. We will compare both algorithms on two artificial multi-view datasets and show that VONG significantly outperforms OGNG$^+$ on both of the sets. Furthermore, we will investigate the impact of the new multi-view strategies on the classification performance independently, and therefore compare VONG to VONG-M. VONG-M is a version of VONG which only implements its multi-view prediction strategy, but adopts the naive multi-labeling of OGNG$^+$.

While standard multi-view online classification approaches follow the assumption that views are generated by the same domain, we extend the multi-view classification challenge in that we assume the data to be composed out of multiple domains, and therefore each view being generated by a different domain. Thereby, we define a domain as a subset of feature space dimensions and a set of unique subcategory/property labels. As a consequence to our extended challenge, each data instance is assigned to multiple domains and thus requires multiple subcategory labels to be specified. This causes most of the existing multi-view classification approaches to be unusable for our task. Usually, an ensemble of classifiers would be used in such scenario, which is not an architecture of choice in life-long learning due to its increased complexity and memory requirement. However, in this chapter we will demonstrate that it is possible to solve our multi-view classification task by providing a single homogenous model that is capable of predicting a category label for an unseen data points in each view-independently.

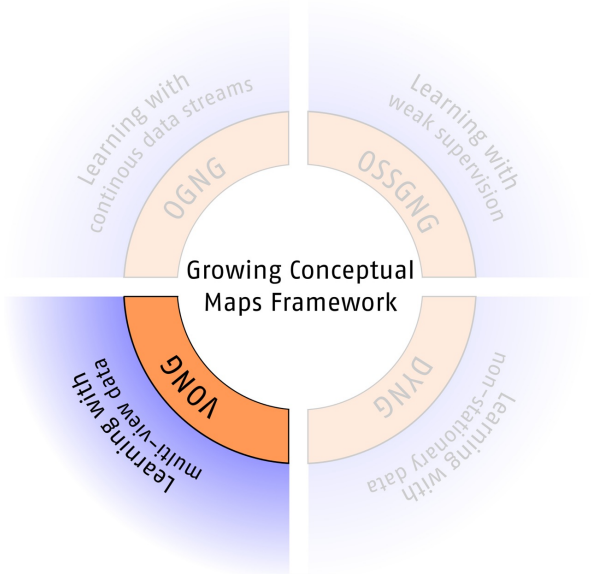


Figure 6.1.: Multi-view Online Growing Neural Gas (VONG) as component of the GCM framework.

6.1. Challenges

In our multi-view classification scenario, the model $\mathcal{M}$ is required to learn the mapping between a data point $x_i \in X$ and a set of subcategory labels $c'_j \in C'$, which are given as tuple (x_i, c'_j). Thereby, feature vectors from X as well as sets of subcategory labels from C' are formed as product of feature subspaces $X = \prod_{k=1}^{o} X_{d_k}$ and subcategory labels $C' = \prod_{k=1}^{o} C_{d_k}$ coming from the different domains $d_k \in D$, with $k = 1, 2, ..., o$.

A "red book", for example, has a certain color, shape and material. Given these domains of "color", "shape" and "material", the data instance of a "red book" (x_i, c'_j) can be described by its composed feature vector x_i and its set of subcategory labels $c'_j = \{$"red", "square", "paper"$\}$. Thereby, we assume the domains and, in particular, their feature subspaces and sets of subcategory labels to be disjoint. For our example, this means that there are no feature subspace dimensions as well as no subcategory labels that are relevant to more than one of the three domains.

A *view* can then be seen as projection

$$\pi : X \times C' \times D \rightarrow \prod_{k=1}^{o} X_{d_k} \times \prod_{k=1}^{o} C_{d_k}, \text{ with}$$

$$\forall k, l, d_l \in D, k \neq l : X_{d_k} = \emptyset \wedge C_{d_k} = \emptyset.$$

The projection function π thereby masks feature space dimensions and excludes category labels that are irrelevant to d_k. In what follows, we denote this masking projection function by $\pi_{d_k}(x_i, c'_j)$.

Given this setting, the following challenges arise:

- **Learning with multiple views**: Usually a classifier is trained to solve a classification task by learning the prediction function $f : X \to C$. However, in our multi-view classification scenario the model $\mathcal{M}$ is required to learn an extended prediction function $f_{multi} : X \times D \to C$, which depends on a domain $d_k \in D$. Furthermore, the conditional independence of domains requires $\mathcal{M}$ to perform multiple classification tasks at once, without them contributing to each other as, for example, in the Co-Training paradigm (see Section 2.3).

- **Grouping of categories by views**: When learning a prediction function f_{multi} per view, the feature space needs to be masked accordingly. This means that $\mathcal{M}$ should incorporate several masking functions π_{d_k} for the different domains, in order to project the input x_i to the feature subspace of d_k and to assign subcategory labels of C_{d_k} to the model $\mathcal{M}$.

- **Compact uniform classification model**: Providing a uniform model in a multi-view classification task is challenging, as the data is heterogenous caused by its compositional nature and its feature subspaces and subcategories coming from multiple domains. Representing such data in a homogenous model requires a step of encoding and decoding in the sense that the diverse data is uniformly represented in $\mathcal{M}$ (encode) and still being recallable by its domain affiliation (decode).

6.2. Contributions

In this chapter, a multi-view extension of OGNG is presented that relies on online multi-view labeling and multi-view prediction strategies. We will make use of our novel approach in an online classification task that involves data coming from several domains. Thereby, we evaluate VONG on two artificial multi-view datasets that including different degrees of domain correlations. We will furthermore introduce OGNG$^+$ as naive multi-label extension of OGNG, and utilize this algorithm as baseline for VONG, while showing that VONG clearly outperforms OGNG$^+$ on both datasets. In order to investigate the impact of the masking on the classification performance of VONG, we also compare VONG against VONG-M, a version of VONG which excludes the masking in when labeling neurons.

In particular, we offer the following contributions:

- **VONG as multi-view extension of OGNG**: We extend OGNG by a novel on-the-fly multi-view labeling and prediction strategy which allow VONG to learn with data coming from multiple domains and store this information in a single network.
- **OGNG$^+$ as naive multi-label extension of OGNG**: We extend OGNG with a naive multi-labeling strategy that builds upon the labeling strategies of OGNG, in order to provide an intuitive baseline that we can compare with.
- **Comparing VONG with OGNG$^+$**: We compare the performance of VONG and OGNG$^+$ in a classification task and show that VONG clearly outperforms OGNG$^+$ on both of our artificial multi-view datasets.
- **Comparing VONG with VONG-M**: We compare VONG to VONG-M and demonstrate the benefit of the masking for each view, which is performed during the assignment of labels to neurons in the VONG network.

6.3. OGNG$^+$: A naive multi-label approach based on OGNG

In this section we introduce OGNG$^+$ as naive multi-label extension of OGNG. OGNG$^+$ will serve us as baseline for VONG as it provides an intuitive approach on adapting OGNG to a multi-view data classification problem. OGNG$^+$ adopts the architecture of OGNG as online classifier and simply modifies its online labeling and online prediction strategies, which is based on the relabel method (see Section 3.4.1) and the single-linkage prediction strategy (see Section 3.3.2). In contrast to the relabel method of OGNG, OGNG$^+$ adopts the complete subcategory label set as given by the stimulus. Therefore, m labels are assigned to each single neuron in the network, with $m = |D|$ being the number of all domains relevant to this task. For the single-linkage prediction strategy, OGNG$^+$ simply returns all subcategory labels of the neuron which provides the closest Euclidean distance to the stimulus.

6.3.1. OGNG$^+$ Algorithm

In the following, the complete OGNG$^+$ algorithm will be described step by step, as depicted in Algorithm 8. The highlighted steps mark the changes of OGNG$^+$ compared to the original OGNG.

1. In the first step, the algorithm starts with two neurons, randomly placed in the feature space.

2. **The first stimulus $x \in R^n$ of the input space (first training example) and its corresponding set of labels is presented to the network.**
3. The two neurons n_1 and n_2 which minimize the Euclidean distance to x are identified as first and second winner.
4. **The set of labels is assigned to n_1. The relabel method is used here (see Section 3.4.1)**
5. The age of all edges that connect n_1 to other neurons is increased by 1.
6. In this step, the local error variable $error(n_1)$ of n_1 is updated. This error variable will be used later in order to determine the location for a newly inserted node.
7. In this step, n_1 and its topological neighbors are adapted towards x by fractions e_b and e_n, respectively.
8. A new connection between n_1 and n_2 is created and the age of the edge is set to 0.
9. All edges with an age greater than a_{max} as well as all neurons without any connecting edge are removed.
10. Depending on the iteration and the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.
11. In this step, the error variables of all neurons are decreased by a factor β.
12. The algorithm stops if the stopping criterion is met, i.e., the maximal network size or some other performance measure has been reached.

6.3.2. Limitations of OGNG$^+$

There are several limitations connected to OGNG$^+$. The first problem is the fact that OGNG$^+$ is not capable of predicting the subcategory of a specific view, as it simply returns all subcategory labels. This also means that OGNG$^+$ does not include mechanisms of encoding and decoding the data coming from multiple domains, as the information about domain affiliations get lost during the data processing step. In contrast, in the next section we will discuss that VONG is capable of storing the domain affiliation in the network implicitly by utilizing its masked labeling and prediction strategies.

Another disadvantage of OGNG$^+$ is closely connected to this lack of storing domain affiliations. When assigning the set of subcategory labels to neurons, the winner neuron is

Algorithm 8 Online Growing Neural Gas$^+$ (OGNG$^+$)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ and its corresponding set of categories from the input set according to the input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: Assign the complete label set of x to n_1 according to the present labeling strategy.
5: Increment the age of all edges emanating from n_1.
6: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

7: Move n_1 and all its topological neighbors (*i.e.* all the neurons connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1})$$

$$\Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
8: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
9: Remove edges with their age larger than a_{max}. If this results in neurons having no emanating edges, remove them as well.
10: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r as follows:
Determine unit n_q with the largest error.
Among the neighbors of n_q, find node n_f with the largest error.
Insert a new node n_r halfway between n_q and n_f as follows:

$$\omega_{n_r} = \frac{\omega_{n_q} + \omega_{n_f}}{2}$$

Create edges between n_r and n_q, and n_r and n_f. Remove the edge between n_q and n_f.
Decrease the error variable of n_q and n_f by multiplying them with a constant α. Set the error n_r with the new error variable of n_q.
11: Decrease all error variables of all neurons n_i by a factor β.
12: If the stopping criterion is not met, go back to Step (2).

determined by having the closest Euclidean distance to the stimulus with consideration of the complete unweighted feature vector. This means that a view having the highest number of dimensions in the feature vector potentially biases the complete decision towards its subcategory. As a result, the classification performance for such views should be higher than those only contributing a small number of dimensions. VONG masks the dimensions of the feature vector for its labeling and prediction strategy in the sense that it simply ignores dimensions that are irrelevant to the view in question.

6.4. Multi-view Online Growing Neural Gas (VONG)

In this section, we introduce *Multi-view Online Growing Neural Gas (VONG)* as multi-view extension of OGNG. At first, we will describe the modifications of VONG that set

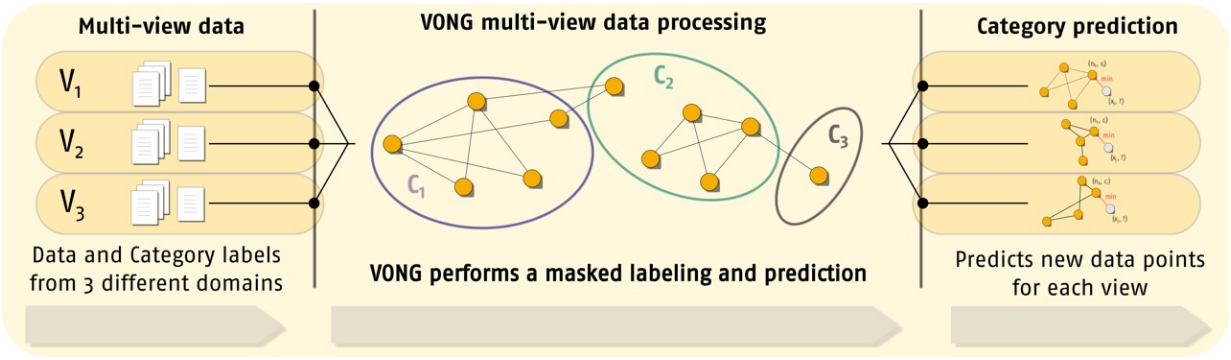


Figure 6.2.: Multi-view Online Growing Neural Gas (VONG) processing pipeline.

both VONG and OGNG apart. We will then explain the VONG algorithm in detail and also discuss its multiplexing architecture.

VONG is an online classifier that utilizes labeled data coming from multiple domains and encodes this information in a uniform classification model on-the-fly. It furthermore is capable of decoding the stored information in the sense that it implements the prediction function f_{multi} and thus predicts subcategory labels to a given stimulus and a given domain. As shown in Figure 6.2 from left to right, a composed feature vector as well as a set of subcategory labels from multiple domains are the input for our VONG algorithm. The data then gets processed and stored inside the uniform VONG network, which then can predict subcategory labels for the single views on demand. Its main advantage lies in its uniform structure, which allows the model, as in all other extensions of OGNG, to grow with a stream data classification task while remaining a reasonable compact form.

6.4.1. VONG as multi-view extension of OGNG

OGNG is an online classification algorithm which is capable of processing stream data on-the-fly. However, it is not designed to work with multi-view data and, even more important, it is not designed to predict multiple category labels for a single given input stimulus. In order to extend OGNG accordingly, VONG modifies the labeling and prediction strategies of OGNG to assign and predict subcategory labels to a given stimulus and a given domain. Thereby, the algorithm requires the domain affiliation, and thus view affiliation, of the feature space dimensions to be known, and furthermore utilizes this information in order to mask the feature vector per view, to ignore irrelevant dimensions. In what follows we will describe these extensions in more detail:

- **Masking of the feature vector**: The process of masking dimensions of the feature vector is crucial to VONG as it allows, in combination with the labeling and

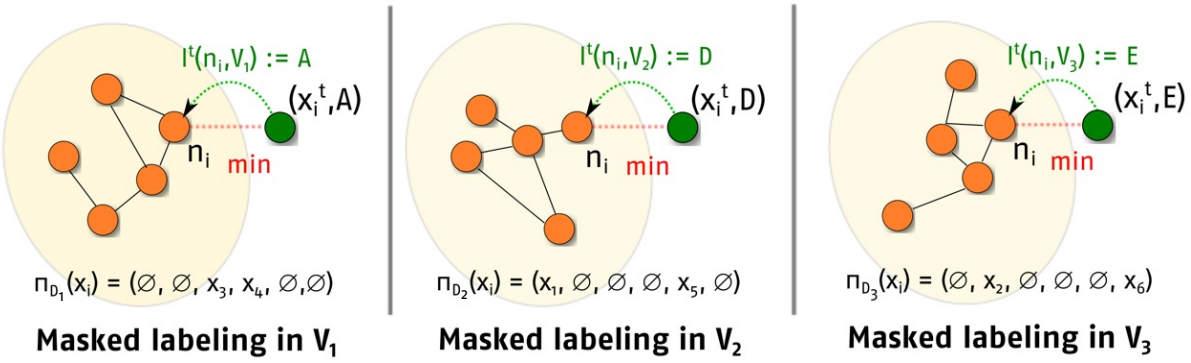


Figure 6.3.: Illustration of the masked multi-view labeling strategy according to VONG.

prediction strategies, to encode and decode the heterogenous data which originates from multiple domains. Thereby, it is assumed that the domain/view affiliation of feature space dimensions to be known. As illustrated in Figure 6.3 and Figure 6.4, the process of masking feature space dimensions causes the VONG network to change its shape per view, so that a subcategory label prediction can be performed in the feature subspace of X_{d_k}.

- **Multi-view labeling strategy**: This labeling strategy is based on the feature space masking and is depicted in Figure 6.3. As shown in the figure, for each view that is been generated by a domain, we perform a single labeling according to the relabel method of OGNG. We thus simply find the closest neuron of the network by only considering the relevant dimensions according to the view in question, and assign the subcategory label to this winner neuron.

- **Multi-view prediction strategy**: In case of the prediction strategy, we also rely on the masking of dimensions and only consider the relevant dimensions of the feature vector for the view in question, to determine the closest neuron in the network. Thereby, the prediction is based on single-linkage, as introduced in Section 3.3.2 and illustrated in Figure 6.4.

It should be mentioned that there is a strong assumption been made in order to apply VONG in the way described. The affiliation of feature dimensions and subcategory labels to a specific view/domain must be known in advance. VONG requires this information in order to apply its masking when labeling data. In the prediction step, no additional information is required, as the masking vector is explicitly stored for each view. However, this assumption is realistic in the sense that in many real applications, such as in data warehouses, the origin of the data is usually known. The assumption does thus not prevent VONG from being applicable in real multi-view based stream data scenarios.

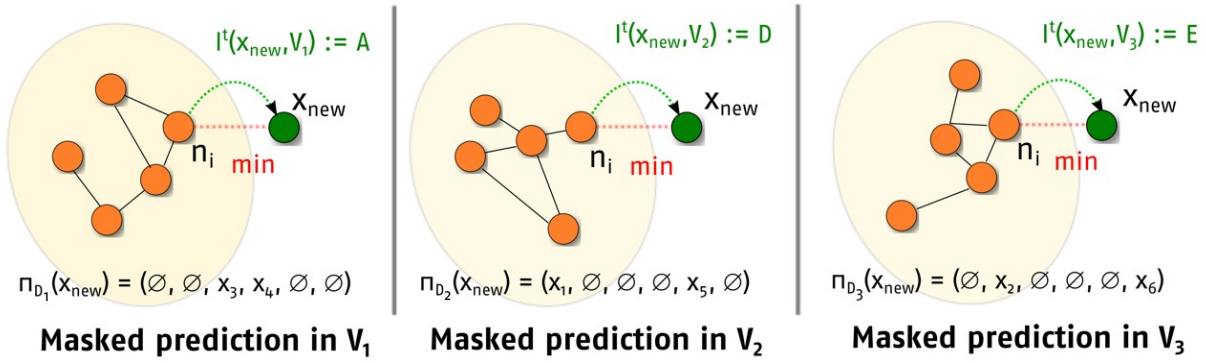


Figure 6.4.: Illustration of the masked multi-view prediction strategy according to VONG.

6.4.2. VONG Algorithm

In the following, the complete VONG algorithm will be described step by step, as depicted in Algorithm 9. The highlighted steps mark the changes of VONG compared to the original OGNG.

1. In the first step, the algorithm starts with two neurons, randomly placed in the feature space.
2. **The first stimulus $x \in R^n$ of the input space (first training example) and its corresponding set of subcategory labels is presented to the network.**
3. The two neurons n_1 and n_2 which minimize the Euclidean distance to x are identified as first and second winner.
4. **For each view, find the closest neuron $n_1^{(d_k)}$ according to the masked Euclidean distance, and assign $n_1^{(d_k)}$ with the label of the corresponding view label of x.**
5. The age of all edges that connect n_1 to other neurons is increased by 1.
6. In this step, the local error variable $error(n_1)$ of n_1 is updated. This error variable will be used later in order to determine the location for a newly inserted node.
7. In this step, n_1 and its topological neighbors are adapted towards x by fractions e_b and e_n, respectively.
8. A new connection between n_1 and n_2 is created and the age of the edge is set to 0.
9. All edges with an age greater than a_{max} as well as all neurons without any connecting edge are removed.

10. Depending on the iteration and the parameter λ, a new node n_r is inserted into the network. It will be inserted half-way between the neuron n_q with the highest local error and its topological neighbor n_f having the largest error among all neighbors of n_q. In addition, the connection between n_q and n_f is removed and both neurons are connected to n_r.

11. In this step, the error variables of all neurons are decreased by a factor β.

12. The algorithm stops if the stopping criterion is met, i.e., the maximal network size or some other performance measure has been reached.

Algorithm 9 Multi-view Online Growing Neural Gas (VONG)

1: Start with two units n_i and n_j at random positions in the input space.
2: Present an input vector $x \in R^n$ and its corresponding set of categories from the input set according to the input distribution.
3: Find the nearest unit n_1 and the second nearest unit n_2.
4: For each view, find the closest neuron $n_1^{(d_k)}$ according to the masked Euclidean distance and assign the label of x to $n_1^{(d_k)}$.
5: Increment the age of all edges emanating from n_1.
6: Update the local error variable by adding the squared distance between ω_{n_1} and x.

$$\Delta error(n_1) = |\omega_{n_1} - x|^2$$

7: Move n_1 and all its topological neighbors (*i.e.* all the neurons connected to n_1 by an edge) towards x by fractions of e_b and e_n of the distance:

$$\Delta\omega_{n_1} = e_b(x - \omega_{n_1})$$

$$\Delta\omega_n = e_n(x - \omega_n)$$

for all direct neighbors of n_1.
8: If n_1 and n_2 are connected by an edge, set the age of the edge to 0 (refresh). If there is no such edge, create one.
9: Remove edges with their age larger than a_{max}. If this results in neurons having no emanating edges, remove them as well.
10: If the number of input vectors presented or generated so far is an integer or multiple of a parameter λ, insert a new node n_r as follows:
Determine unit n_q with the largest error.
Among the neighbors of n_q, find node n_f with the largest error.
Insert a new node n_r halfway between n_q and n_f as follows:

$$\omega_{n_r} = \frac{\omega_{n_q} + \omega_{n_f}}{2}$$

Create edges between n_r and n_q, and n_r and n_f. Remove the edge between n_q and n_f.
Decrease the error variable of n_q and n_f by multiplying them with a constant α. Set the error n_r with the new error variable of n_q.
11: Decrease all error variables of all neurons n_i by a factor β.
12: If the stopping criterion is not met, go back to Step (2).

Complexity in time and space

Equally to the other GNG extensions of the GCM framework, VONG does not increase the complexity in time and space compared to OGNG. GNG itself as described in Section 2.1.1, has on average and worst case a time complexity of $\mathcal{O}(log_2 n)$ and a complexity in space of $\mathcal{O}(n)$, due to its storage in a TreeGNG. VONG modifies the labeling in a way that it is required to determine the nearest neighbor in each view, and therefore requires to compare the stimulus to each of the neurons k times, with $k = |D|$ being the number of views. This leads to an overall time complexity to be in $\mathcal{O}(k\ log_2 n)$.

Concerning the complexity in space, VONG requires to store all neurons, equally to OGNG and GNG. In order to store the weights of the feature vectors according to the masking process, k additional vectors are required to be stored, with $k = |D|$ being the number of views. For each view, a single weight vector is required and as we can assume the number of views to be less than the number of neurons required, the additional space complexity lies in $\mathcal{O}(n)$. Thus, the overall complexity in space of VONG lies in $\mathcal{O}(n)$.

6.4.3. Multiplexing of views

In this section, we want to discuss the multiplexing architecture of VONG and address how the encoding and decoding process manifests in the multi-view labeling and multi-view prediction strategy. In Figure 6.5 the architecture of VONG is illustrated. According to the figure, VONG also consists of a two-level architecture as introduced with OGNG (see Section 3.4.3). Equally to OGNG, these levels are not explicitly implemented but implicitly arise out of the mechanics of VONG. The two-levels distinguish between a feature level (lower part of the image) and a category level (upper part of the image). As for OGNG, the shape of the network is exclusively determined by the complete feature space of the input vectors in the feature level. At this level, no feature space masking is performed when adapting to the stimulus, which means that VONG delivers an equal feature level map compared to OGNG and GNG. However, the feature space masking is performed in the category level for each view, and therefore allows VONG to store and recall the subcategory label information in the network while retaining its view/domain affiliation. Similar to a multiplexer and demultiplexer, the masked labeling process can be seen as encoding of the data (multiplexer), while the masked prediction decodes the information stored in the network per view (demultiplexer).

However, an interesting and important fact is that the data is not explicitly encoded in the network, as it does not distinguish between subcategory labels from one or another view. Instead, the encoding and decoding is only carried out by the masked labeling and prediction strategies. The feature level therefore can be seen as a representative model of the actual data, while the masked labeling and masked prediction of the category level

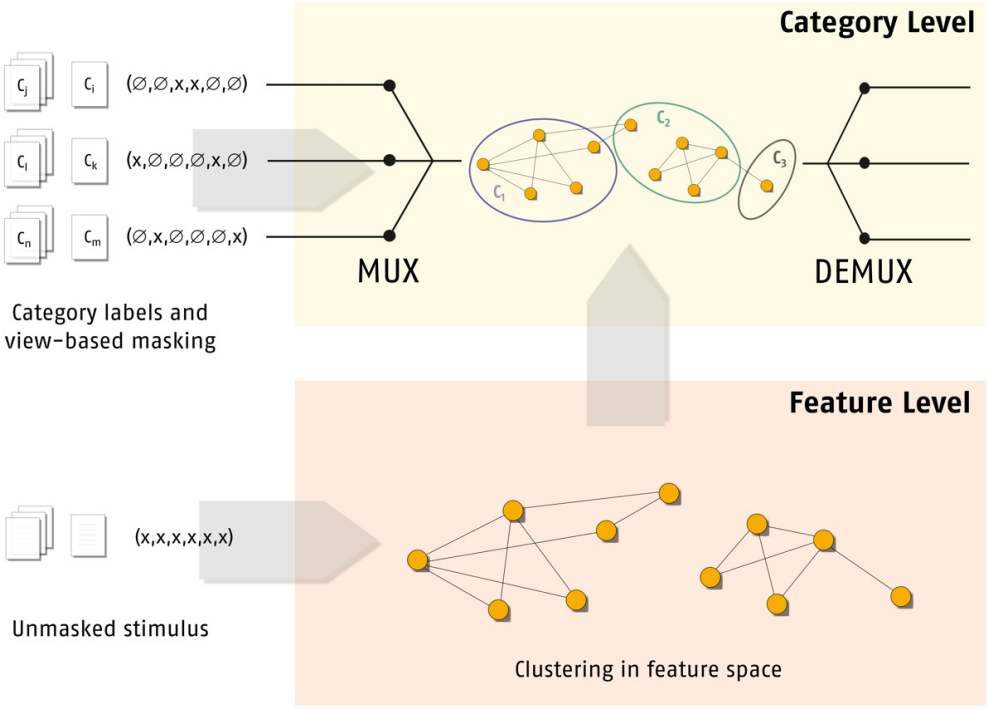


Figure 6.5.: The implicit multiplexer architecture of VONG.

generate individual views on the model. This implicit data encoding in a single network is an advantage of VONG, as it allows VONG to retain its uniform classification model, and thus completely inherits the advantages of OGNG.

6.5. Datasets

In this section, we describe the datasets used in our evaluation of VONG. We introduce two artificial multi-view datasets, in order to show specific effects in a small controlled environment. However, as VONG inherits all attributes of OGNG, it is possible to use VONG in a stream data scenario in which the task is to classify data coming from multiple domains.

SPIRAL2

For this dataset we generate three two-dimensional domains, inspired by standard clustering benchmarking sets. In Figure 6.6, each domain of the dataset is illustrated. As shown

in the figure, each domain consists of five subcategories, *i.e.* Category 1-5 (domain D_1), Category I-V (domain D_2) and Category A-E (domain D_3). The first domain D_1 (top), is generated similarly to SPIRAL1 as introduced in Chapter 3. It is an Archimedean spiral $r(\phi) = a\phi$ with ϕ and $r(\phi)$ being the polar coordinates and $a = 1$. The spiral was generated with an angle between [0:720] and equally divided into five categories (144 degree per category) holding overall 500 data points. The data was normalized and translated so that its values range between [0:1]. As for SPIRAL1, we randomly sampled data points for each category and each angle ranging form $(x-k, y-k) - (x+k, y+k)$ with $k = 0.047$.

For the second domain D_2 (bottom left), we generated five circles according to the standard parametric equation $x = x_M + rcos(\alpha)$ and $y = y_M + rsin(\alpha)$, with (x_M, y_M) being the center of the circles and r being the radius. The categories are generated from Category I-V with a radius of $r = \{0.2, 0.4, 0.6, 0.8, 1.0\}$. We also normalized and translated the data to provide a value range of [0:1], and randomly sampled data points for each category and each angle ranging form $(x-k, y-k) - (x+k, y+k)$ with $k = 0.012$.

For the third domain D_3 (bottom right), we generated five half opened circles for the Categories A-E at the positions $(0.1, 0.7), (0.3, 0.3), (0.5, 0.7), (0.7, 0.3), (0.9, 0.7)$. The angle of the half opened circle ranges from [0:200] with alternating sign, which means that for Category A it ranges from [0:200], for Category B it ranges from [0:-200], *etc.* The choice of an angle of 200 degrees is aimed at delivering a slight overlap along the x-axis.

It is also important to mention that the domains D_1 and D_3 are conditional depend, which means that a bijective mapping function between the categories of both domains exist.

OBJ-SCT

This data set is based on colored 300×300 pixel images showing various simple geometric objects. Those objects are colored and include one of five textures. In Figure 6.8, five example images are depicted. The dataset consists of 5225 instances coming from three domains, *i.e.* shape, color, texture. As for the shapes, we chose the five simple geometric shapes “circle”, “ellipse”, “rectangle”, “square” and “triangle” as shown in the figure. For the colors, we chose 11 colors of the *Munsell Color Schema*, considering the color perception of a US native speaker according to data provided in the *World Color Survey Database (WCS)* [CKR05]. The textures are based on the Image Segmentation Dataset [Bla98] of the UCI machine library, which we already used in our OGNG experiments in Chapter 3. In particular, the textures are “brickface”, “sky”, “foliage”, “cement”, “window”, “path” and “grass”.

For the shape features, we measured the distance from the center of the image to the outer contour of the object for the angles $\alpha = \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$. This scan procedure and the eight scans $s_1 - s_8$ are illustrated in Figure 6.7 (left). For

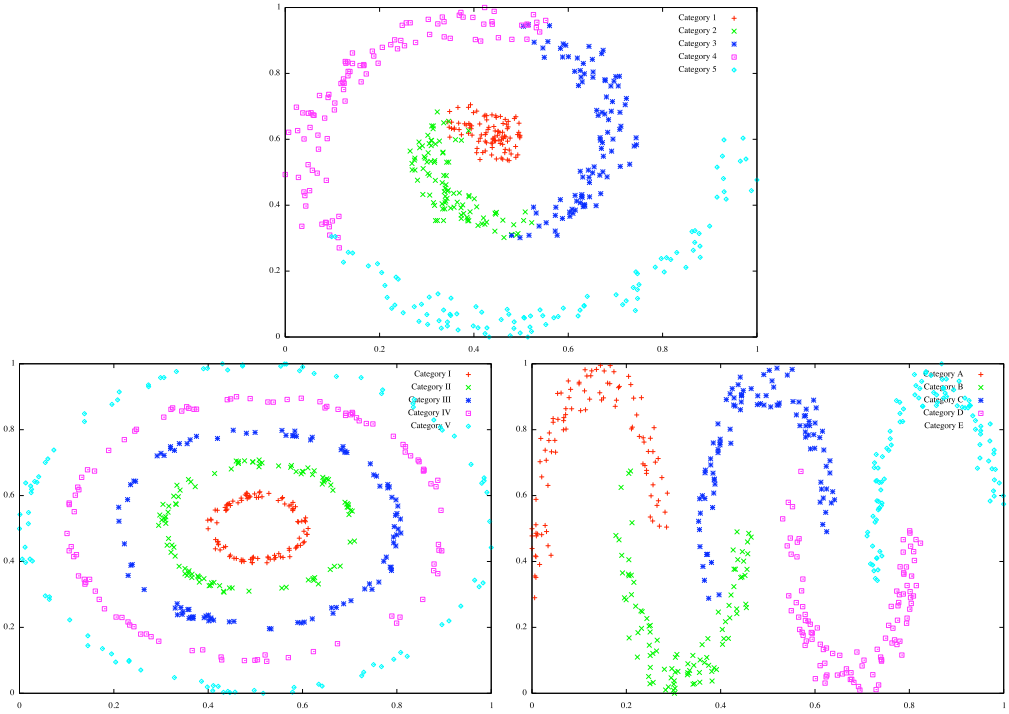


Figure 6.6.: All three views on the dataset SPIRAL2, including five categories per domain.

the color features, we simply averaged the color of the image in the RBG space, only considering pixel information inside of the outer contour of the object. For the texture features, we performed a *Local Fourier Transformation (LFT)* according to Zhou *et al.* [ZFS01]. Therefore, as illustrated in Figure 6.7 (right), we consider a 3×3 pixel patch as a periodic signal that is assumed to be descriptive for the texture. We then divide the image into four patches and applied the proposed LFT algorithm on each of them. For each patch, we then average over quantized bins that are based on the LFT coefficients. As a result, we extract 16 values including the average and the standard deviation for the extracted bins for each patch. We compose those patches to a 64 dimensional feature vector. A more detailed description of the algorithm can be found in Zhou *et al.* [ZFS01].

We specifically chose the two datasets, in order to compare a heterogenous set of domains to a homogenous set of domains. SPIRAL2 is more homogenous in the sense that it provides an equal number of five categories and two dimensions for each domain. OBJ-SCT, in contrast, includes eight dimensions and five categories for the shape domain, 11 categories with three dimensions for the color domain and seven categories with 64 dimensions for the texture domain. Another difference between both datasets is the fact that D_1 and D_3 of SPIRAL2 are correlated, while all other dimensions of both datasets are uncorrelated.

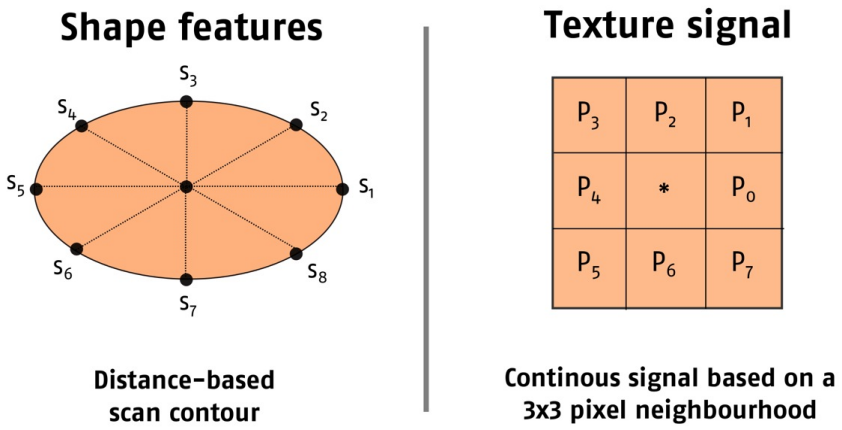


Figure 6.7.: Shape features extracted by a contour scan (left) and the continuous texture signal based on a 3x3 pixel neighborhood (right).

6.6. Evaluation

In this section, we evaluate VONG as multi-view extension of OGNG. We experimentally compare VONG to OGNG$^+$ as naive multi-label classification approach based on OGNG, which serves us as baseline in our experiments. We will show that VONG outperforms OGNG$^+$ on both of our datasets. Furthermore, we will show the benefit of the masked multi-view labeling strategy of VONG, as we additionally compare VONG to VONG-M, a version of VONG which does not mask dimensions when assigning labels to neurons.

6.6.1. Parameters & VONG-M

For the evaluation, we chose a fixed set of GNG parameters to use throughout all of our experiments with VONG. The set was determined through preliminary experiments on the basis of trial-and-error and can be listed as follows: insertion parameter $\lambda = 30$; maximum age $a_{max} = 100$; adaptation parameter for winner $e_b = 0.1$; adaptation parameter for neighborhood $e_n = 0.0006$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.0005$ and allowed a maximum number of neurons $n_{max} = 1000$. Compared to other experiments with GNG-based classifiers in this thesis, the maximum number of neurons is about 10 times higher than in the other experiments. The reason for this lies in the fact that OBJ-SCT includes a 10 times higher amount of data points compared to SPIRAL1, for example. We used those values for both datasets to maintain a comparability.

In order to investigate the impact of the masked multi-view labeling strategy of VONG, we developed VONG-M as version of VONG which implements the multi-view prediction

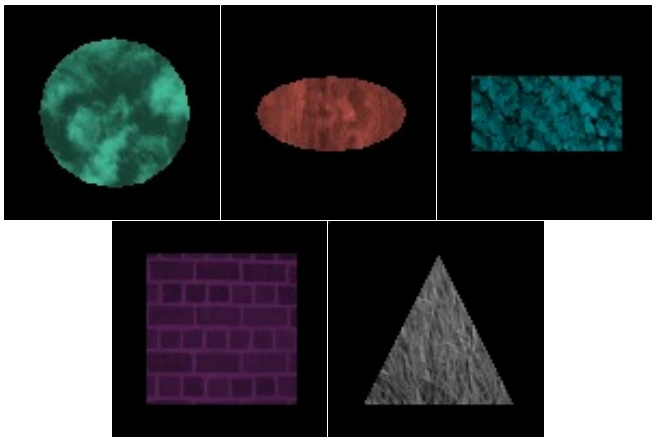


Figure 6.8.: Five samples of the OBJ-SCT dataset.

strategy of VONG and also adopts the naive multi-labeling of OGNG$^+$. VONG-M is thus still capable of predicting subcategory labels per view.

6.6.2. Experiments & Results

For our experiments with VONG, we performed a 10-fold cross validation and evaluated the precision of the algorithms for our two datasets SPIRAL2 and OBJ-SCT. We also considered the *F1-score* as evaluation measure, but since the algorithms always deliver the correct number subcategories per view, with a $recall = 1.0$, we chose *precision* to be the only evaluation metric used in our experiments. Precision is defined as $prec = \frac{TP}{TP+FP}$, with TP being *true positives* and FP being *false positives*.

	VONG	VONG-M	OGNG$^+$
SPIRAL2	**99.3%**	87.4%	86.0%
OBJ-SCT	**80.9%**	74.7%	65.0%
Average	**90.1%**	81.1%	75.5%

Table 6.1.: Micro-averaged precision of VONG, VONG-M, OGNG$^+$.

The results of our experiments are shown in Table 6.1, with the best individual result highlighted in each row. The table shows the average *micro-averaged precision* of VONG, VONG-M and OGNG$^+$ on the datasets SPIRAL2 and OBJ-SCT, as well as an average performance for the algorithms averaged over the datasets in the last row. For the micro-averaged precision, we average the precision per fold and then average over all folds.

Results

The results license the following observations:

- **Comparison of VONG and OGNG$^+$**: According to Table 6.1, VONG significantly outperforms OGNG$^+$ on both datasets. For SPIRAL2, VONG scores with an average precision of 99.3% compared to 86.0% of OGNG$^+$. For OBJ-SCT, VONG yields an average precision of 80.9% compared to 65.0%.
- **Comparison of VONG and VONG-M**: According to Table 6.1, VONG also outperforms VONG-M for SPIRAL2 as well as for OBJ-SCT. For SPIRAL2, VONG provides a performance of 99.3% compared to 87.4% of VONG-M. For OBJ-SCT, VONG scores 80.9% compared to 74.7% of VONG-M.
- **Comparison of VONG-M and OGNG$^+$**: According to Table 6.1, VONG-M outperforms OGNG$^+$ for OBJ-SCT with 74.7% compared to 65.0% for OGNG$^+$. For the first dataset SPIRAL2, VONG-M delivers a slightly better performance with 87.4% compared to 86.0% for OGNG$^+$, which we do not consider as being significant.

We performed a t-test and could show, with a p-value of $p < 0.05$, that the performance of VONG is significantly better than those of VONG-M and OGNG$^+$ on both datasets .

6.6.3. Discussion

Our results show the advantage of the multi-view labeling strategy and the multi-view prediction strategy, as implemented in VONG. We can answer the question that it is indeed possible to construct a multi-view classifier that consists of a single uniform model. While the difference in performance of VONG and OGNG$^+$ is 13.3% for SPIRAL2, it is slightly higher for the OBJ-SCT dataset with a difference 15.9%. We assume this to be the effect of the more heterogenous character of OBJ-SCT compared to SPIRAL2, while being generated by more diverse domains. Comparing the results of VONG and VONG-M, we can see that VONG improves the performance of VONG-M with 11.9% for SPIRAL2, and with 6.2% for OBJ-SCT. In this case, the impact of the heterogeneity of OBJ-SCT is noticeable higher.

The results also show the benefit of the multi-view masking procedure when comparing VONG and VONG-M to OGNG$^+$, due to the fact that OGNG$^+$ neither performs a masking of feature space dimensions when labeling neurons, nor when prediction unseen data points. In fact, comparing VONG and VONG-M reflects the benefit of the multi-view labeling strategy, while a comparison of VONG-M and OGNG$^+$ quantifies the performance boost caused by the multi-view prediction strategy. Comparing the average results, we

can quantify the benefit of the multi-view labeling strategy with an increase in its performance of on average 9% and the benefit of the multi-view prediction strategy with with an increase in its performance of on average 5.6%. The higher impact of the multi-view labeling strategy on the performance of the classifier is understandable, as it is responsible for encoding process in which information coming from multiple domains are actually stored in the network. In contrast, the multi-view prediction strategy basically performs a decoding of the information stored in the network, which therefore only makes sense if the information is stored properly. Having mentioned that, it is interesting to see the effect of the prediction strategy as implemented by VONG and VONG-M. The multi-view prediction strategy has an undeniable positive effect on the performance of the classifier, although VONG-M and OGNG$^+$ basically learn an equal model. This fact underlines the plausibility of the VONG architecture and its masked multi-view labeling and multi-view prediction strategy even more.

6.7. Summary

In this chapter, we have introduced VONG as multi-view extension of OGNG, which is capable of learning with data coming from multiple domains and storing this information in a single uniform network. VONG modifies the labeling and prediction strategy of OGNG by incorporating a masking of feature space dimensions per view, in order to label and predict for each view independently. We have also introduced OGNG$^+$ as naive multi-label extension of OGNG, which simply adopts the complete set of subcategory labels as presented by the stimulus, and predicts a complete set of subcategory labels as given by the closest neuron in the network. OGNG$^+$ thereby extends the label capacity of each neuron, as for OGNG$^+$ neurons hold multiple labels (one per view) at once. We discussed the implicit multiplexing architecture of VONG including the process of encoding and decoding information implemented by the masked multi-view labeling and multi-view prediction strategies.

We have evaluated VONG and OGNG$^+$ on two artificial multi-view datasets and have shown that VONG significantly outperforms OGNG$^+$ on both of the datasets, and thereby proving the benefit of the proposed architecture. We furthermore compared VONG to VONG-M, a version of VONG that only implements the multi-view prediction of VONG, and adopts the naive multi-labeling strategy of OGNG$^+$. We could separately show and furthermore quantify the performance boost given by the multi-view prediction strategy and the multi-view labeling strategy. We concluded that it is possible to provide a uniform classification model which is capable of learning with multi-view data and that the of VONG implemented multi-view labeling and multi-view prediction strategies are reasonable.

In the next chapter, we will demonstrate the applicability of the GCM framework, *i.e.* OGNG and DYNG, in real stream data scenarios as a proof-of-concept. We will also introduce a straightforward approach to visualize GCM-based Conceptual Maps, which allow a real-time tracking of the included categories and a certain degree of interpretation possibilities of the data.

Chapter 7

Applications for Growing Conceptual Maps

In this chapter, we will demonstrate and highlight three important aspects of the Growing Conceptual Maps Framework (GCM): i) its *capability as stream data classifier* ii) its *flexibility* when integrated into a complex and domain-specific classification framework, and iii) its *simple architecture* which can be visualized straightforwardly. We will demonstrate those abilities by applying GCM onto two types of applications. Those applications are in particular the classification of *textual stream data* as arising in social media applications, and *Human Activity Classification*, with both applications involving real datasets. We will furthermore introduce a straightforward visualization schema in order to visualize the Conceptual Maps as generated by the GCM framework in a two-dimensional space. In many applications visualizations are important as they allow us as humans to track the internal state of the classification model, and furthermore indicate the relation between the different categories of the stream.

At first, we will apply our algorithms, DYNG in particular, to the classification of textual documents in a stream, using two corpora for evaluation: the Reuters RCV1 v2 corpus and the Twitter Corpus (TwitterLL). The datasets differ from each other in that Reuters RCV1 v2 contains 10 times more data points compared to TwitterLL, while it includes 14 times less categories. Our goal is to demonstrate the advantage of DYNG compared to OGNG, and also to compare the two approaches against standard classification approaches such as *k-Nearest Neighbor (kNN)* and a linear multi-class SVM. Thereby, we provide a proof-of-concept for the general applicability of OGNG and DYNG to various classification problems. Then we will introduce a novel online human activity classifier based on two OGNG (see Chapter 3) networks. We will compare this two-layered classifier to a state-of-the-art offline human activity classifier framework, as proposed by Laptev [Lap05], and show that our online classifier performs comparably without the requirement of storing any data points explicitly.

In the last section of the chapter, we will introduce a simple two-dimensional visualization of a GCM network and thereby highlight the strength of the uniform model as provided by GCM. The visualization is derived from the GCM network by interpreting its topology and allows to track emerging categories as well as their relation in the GCM network on-the-fly.

7.1. Classification of textual data in social media streams

In recent years, the structuring and classification of mass media has become more and more important, as the available data rapidly increases on a daily basis. Especially, information coming from social media platforms, such as Flickr[5], Facebook[6] or Twitter[7] is important to the society. Although concepts such as *hashtags* in Twitter allow the user to label the posted tweets, the data remains unstructured in the sense that only a small percentage of such labels are given. Similarly, for news feeds coming, for example, from *Reuters Group news agency*[8] only a weak supervision in form of labels that describe the general topic of the document can be given.

In this section, we thus evaluate DYNG as part of the GCM framework on two stream data classification text corpora. On the two datasets TwitterLL and Reuters RCV1 v2, we will demonstrate the advantage of DYNG as classifier, as the data can be assumed to be non-stationary to a certain degree. We will show that DYNG outperforms OGNG, especially in the beginning phase of the processing when many new categories are introduced. Furthermore, we will show that DYNG outperforms a kNN classifier with $k = 1$ and even a linear multi-class SVM, when allowing a similar memory capacity for all algorithms.

It should be mentioned that parts of this section have already been presented at the *European Symposium on Artificial Neural Networks* in 2013 [BC13].

7.1.1. The TwitterLL corpus

Since its inception in 2006, Twitter became one of the most important social media and micro-blogging services counting over 200 million active users in the year 2012, who posted on average 340 million tweets per day[9]. We created a Twitter corpus (TwitterLL) based on Twitter messages (tweets) by making use of the Twitter-API. We crawled 82.095 tweets

[5]Flickr, `https://www.flickr.com`
[6]Facebook, `https://www.facebook.com`
[7]Twitter, `https://www.twitter.com`
[8]Reuters Group news agency, `http://www.reuters.com`
[9]Twitter turns six, *Twitter Blog*, `https://blog.twitter.com/2012/twitter-turns-six`

assigned to 14.040 different hashtags in the time period between May 3rd 2012, and June 5th 2012. Therefore, we included only tweets that are tagged with the hashtag "Berlin" or that provided a Place-ID of Berlin, as provided by the Twitter-API. Our dataset thus includes 82.095 documents assigned to 14.040 category labels, as depicted in Table 7.1. The table compares the number of data points, the number of categories, the number of feature space dimensions, as well as the average number of data points per category for both stream datasets (TwitterLL and Reuters RCV1 v2).

	#Data points (DP)	#Categories (C)	#Feature space dim.	Avg. #DP per C
TwitterLL	82.095	14.040	2.623	5
Reuters RCV1 v2	804.414	103	832	7809

Table 7.1.: Statistics of the stream datasets TwitterLL and Reuters RCV1 v2.

As for the pre-processing, we generated a feature vector for each tweet based on the dot product of *term frequency (TF)* and *inverse document frequency (IDF)*, which is defined as follows:

$$tf.idf(t,d) = tf(t,d) \times log\frac{N}{df(t)}$$

Thereby, $tf(t,d)$ represents the frequency of the appearance of a term t in a document d, and $df(t)$ denotes the number of documents that contain t, with N being the total number of documents in the corpus. The number of documents N usually is unknown in a stream data setting, which would require to choose a different set of textual features. However, for our experiments we assumed N to be known, in order to extract the TF-IDF features. The feature vector for each document was then created by the TF-IDF value for each term in the corpus, and initially included 456.837 dimensions. In order to reduce the dimensionality of the vector, we removed terms that occur in less than 20% of the documents, and therefore could reduce the dimension of the TF-IDF feature vector from 456.837 to 2.623 dimensions. For every tweet we took the most common hashtag as category label.

7.1.2. The Reuters RCV1 v2 corpus

Reuters is the largest international text and television news agency worldwide, with thousands of stories that are produced in multiple languages daily. The Reuters RCV1 corpus [LYRL04] is a manually annotated text corpus derived from their online database. It includes exclusively english stories that have been produced by Reuters journalists between August 20th 1996 and August 19th 1997. For research purposes, the text documents are available and provided in a XML format. Its categories are grouped into Topics, Industries and Regions. For our experiments we only considered the Topics as category labels.

Those labels mainly consist of topics around economics, government and markets. In our experiments, we particularly use the Reuters RCV1 v2 corpus, which is a revisited version of the original Reuters RCV1 corpus. Reuters RCV1 v2 consists of 804.414 documents assigned to 103 different categories, as depicted in Table 7.1.

We use the TF-IDF feature vectors that have been provided by the authors of the dataset[10]. The feature vector initially included 47.236 dimensions, which we reduced to 832 by removing the terms appearing in less than 20% of the documents. For the category labels we chose the most popular topic for each document.

7.1.3. Experiments & Parameters

We evaluate DYNG in comparison to OGNG, and in comparison to the offline classifiers 1NN and SVM as baselines. DYNG and OGNG process a continuous stream of data while considering each data point only once. Every 1000th example, we let the algorithms perform a prediction for the next 1000 data points, for which we remove the category labels during the prediction. This allows us to test the accuracy on unseen data points that are next in the stream. While DYNG and OGNG are online learning algorithms, not requiring the explicit storage of data, a standard SVM learns in batch mode using a fix set of training examples. Also the 1NN classifier is basically "trained" by simply remembering all examples. In order to provide a comparable learning scenario, we let both offline classifiers store up to 5000 training examples of the stream in a FIFO principle, and then retrain after every 1000th examples in order to perform a prediction on the next 1000 examples. Preliminary experiments have shown that for 5000 training examples the number of stored vectors by the SVM (support vectors and training data) is comparable to the number of prototypes produced by DYNG.

For DYNG and OGNG, we empirically determine parameter settings on a trial-and-error basis. For the Reuters corpus, the DYNG/OGNG parameters are set as follows: insertion parameter $\lambda = 300$; maximum age $a_{max} = 100$; adaptation parameter for winner $e_b = 0.1$; adaptation parameter for neighborhood $e_n = 0.0006$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.0005$ and $\tau = 0.3$ (DYNG). We used the same settings for TwitterLL, except for $\lambda = 30$. For both DYNG and OGNG we select the relabel-method (see Section 3.4.1) as labeling strategy and single-linkage as prediction strategy. The SVM is trained in a one-vs-all mode and uses a linear kernel. The results of our experiments are shown in Figure 7.1. The two figures show the classification accuracy for four learning approaches (DYNG, OGNG, SVM, 1NN) compared to a majority baseline over the number of data points seen for TwitterLL and Reuters RCV1 v2.

[10] http://jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

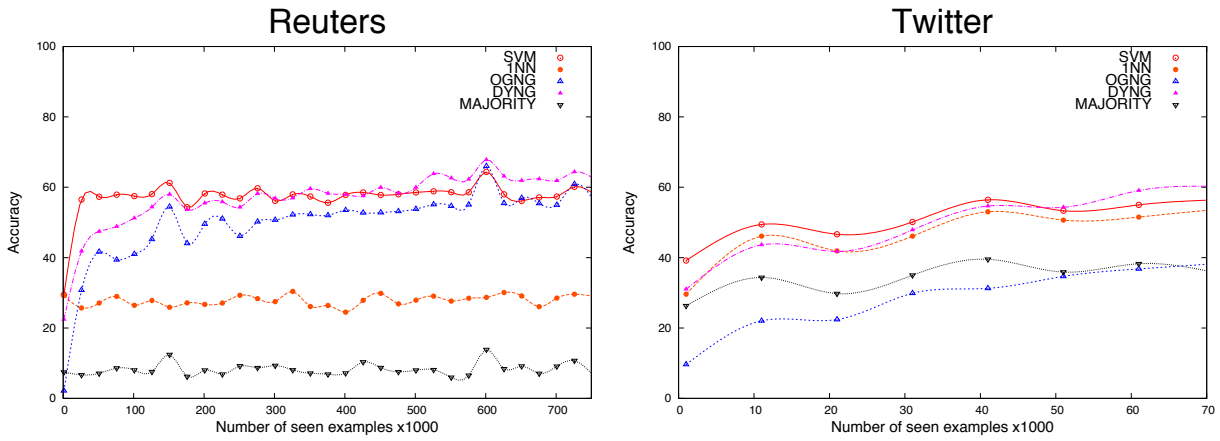


Figure 7.1.: Classification results of DYNG, OGNG, SVM and 1NN.

7.1.4. Results & Discussion

The results show that DYNG clearly outperforms OGNG on both datasets, especially in the heavily non-stationary beginning phase where many new classes are encountered. In this first phase DYNG improves the accuracy of OGNG by up to 20.28% (5.37% on average) on the Reuters data set and up to 21.4% (20.95% on average) for the Twitter data set. It is striking that for the Reuters data set DYNG starts (Iteration 0-25) with an accuracy similar to 1NN and outperforms the SVM starting from iteration 450 on with an improved accuracy of up to 5.53% (3.79% on average). Thereby, DYNG stores on average around 6368 vectors (neurons) compared to 5000 vectors (training examples) of 1NN and 9599 vectors (support vectors and training examples) of the SVM. This shows the benefit of a continuous learning process.

On the Twitter dataset, the DYNG, SVM and 1NN show similar results. DYNG outperforms the SVM starting from iteration 50 on with an improved accuracy of 4.04% (2.97% on average). Thereby, DYNG stores on average 9703 vectors compared to 9235 vectors of the SVM. The weak performance of OGNG shows the benefit of the neuron insertion strategies of DYNG, as OGNG is not designed to adapt to the huge amount of classes as quickly as DYNG.

It should be mentioned that we also included the HullerSVM [BB05] in preliminary experiments that have not been report. HullerSVM is basically an incremental online version of a Support Vector Machine. Therefore, HullerSVM dynamically identifies and stores training examples as support vectors. However, in order to extend HullerSVM to a multi-class classification problem, and to store support vectors for each category following the

principles of HullerSVM, has caused the memory requirement of HullerSVM to increase dramatically in such a way that it has not been applicable to our stream datasets.

7.2. Human activity classification

The recognition and classification of human activity is important in many application domains including *smart homes* [CCEF09], *surveillance systems* [VV05], *ambient intelligence* [PST07], etc. In particular, we address the task of classifying human activity into a given set of activity types on the basis of video data, sequences of 2D images in particular. State-of-the-art approaches extract features from space-time volumes, *e.g. Space-Time Interest Points (STIP)* as introduced by Ivan Laptev [Lap05]. Their advantage lies, in contrast to model-driven motion features [BGC12], in their compact and robust representation of human actions, as they reduce the input space by identifying local feature points. Training a human action classifier model based on STIPs typically includes the clustering (with *e.g.* k-Means) of video features to yield bag-of-visual-words clusters that can be used to derive a histogram-based representation of a video clip by indicating the number of STIPs being assigned to each cluster. Then a classifier (*e.g.* SVM) is trained to learn to classify image sequences into a set of given human activity types.

There are several drawbacks in this classical approach. First of all, the approach requires an architecture that comprises heterogeneous algorithms, *e.g.* a feature extractor, a clustering algorithm and a classification algorithm. An architecture that is more uniform and compact relying on one algorithm would be simpler, easier to implement and thus preferable. Further, the approach is not online, requiring to store a number of examples in memory or on disk in order to recompute the cluster and retrain the classifier at regular intervals. To circumvent these limitations, we present a new architecture which is based on Online Growing Neural Gas (OGNG), which has been presented earlier (see Chapter 3). In this section we present a two-layer architecture which consists of two maps that we call *feature-map* and *class-map*, respectively. In the first layer (feature-map), STIPs are dynamically clustered according to their similarity in the feature space, yielding a growing set of "visual words" that can also change over time. A *Human Activity Signature (HAS)* for each space-time volume is then formed by an histogram indicating the activity of each prototype / neuron in the feature-map. In the second layer (class-map), space-time volumes are clustered by their HAS and labelled according to the corresponding activity.

Several parts of this section already have been published and presented at the *Workshop on New Challenges in Neural Computation* [PBC13] in 2013.

We compare our architecture to the classical architecture proposed by Laptev [Lap05] based on a k-means based feature discretization as well as an SVM-based classification, showing

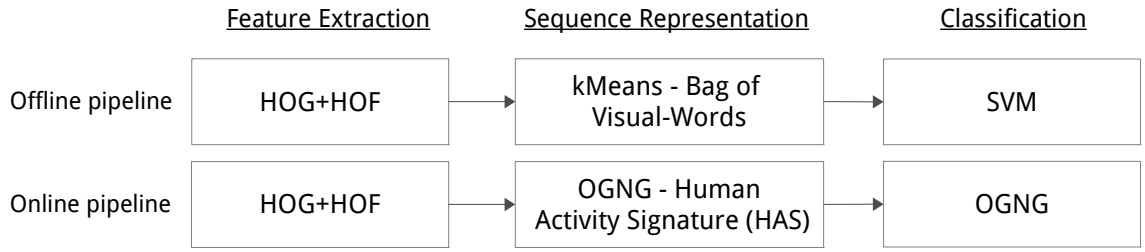


Figure 7.2.: Processing pipelines of the human activity classifiers.

that our approach yields comparable results to the latter approach, while proposing a uniform architecture based on two OGNG maps and circumventing the need to store examples to process them offline. Our approach is thus suitable in a life-long learning setting in which there is a never-ending data stream that needs to be processed on-the-fly as in the applications mentioned above.

7.2.1. Human Activity Classification with OGNG

In this section we describe our two-layer online approach to human activity classification that exploits topological maps - Growing Neural Gas maps in particular - at both layers. We call our approach *Online Human Action Classifier (OHAC)*. A typical processing pipeline in human activity recognition is comprised of the following three steps:

1. **Extraction of video features:** Extraction of distinctive features from a sequence of video frames.
2. **Representation of video features:** Calculating video signatures from the extracted features that capture the similarity between different video sequences.
3. **Classification of video signatures:** Training a classifier to learn to recognize a set of previously observed classes of human actions.

Our algorithm covers the steps 2 and 3 of the typical processing pipeline. For the first step, the extraction of video features, we use HOG+HOF features calculated around sparsely detected spatio-temporal interest points (STIPS) as proposed by Laptev in [Lap05]. The detection of the points and the extraction of the feature descriptors is completely online in the sense that no global information is required. STIPs are detected as local maxima of a Harris-Corner-Function extended into the spatio-temporal domain. The spatio-temporal Harris Corner function responds to points in space-time where the motion of local image structures is non-constant. Aside from sensor noise and other disruptions, the motion of local image structures is primarily the result of forces acting on the corresponding

physical objects. Hence the local neighborhood of the detected points can be expected to provide meaningful information about motion primitives in that point of space-time. The combination of STIPs and HOG+HOF feature descriptors has already shown promising results in several synthetic [Lap05] and real world datasets [Lap05, LMSR08].

Static Human Action Classifier according to Laptev

As baseline we use an offline approach proposed by Laptev [Lap05]. This approach uses k-means for clustering and an SVM for classification. The video sequence is represented as a *bag of visual words* histogram. The visual vocabulary is build by clustering the feature vectors in the training set into a predetermined number of clusters.

Algorithm 10 Static Human Action Classifier according to Laptev

1: Cluster the training set into a predetermined number of clusters using kMeans.
2: Initialize Histogram H with one entry for each prototype vector.
3: Present an input vector $x \in S$ from the extracted features of the video sequence.
4: Find the nearest prototype p_x to the presented input.
5: Increment the corresponding histogram entry p_x by one.
6: repeat step 3 until all features are processed.
7: Normalize H using L_1 norm
8: Train the SVM with the normalized histogram and the label l of the current sequence.

In step 1, the *visual vocabulary* is build in advance by clustering all feature vectors in the training set into a predetermined number of clusters[11] using kMeans. Each cluster stands with its prototype vector for one distinct *visual word* in the *visual vocabulary*. Steps 3-6 iterate over the feature vectors in the current training sequence. Each feature vector is assigned to its nearest prototype vector and incorporated into the histogram. The resulting histogram represents the given video sequence as a bag of visual words. To compensate for different counts of features in different sequences, the histogram is normalized using the L_1 norm. In step 8 the SVM is trained with the normalized histogram and the corresponding label of the current sequence.

Online Human Activity Classifier (OHAC)

The Online Human Activity Classifier (OHAC) consists of two independent OGNG networks. The first network is what we call the feature-map. It utilizes the OGNG algorithm for clustering incoming data in feature space. The second network is called the class-map as it uses the label information from a training sequence to assign class labels to the nodes in the network according to the relabel strategy presented in Section 3.4.1. Video

[11] We use $k = 4000$ for the number of clusters following Laptev in [LMSR08]

sequences are represented as Human Activity Signatures (HAS), which are represented by an histogram indicating the activity of each neuron in the feature-map while iterating through the respective video sequence.

Algorithm 11 Online Human Activity Classifier (OHAC)

1: Initialize the *feature-* and *class-maps*.
2: Initialize the *HAS* histogram H with two (number of initial nodes in the *feature-map*) entries $h_1 = 0, h_2 = 0$.
3: Present an input vector $x \in S$ from the extracted features of the video sequence.
4: Find the nearest unit n_x from the *feature-map*.
5: **if** node n_x is new **then**
6: insert new entry into the histogram at position x
7: **end if**
8: Increment the corresponding histogram entry h_x by one.
9: repeat step 3 until all features are processed.
10: Normalize H using L_2 norm.
11: Update the OGNG *class-map* with the normalized *HAS* histogram H and label l of the video sequence.

Algorithm 11 is initialized by first initializing the OGNG network and creating an empty bag of visual words histogram (steps 1-2). The histogram starts with two entries, one for each of the two initial neurons in the OGNG network. In steps 3-5 the next input vector x is presented to the feature-map and the node n_x that is closest to the presented stimulus is located. If the located node n_x is newly inserted into the feature-map, a new histogram entry is created at position x. In step 8 the histogram entry at position x is incremented. When all input vectors in the sequence are processed, the histogram is normalized by the L_2-norm. The normalization compensates for different numbers of extracted feature vectors in different video sequences. The normalized histogram is then used to update the class-map OGNG network with the label l of the video sequence. To predict the label of a previously unseen video sequence S, all input vectors $x \in S$ are incorporated into the histogram H by the number of the feature-map node n_x nearest to them (steps 3-5). The histogram is then normalized and the label l is predicted by the class-map according to thesingle linkage strategy (see Section 3.3.2).

7.2.2. KTH human motion dataset

As a dataset for the evaluation we use the KTH human action dataset [SLC04]. This video database consists of six categories of human actions (walking, jogging, running, boxing, hand waving sand hand clapping). All actions are performed several times by 25 subjects in four different scenarios (outdoors, outdoors with scale variations, outdoors with different clothes and indoors). In particular, our dataset includes 600 video files showing 25 subjects performing six actions in four scenarios.

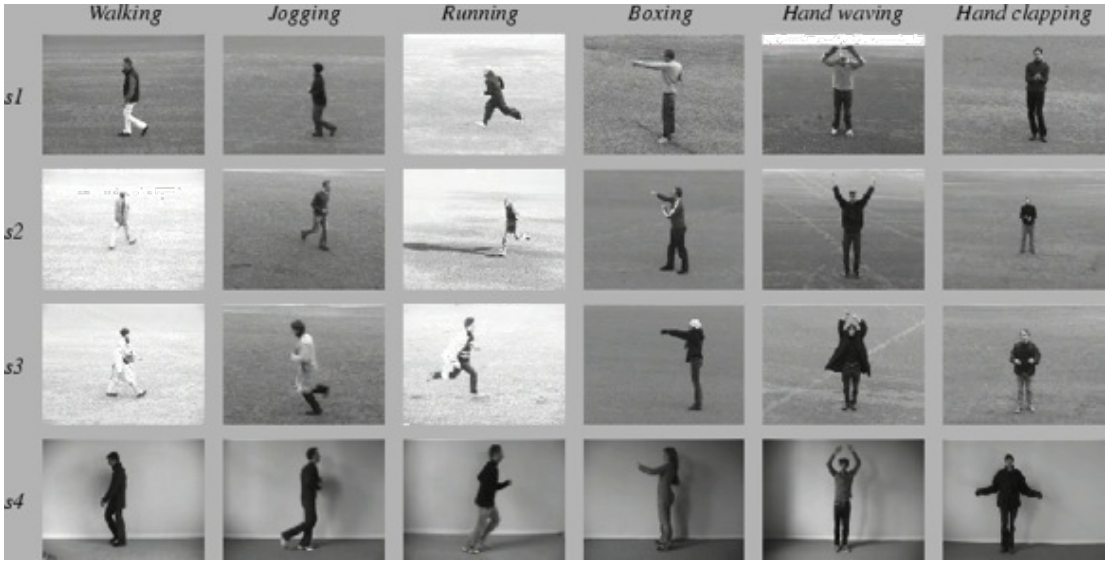


Figure 7.3.: Examples of the KTH-Dataset.[12]

7.2.3. Experiments & Results

We evaluated the accuracy of OHAC and our baseline on the KTH human action dataset. We generated 15 training and test sets by separating the 600 video clips into 300 training examples and 300 test examples for each set. We furthermore took care that each of the six categories was equally distributed in the training and test set. We averaged our accuracy results over the 15 runs and also determined a best and worst result of the 15 runs.

The OGNG parameters are set as follows: insertion parameter $\lambda = 50$; maximum age $a_{max} = 120$; adaptation parameter for winner $e_b = 0.3$; adaptation parameter for neighborhood $e_n = 0.0018$; error variable decrease $\alpha = 0.5$; error variable decrease $\beta = 0.0005$. We also allowed a maximum of 4000 neurons for the feature-map and 200 for the class-map.

Our results are depicted in Table 7.2. The matrices show the confusion matrix of our baseline (left) and OHAC (right). Thereby, each row represents the human action categories to be classified, while each column holds the percentage of examples that have been classified into the category written on top of the matrices. Overall, OHAC achieves an averaged accuracy of 93%, while our baseline yields an accuracy of 95%. We performed a t-test and could not prove that the results of both approaches are statistically significant. We thus consider the classification performance of the algorithms to be comparable. The confusion matrix shows that both algorithms are having issues to distinguish between jogging,

[12] http://www.nada.kth.se/cvap/actions/actions.gif

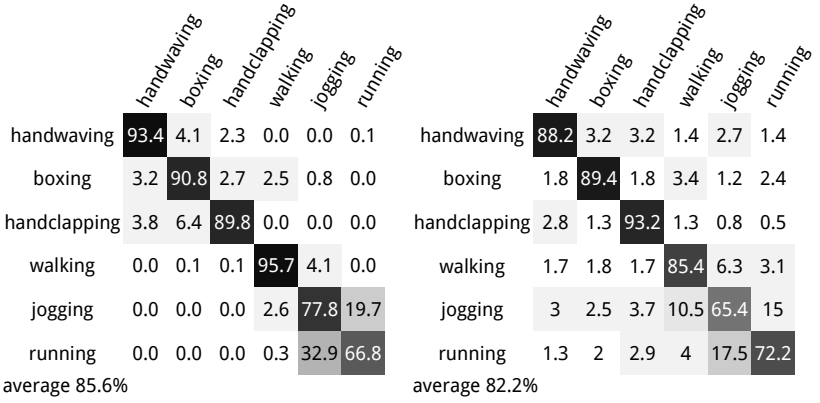


Table 7.2.: Confusion matrix of our Baseline (left) and OHAC (right) on the KTH human action database, averaged over 15 runs.

running, which is intuitively understandable as we as humans also would consider those two activities to be closer to each other compared to the other four. Furthermore, it is interesting that OHAC confuses the categories "handclapping" and "running" slightly less with 93.2% and 72.2% compared to 89.8% and 66.8% of our baseline. It should also be mentioned that the confusion of OHAC is spread more uniform compared to the baseline approach, which can be explained by the generative nature of OHAC, due to its OGNG layers, compared to the discriminative character of our baseline and *i.e.* of the underlying SVM classifier.

7.2.4. Discussion

In this section we have presented a novel human activity classifier model based on Online Growing Neural Gas (OGNG). The model provides a compact architecture and consists of two layers, allowing the storage of human actions in a more memory efficient structure. While the first layer (feature-map) dynamically clusters STIPs and serves as base for the creation of histogram-based signatures of a human action, the second layer (class-map) builds a classification model that builds upon those human action signatures. The advantage of this novel architecture lies in its ability to perform a human action classification task online as the model stepwise adapts to new data and grows incrementally. The uniform character of the algorithm is desirable, as that its simplicity allows an easy implementation and integration into existing systems. In most cases, heterogeneous offline human action recognition approaches have been proposed [Lap05, YS05], that generate action signatures by clustering STIPs with static clustering algorithms (such as k-Means) and classifying them with an offline classifier that needs to be retrained as new data arrives. We have

experimentally shown on the KTH dataset that our approach reaches comparable performance to a classical offline SVM-based classification approach while performing online and avoiding the need to store training examples explicitly, thus being suitable in lifelong stream data settings.

7.3. Visualization of Conceptual Maps

In this section, we will introduce a visualization schema for Growing Conceptual Maps which basically interprets the GCM network and generates a two-dimensional map utilizing those derived information. In what follows, we will explain how to generate a two-dimensional visualization of a Conceptual Map as derived from the GCM network. We will also describe how alternative visualizations of GCM-related networks, *i.e.* the class-map of OHAC, can be generated. In addition, a visualization of the three datasets of this chapter, as well as a visualization of the class-map of OHAC are shown in figures A.1-A.6 of Appendix A.

7.3.1. A visualization schema for Growing Conceptual Maps

Although a direct visualization of a GCM network is possible, in many domains it can be problematic to project the high-dimensional feature space into a low-dimensional feature space. Preliminary visualizations of the TwitterLL dataset, for example, have shown that a projection from the 2.623 dimensional feature space into a two-dimensional space using a Principal Component Analysis (PCA) [Pea01] does not result in a map which can easily be understood, in the sense that the categories are not separable in such low-dimensional space. Instead, we decided to interpret the characteristics of the map and to visualize the interpretation of the Conceptual Map. In fact, we provide a visualization of the Conceptual Map only including the stored categories, instead of the neurons itself. However, in order to be truthful to the topology of the our network and its included connected and relating categories, we also derived topological information from the GCM network and incorporated it into the Conceptual Map visualization. In particular, the visualization is based on the following elements:

- **Nodes**: Nodes m are introduced in our visualization map $\mathcal{V}$ according to the currently manifested categories in the GCM network $\mathcal{M}$.
- **Size of a node**: The size of a node m_i reflects the logarithmic number of neurons in $\mathcal{M}$, which are labeled with the category label of m_i.

- **Links**: Links $link_k(c_i, c_j)$ are introduced between categories c_i and c_j, in case there are neurons n_i, n_j in $\mathcal{M}$, with $l^t(n_i) = c_i$ and $l^t(n_j) = c_j$, and n_i and n_j being connected by an edge.
- **Size of a link**: Similar to the size of a neuron, the size of a link $link_k(c_i, c_j)$ between the categories c_i and c_j in $\mathcal{V}$ is determined by the logarithmic number of links which exist in $\mathcal{M}$ between neurons labeled with c_i and c_j, respectively.
- **Dynamics of the map**: Inspired by Provot [Pro95], the dynamics of the network follow the principle of a simple mass-spring model, in which attracting and repellent forces exist between every pair of nodes (repellent forces) and between every connected pair of nodes (attracting forces).
- **Coloring schema**: In order to improve the readability of $\mathcal{V}$ further, we also let colors reflect the relation between categories. A unique color is initially assigned to a newly inserted category node. We use the RBG color schema and set the color of a link $link_k(c_i, c_j)$ between two categories c_i and c_j to a resulting mixture color between both of their colors, weighted by their presence in the network. In particular, the color is set as follows: $color(link(c_i, c_j)) = \alpha_{c_i} color(c_i) + \alpha_{c_j} color(c_j)$, with $\alpha_{c_k} = \frac{\#\text{Neurons labeled with } c_k}{max(\#\text{Neurons per category})}$. The growth of a link furthermore reflects the logarithmic number of links between neurons of both categories c_i and c_j in the GCM network.

As a consequence, this means that connected categories in $\mathcal{V}$ relate to each other in $\mathcal{M}$. However, if categories are not connected in $\mathcal{V}$ we have no information about their relation. Even if categories define a neighborhood in $\mathcal{V}$, they cannot be assumed to be neighbors in $\mathcal{M}$.

7.3.2. A generic visualizations concept

The visualization schema that we proposed in this section is not limited to a visualization of categories of a GCM network. In fact, we also applied this schema to the class-map of OHAC while processing the KTH motion dataset. The visual class-map only distinguishes from a visual Conceptual Map in that each node of the map represents a neuron of the class-map, instead of its category only. The visual class-map can be seen as a more detailed visual Conceptual Map, as each representative (each neuron) for each category is visualized.

In general, it is valid to claim that this visualization schema can be applied to any topological feature map, and even to any other prototype-based clustering approach which includes category label information.

7.3.3. Examples of visualized Conceptual Maps

We visualized $\mathcal{V}$ during the processing of TwitterLL, Reuters RCV1 v2 and KTH through $\mathcal{M}$, and generated a video clip for each map in order to tracking the stream datasets and motion dataset, respectively, in an online fashion. Some snapshots of these video clips are shown in Appendix A. In particular, in Figure A.2-A.5 there are four images of TwitterLL and four images of Reuters RCV1 v2, showing snapshots that have been taken at different points in time. In Figure A.1, we also visualized an image of the KTH dataset, showing the six human activity categories. It should be mentioned that these visualization do not show all categories of $\mathcal{M}$, instead they only show the most popular categories at the moment in time when the snapshot was taken. In the following, we will briefly highlight some of our observations according to these images.

Visualization of TwitterLL

Most popular categories in the TwitterLL visualization:

- *Berlin* - "berlin": The city of Berlin.
- *re:publica 2012* - "rp12": A german conference on the topics of Web 2.0 and social media.
- *Piraten Partei* - "piraten": A german political party who are known for their online presence.
- *Echoverleihung 2012* - "echo2012": A german music gala.
- *European Poker Tour Season 2012* - "ept2012"
- *Earth hour 2012* - "earthhour": An international event organized by the *World Wide Fund Nature (WWF)* which took place on March 31th 2012.
- *SoundCloud* - "soundcloud": An online music distribution platform based in Berlin.
- *Facebook* - "fb": One of the most popular online social networking service.

The most obvious observation is that the category "berlin" builds the center of the Conceptual Map, which is not surprising as the dataset was crawled according to tweets including this hashtag. Also the category "piraten" is a stable and noticeable category. Besides categories of general nature, such as "news", "fb", "soundcloud", "jobs", or "travel", there are categories that relate to events which happened during that specific time period. Those categories are "rp12", "echo2012" and "ept2012". It is interesting that for the ladder two categories there are connections between "echo" and "soundcloud", and between "ept2012" and "poker". The visualization even shows categories which have been of interest for a very

short time. The category "earthhour" only appears in $\mathcal{V}$ video clip for a very short period in time and is depicted in Figure A.3 (top), after $\mathcal{M}$ having processed 85% of the data.

Visualization of Reuters RCV1 v2

In contrast to TwitterLL, the categories of Reuters RCV1 v2 seem more stable as shown in the visual Conceptual Map in Figure A.4 and Figure A.5. This is also indicated by the size of the categories depicted in Figure A.5 (bottom), when comparing to the visual Conceptual Map of TwitterLL. The included categories are mostly coming from the finance market field, which is understandable as this is one of the main areas of the Reuters news agency. The most prominent categories are: "accounts", "investment research", "full-year results", "sales and markets" and "stock exchanges".

Visualization of KTH

The KTH motion dataset is visualized in Figure A.1. In contrast to the other two visualizations, only one image is provided as the dataset only consists of six categories, and we additionally visualized the class-map in order to inspect the conceptualizations of $\mathcal{M}$ in more detail. However, the visual Conceptual Map is most interesting as it shows the six human activities that have been classified by $\mathcal{M}$. The first observation which can be drawn is that we can identify two groups of categories, actions performed with hands ("boxing", "handclapping", "handwaving"), and actions performed with feet ("jogging", "walking", "running"). A second observations manifests in the fact that the connections between actions performed with feet are much closer than those performed with hands. As this is also reflected by the links between the nodes of both groups, the actions performed with feet can be assumed to are more similar (according to our feature space), than the actions performed with hands.

Visualization of the KTH OHAC class-map

Figure A.6 shows the class-map of OHAC after $\mathcal{M}$ having processed 100% of the KTH motion data. The advantage of this visualization, compared to the previous visualization of the KTH motion dataset, lies in its granularity. The class-map visualization also indicates impurities of the clusters within $\mathcal{M}$. During the processing we observed the two emerging groups of hand actions and foot actions. However, Figure A.6 also reveals that there is a high degree of confusion between the categories "jogging", "walking", and "running".

7.3.4. Discussion

The visualizations show that the visual Conceptual Map $\mathcal{V}$ allows us to track the evolution of the GCM-based classifier on-the-fly, seeing new categories emerging while others disappear. In the TwitterLL visualization, we could observe categories which exclusively have been relevant in that period of time. Besides a number of more generic categories, the visual Conceptual Map informed us about current trends in Twitter and furthermore also indicated how they relate to each other. For the Reuters RCV1 v2 dataset, we perceived a more generic and static category distribution, which is plausible due to the smaller number of categories involved with 103 categories compared to 14.040 categories for TwitterLL.

For the KTH motion dataset, we could observe two groups of action categories, hand-based actions and feet-based actions. This shows that $\mathcal{M}$ successfully represents the similarities between the action categories, as we also observed these relations in our experiments in Section 7.2. Furthermore we can confirm the high degree of confusion between actions performed by foot according to Table 7.2.

7.4. Summary

In this chapter, we applied DYNG and OGNG on two real stream datasets as proof-of-concept. We could show that DYNG outperforms OGNG on both datasets, especially in the beginning phase of the processing, in which many new categories are introduced to the classification model. We could also show that DYNG outperforms a 1NN classifier and a linear multi-class SVM, after a certain number of iterations and when allowing a comparable amount of memory. We could thus demonstrate the advantage of DYNG and OGNG in a real stream data scenario. As a second contribution, we introduced a novel online human activity classifier based on two OGNG networks, in order to dynamically generate feature vectors and to classify those according to the presented action. We compared our online approach to a state-of-the-art (offline) human activity classifier and have shown that our online version does not significantly deteriorate the classification performance with the benefit of no additional memory to be required.

Chapter 8

Conclusion

In this thesis, we introduced Growing Conceptual Maps as a novel classification framework, which is based on topological feature maps. The framework provides solutions for the life-long learning challenges involving stream data classification, weak supervision, non-stationary data and multi-view data. The framework has been implemented on the basis of Growing Neural Gas, which has been successively extended towards meeting these challenges. We furthermore applied the GCM framework onto two real stream datasets as proof-of-concept, and introduced a novel online human classifier based on the framework. As a last contribution we introduced two straight forward visualization of the GCM network, in order to allow a two-dimensional representation of the in the network included categories and their relations.

We introduced Online Growing Neural Gas (OGNG) as extension of GNG, which is capable of classifying data on the fly while processing data in one-pass, performing online updates of the model and allowing a fast prediction of unseen data. OGNG circumvents the need of storing any data explicitly and provides a uniform architecture that includes a time complexity in average and worst case of $\mathcal{O}(log_2 n)$ and a space complexity in average and worst case of $\mathcal{O}(n)$. In order to extend GNG to an online classifier, we introduced novel online labeling strategies (relabel, freq, limit) and compared those to standard offline labeling strategies (min-dist, avg-dist, majority), and furthermore experimentally showed that the online labeling strategies do not deteriorate the classification performance of OGNG, with the benefit of processing data online. Also the implicit two level architecture of OGNG including its feature and category level have been discussed, allowing OGNG to still provide a clustering when trained without labels. We also could show that OGNG can compete with a linear multi-class SVM, while outperforming the SVM on two of our datasets. Furthermore, as a last contribution in terms of OGNG, we have experimentally proven that letting the clustering be guided by the category distribution significantly deteriorates the classification performance.

We introduced Online Semi-supervised Growing Neural Gas (OSSGNG) as semi-supervised extension of OGNG that allows the utilization of weakly labeled data in a classification task. Although its modifications are minor compared to OGNG, it significantly outperforms OGNG when applied to appropriate semi-supervised data. In particular, OSSGNG introduces an additional prediction strategy for unlabeled data that predicts labels for those data points and then utilize this data for the classification task. We could show that OSSGNG outperforms existing semi-supervised GNG extensions, *i.e.* SSGNG, which train in a batch mode inspired by the Expectation-Maximization Algorithm. Furthermore, our results show that OSSGNG is competitive towards standard semi-supervised classification algorithms, while retaining the advantages of OGNG of which lie in the complexity of OGNG. Our experiments have shown the benefit of the labeling dynamics of OSSGNG that allow the model to quickly adapt to categories, for which labels are given in delay. Thereby, we demonstrated that OSSGNG can benefit from already learned unsupervised categories, while improving its classification performance rapidly.

As a third contribution to the GCM framework, we introduced Dynamic Online Growing Neural Gas (DYNG) as extension of OGNG with an emphasis on non-stationary data and concept-drift. Existing non-stationary extensions of GNG, *i.e.* Growing Neural Gas with Utility, are not equipped with mechanisms to classify data, and furthermore operate in a destructive way by substituting the neuron removal strategy of GNG. DYNG, instead, introduces two novel neuron insertion strategies that allow the algorithm to quickly adapt to newly emerging categories, until its classification performance converges. Therefore, DYNG tracks the classification performance of each known category independently and inserts neurons, in order to quickly provide a reasonable representation of each category. The implicit short-term memory and long-term memory of DYNG has been discussed, which manifests in its neuron insertion and removal strategies. In our evaluation, we compared DYNG to an online classifier version of GNG-U (OGNG-U), showing that DYNG clearly outperforms OGNG-U and OGNG, and thus solves the plasticity-stability problem in a better way. We also compared DYNG to a linear multi-class SVM showing that it achieves a comparable classification accuracy, while providing a higher performance stability. Our experiments have also shown that the memory usage of DYNG is slightly higher compared to those of OGNG, which can be considered as being minor due to DYNG sharing the time and space complexity of OGNG.

With Multi-view Online Growing Neural Gas (VONG), we introduced a multi-view extension of Online Growing Neural Gas that modifies the OGNG labeling and prediction strategy in order to process and store data coming from multiple domains in a single network. These domains generate views on the given data for which the feature vector is composed by dimensions that are uniquely relevant to each of domains. VONG has proven to be capable of processing such data under the assumption of those relevant dimensions are given in the processing phase. The novel multi-view labeling and multi-view prediction strategy, as implemented by VONG, preform a feature space masking per view, in order

to assign and predict labels in each of the views' feature subspaces. As those masking is only applied to VONGs' category level, it still provides a clustering equal to OGNG. We evaluated VONG on two artificial multi-view datasets showing that VONG outperforms a naive multi-label classification approach based on OGNG. We also investigated and quantified the impact of both multi-view strategies (labeling and prediction) on the overall performance. Our results show that the multi-view labeling as well as the multi-view prediction strategy significantly influence the classification accuracy, which proves our proposed architecture of VONG to be reasonable. Although VONG and OGNG share the same complexity in space, VONG increases the time complexity by the factor k, with k being the number of views. However, due to its equal space and time complexity in each view, compared to OGNG, it still should be considered as being applicable to real-time applications.

As last contribution in this thesis, we applied the GCM framework to three real time applications coming form the domain of textual stream data and human activity classification. For the stream data classification scenario, we applied DYNG and OGNG on two corpora (TwitterLL and Reuters RCV1 v2) and could prove their capabilities as stream data classifiers, by showing that both achieve reasonable results while DYNG outperforms OGNG in phases in which a lot of new categories emerge. We could also show that DYNG even outperforms a 1NN classifier and a linear multi-class SVM when allowing a comparable amount of memory for all compared algorithms. In the human activity classification scenario, we have built an online human activity classifier based on two OGNG networks that allow a dynamic online feature vector generation, as well as an online classification of the presented human actions. We compared the Online Human Activity Classifier (OHAC) with a state-of-the-art (offline) human activity classifier and have shown that OHAC does not deteriorate the classification performance significantly, while providing an on-the-fly classification. We also introduced a visualization schema for topological feature maps and could show that the visualization of Conceptual Maps allow the underlying GCM network to be interpreted to some degree, while providing information about relevant categories and their relation.

Outlook

There are possibilities to improve the Growing Conceptual Maps Framework, as well as finding a larger variety of applications to which GCM renders itself ideally applicable. A shortcoming of the GCM framework lies in the assumption of VONG that the mapping between domains and their relevant dimensions is given. However, learning a dimension weighting is a research field on its own and out of the scope of this thesis. Approaches from the area of *relevance learning*, such as *Generalized Relevance Learning Vector Quantization (GRLVQ)* [HV02] and *Incremental GRLVQ* [KLR08] address this challenge and have

proven to be successful. In fact, in preliminary experiments we tried to solve this problem for the OBJ-SCT without success. We applied several approaches including a brute-force method, an Evolutionary Algorithm, as well as techniques coming from relevance learning in order to identify the relevant dimensions for each domain. Although in most real scenarios those dimensions can be assumed to be known, as the feature vector is composed by known feature subspaces coming from those domains, it would extend the flexibility of VONG and allow it to be applied in more multi-view data applications.

In terms of applications, it would be interesting to apply VONG in a multi-modal classification setting in which each domain is derived from a different modality, such as audio, video, text and natural language. Thereby, VONG could contribute to solving the symbol grounding problem in that its visual Conceptual Map could indicate existing relations between such multi-modal concepts. Inspired by OHAC, it also would be interesting to apply the GCM framework in application, in which its properties of clustering and classification could equally contribute to a more complex framework.

Appendix A

Conceptual Map Visualizations

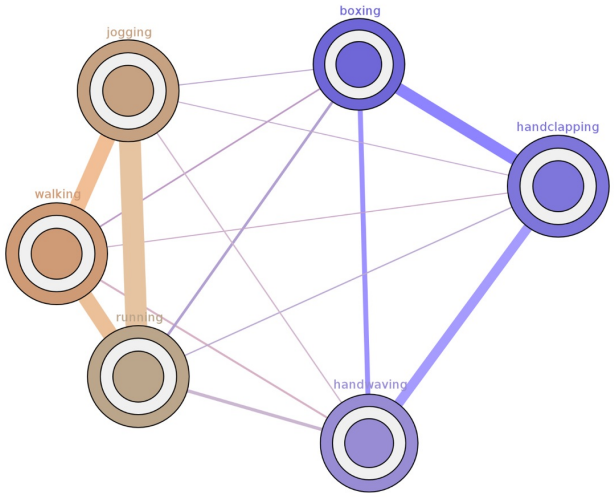


Figure A.1.: Visualization of the KTH-based Conceptual Map after having processed the dataset.

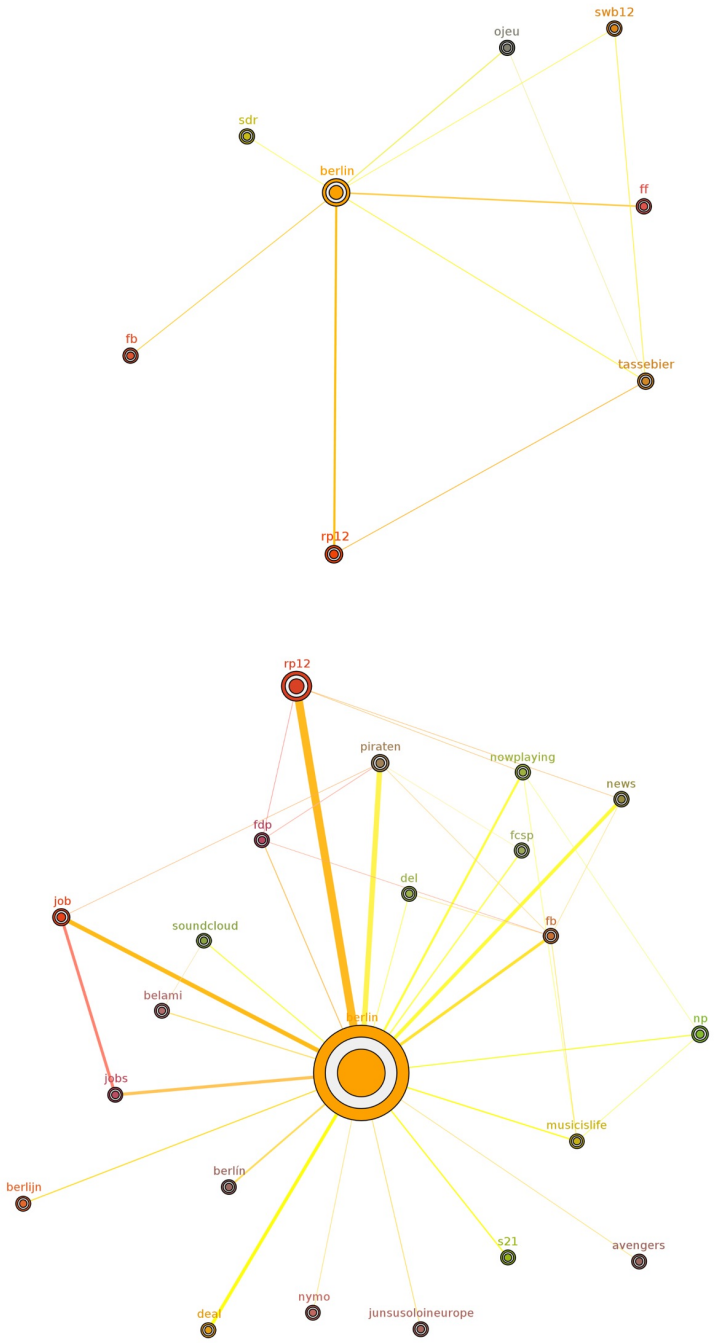


Figure A.2.: Visualization of the TwitterLL-based Conceptual Map after having processed 2% (top) and 20% (bottom) of the data.

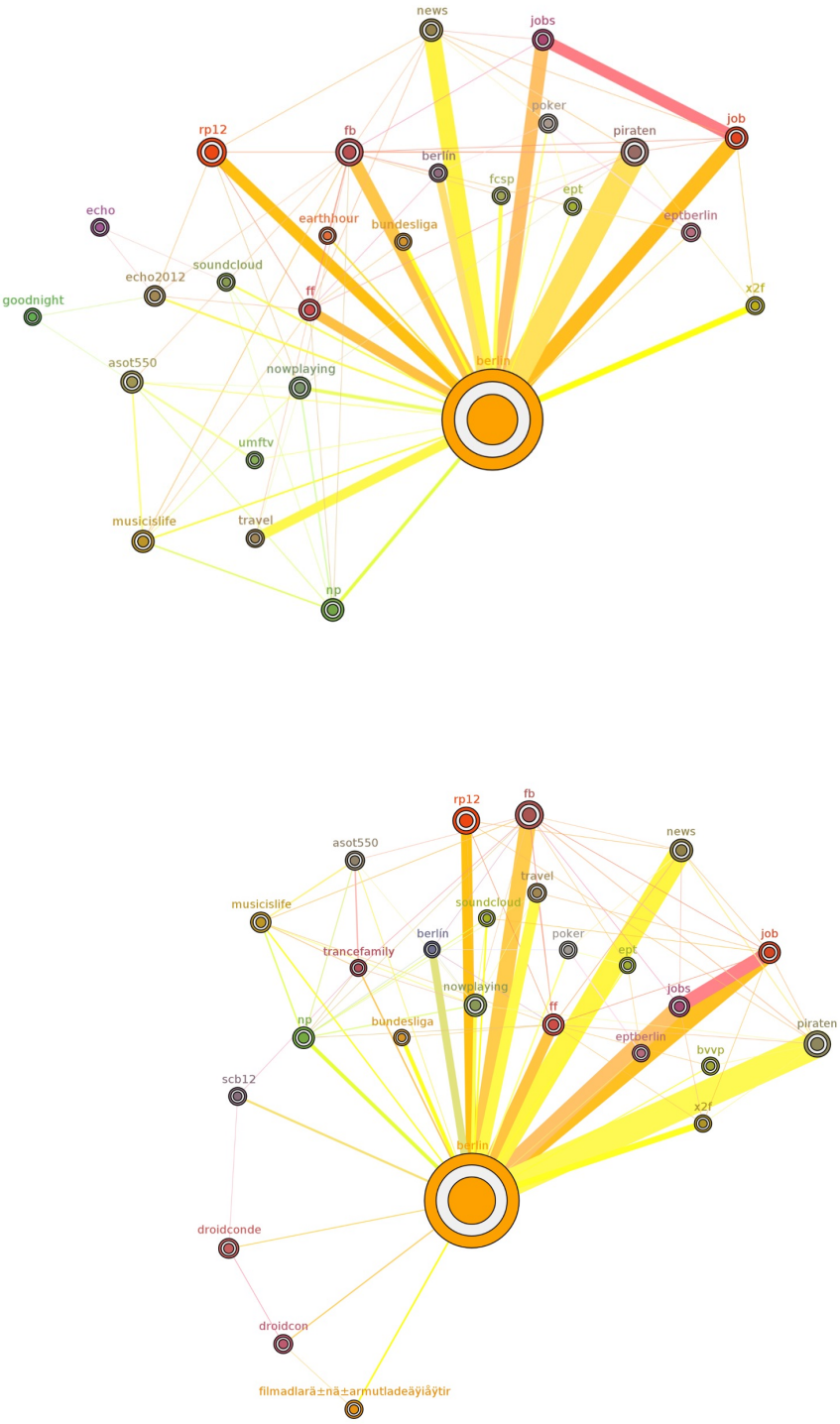


Figure A.3.: Visualization of the TwitterLL-based Conceptual Map after having processed 85% (top) and 100% (bottom) of the data.

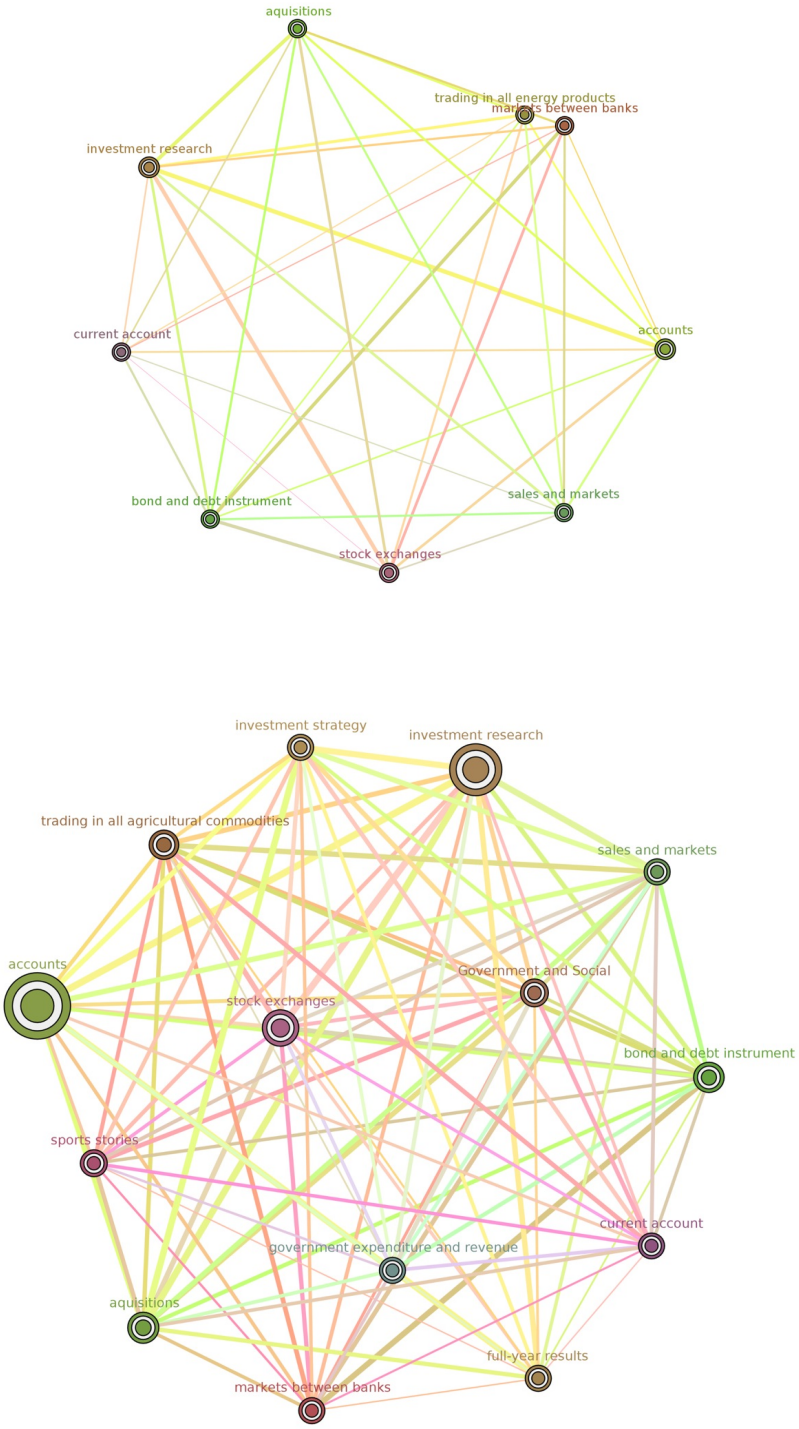


Figure A.4.: Visualization of the ReutersRCV1v2–based Conceptual Map after having processed 5% (top) and 50% (bottom) of the data.

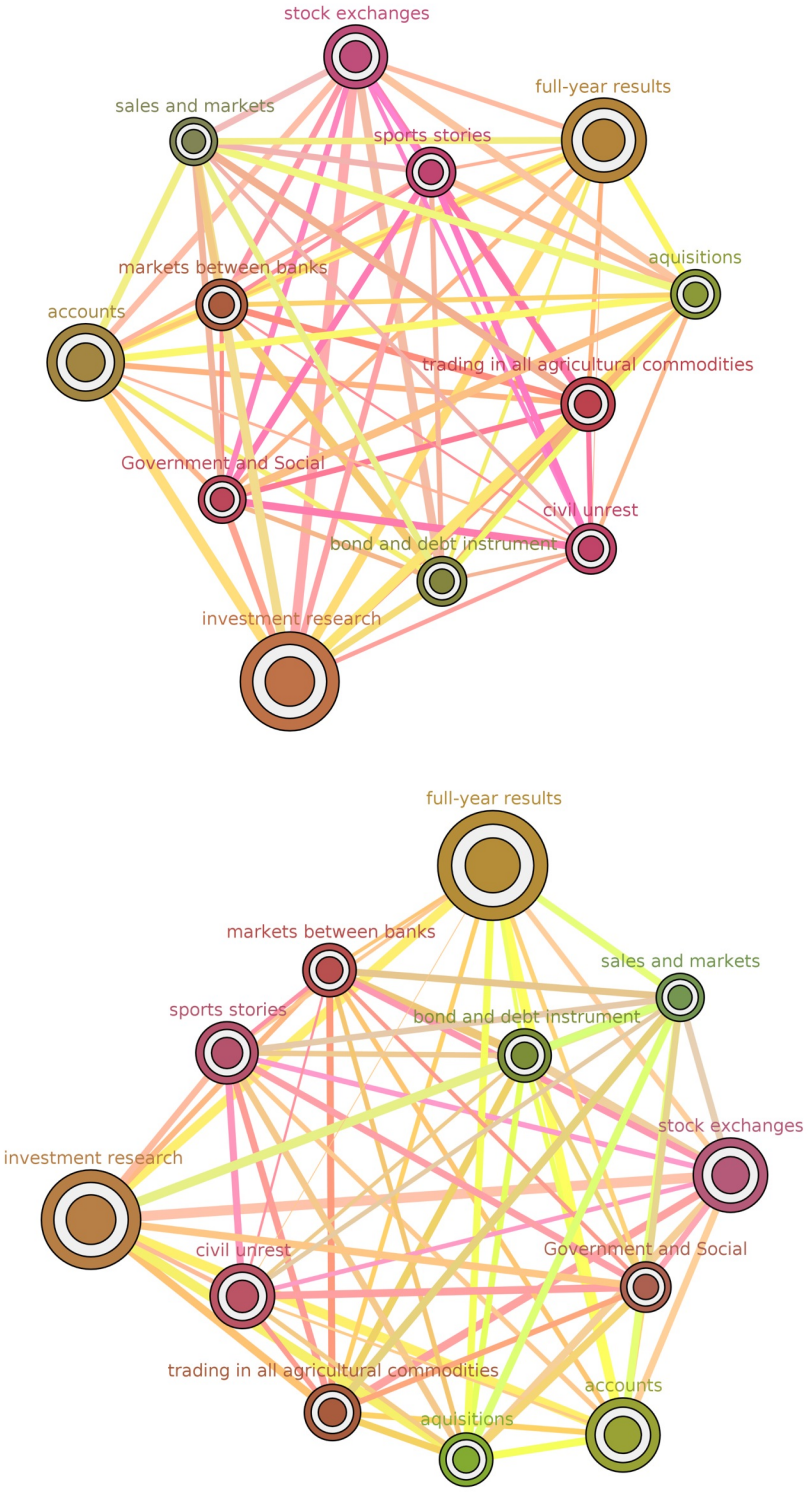


Figure A.5.: Visualization of the ReutersRCV1v2-based Conceptual Map after having processed 88% (top) and 100% (bottom) of the data.

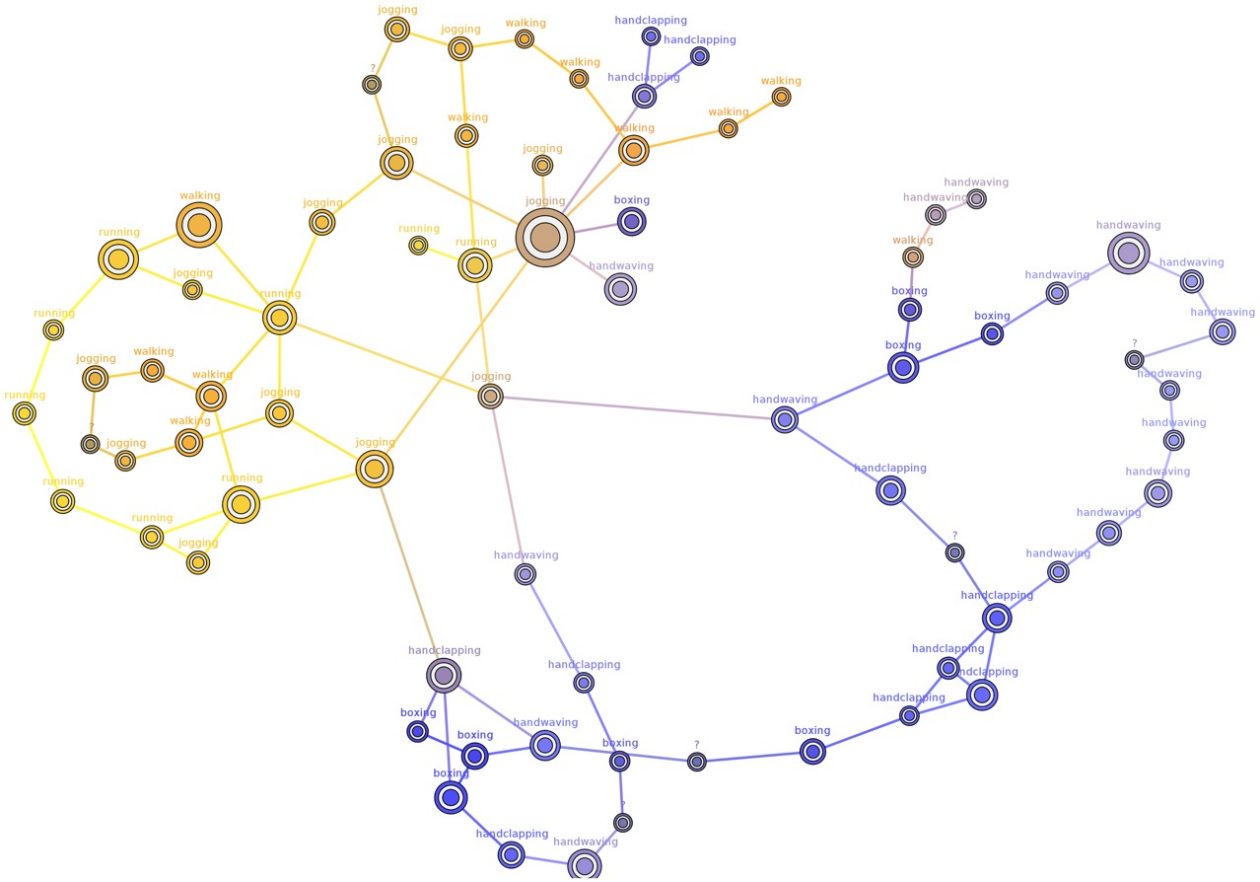


Figure A.6.: Visualization of the KTH class map after having processed the dataset.

Appendix B

Bibliography

[ABM92] Ahn, Woo-kyoung ; Brewer, William F. ; Mooney, Raymond J.: Schema acquisition from a single example. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 18 (1992), Nr. 2, S. 391–412

[AHWY04] Aggarwal, Charu C. ; Han, Jiawei ; Wang, Jianyong ; Yu, Philip S.: On demand classification of data streams. (2004), S. 503–508

[And73] Anderberg, M.R.: *Cluster analysis for applications*. Academic Press, New York, 1973

[Bag00] Bagnall, Richard G.: Lifelong learning and the limitations of economic determinism. In: *International Journal of Lifelong Education* 19 (2000), Nr. 1, S. 20–35

[BB05] Bordes, Antoine ; Bottou, Léon: The Huller: a simple and efficient online SVM. In: *Proceedings of the European Conference on Machine Learning* (2005), S. 505–512

[BBD00] Bradley, PS ; Bennett, KP ; Demiriz, Ayhan: Constrained k-means clustering. In: *Microsoft Research, Redmond* (2000), S. 1–8

[BC11a] Beyer, O. ; Cimiano, P.: Online labelling strategies for growing neural gas. In: *Proceedings of the 12th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)* (2011), S. 76–83

[BC11b] Beyer, O. ; Cimiano, P.: Online semi-supervised growing neural gas. In: *Workshop New Challenges in Neural Computation (NC2)* (2011), 30.08.2011, S. 21–23

[BC12] Beyer, Oliver ; Cimiano, Philipp: Online Semi-supervised Growing Neural Gas. In: *International Journal of Neural Systems* 22 (2012)

[BC13] Beyer, Oliver ; Cimiano, Philipp: DYNG: Dynamic Online Growing Neural Gas for Stream Data. In: *European Symposium on Artificial Neural Networks* (2013), S. 497–502

[BDFS10] Bellas, Francisco ; Duro, Richard J. ; Faiña, Andrés ; Souto, Daniel: Multilevel darwinist brain (mdb): Artificial evolution in a cognitive architecture for real robots. In: *Autonomous Mental Development, IEEE Transactions on* 2 (2010), Nr. 4, S. 340–354

[BDH03] Barroso, Luiz A. ; Dean, Jeffrey ; Holzle, Urs: Web search for a planet: The Google cluster architecture. In: *Micro, Ieee* 23 (2003), Nr. 2, S. 22–28

[Bel56] Bellman, Richard: Dynamic programming and Lagrange multipliers. In: *Proceedings of the National Academy of Sciences of the United States of America* 42 (1956), Nr. 10, S. 767

[Bel04] Belkin, I. & Niyogi P. M. & Matveeva M. M. & Matveeva: Regularization and semi-supervised learning on large graphs. In: *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory (COLT)* (2004), S. 624–638

[BF08] Barbakh, W. ; Fyfe, C.: Online Clustering Algorithms. In: *International Journal of Neural Systems* 18 (2008), Nr. 3, S. 185–194

[BGC12] Beyer, Oliver ; Griffiths, Sascha ; Cimiano, Philipp: Towards action representation within the framework of conceptual spaces: Preliminary results. In: *The 8th International Cognitive Robotics Workshop (CogRob)* (2012), S. 8–15

[Bla98] Blake, Merz C. C.L.: UCI repository of machine learning databases. (1998)

[Blu98] Blum, T. A.& M. A.& Mitchell: Combining Labeled and Unlabeled Data with Co-Training. In: *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)* (1998), S. 92–100

[BM98] Blum, Avrim ; Mitchell, Tom: Combining labeled and unlabeled data with co-training. In: *Proceedings of the eleventh annual conference on Computational learning theory* (1998), S. 92–100

[BRC+07] Bettencourt, Luís MA ; Ribeiro, Ruy M. ; Chowell, Gerardo ; Lant, Timothy ; Castillo-Chavez, Carlos: Towards real time epidemiology: data assimilation, modeling and anomaly detection of health surveillance data streams. In: *Intelligence and Security Informatics: Biosurveillance*. Springer, 2007, S. 79–90

[CA06] Corduneanu A, Jaakkola T.: Data dependent regularization. In: *Semi-supervised Learning* (2006), S. 163–182

[CBV11] Cruz-Barbosa, R. ; Vellido, A.: Semi-supervised Analysis of Human Brain Tumour from Partially Labeled MRS Information using Manifold Learning Models. In: *International Journal of Neural Systems* 21 (2011), Nr. 1, S. 17–29

[CCEF09] Chan, Marie ; Campo, Eric ; Estève, Daniel ; Fourniols, Jean-Yves: Smart homes—current features and future perspectives. In: *Maturitas* 64 (2009), Nr. 2, S. 90–97

[CD97] Chaudhuri, Surajit ; Dayal, Umeshwar: An overview of data warehousing and OLAP technology. In: *ACM Sigmod record* 26 (1997), Nr. 1, S. 65–74

[CEQZ06] Cao, Feng ; Ester, Martin ; Qian, Weining ; Zhou, Aoying: Density-based clustering over an evolving data stream with noise. In: *Proceedings of the 2006 SIAM International Conference on Data Mining* (2006), S. 328–339

[Cha06] Chapelle, B. & Zien A. O. & Schölkopf S. O. & Schölkopf: *Semi-supervised Learning*. MITPress, 2006

[CKMS03] Cormode, Graham ; Korn, Flip ; Muthukrishnan, S ; Srivastava, Divesh: Finding hierarchical heavy hitters in data streams. In: *Proceedings of the 29th international conference on Very large data bases-Volume 29* (2003), S. 464–475

[CKR05] Cook, Richard S. ; Kay, Paul ; Regier, Terry: The world color survey database: History and use. In: *Handbook of Categorisation in the Cognitive Sciences. Amsterdam and London: Elsevier* (2005)

[CL11] Chang, Chih-Chung ; Lin, Chih-Jen: LIBSVM: A library for support vector machines. In: *ACM Transactions on Intelligent Systems and Technology* 2 (2011), S. 27:1–27:27. – Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[CO05] Chapelle O, Zien A.: Semi-supervised classification by low density separation. In: *Proceedings of the 10th Int. Workshop on Artificial Intelligence and Statistics* (2005), S. 57–64

[CUD12] Christoudias, C ; Urtasun, Raquel ; Darrell, Trevor: Multi-view learning in the presence of view disagreement. In: *arXiv preprint arXiv:1206.3242* (2012)

[DAD05] Doherty, Kevin ; Adams, Rod ; Davey, Neil: TreeGNG-hierarchical topological clustering., 2005, S. 19–24

[DBB85] Dewey, John ; Boydston, Jo A. ; Baysinger, Patricia: *Democracy and education, 1916*. Southern Illinois University Press Carbondale, 1985

[Elm90] Elman, J.L.: Finding structure in time. In: *Cognitive Science* 14 (1990), S. 179–211

[EP07] Evgeniou, A ; Pontil, Massimiliano: Multi-task feature learning. In: *Proceedings of the conference on Advances in neural information processing systems: Proceedings of the 2006 conference* 19 (2007), S. 41

[FOH07] Furao, Shen ; Ogura, Tomotaka ; Hasegawa, Osamu: An enhanced self-organizing incremental neural network for online unsupervised learning. In: *Neural Networks* 20 (2007), Nr. 8, S. 893–903

[Fri94] Fritzke, B.: Growing cell structures: A self-organizing network for unsupervised and supervised learning. In: *Neural Networks* 7 (1994), Nr. 9, S. 1441–1460

[Fri95] Fritzke, B.: A growing neural gas network learns topologies. In: *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)* (1995), S. 625–632

[Fri97] Fritzke, Bernd: A self-organizing network that can follow non-stationary distributions. Springer, 1997, S. 613–618

[FV08] Fornells, Martorell J.M. Golobardes E. Garrell J. A. ; Vilasís, X.: Management of relations between cases and patterns from SOM for helping experts in breast cancer diagnosis. In: *International Journal of Neural Systems* 18 (2008), Nr. 1, S. 33–43

[GM05] Gaber M.M., Krishnaswamy S. Zaslavsky A. A. Zaslavsky A.: Mining data streams: a review. In: *ACM Sigmod Record* 34 (2005), Nr. 2, S. 18–26

[GMMO00] Guha, Sudipto ; Mishra, Nina ; Motwani, Rajeev ; O'Callaghan, Liadan: Clustering data streams. In: *Foundations of computer science, 2000. proceedings. 41st annual symposium on* IEEE, 2000, S. 359–366

[Gro87] Grossberg, Stephen: Competitive learning: From interactive activation to adaptive resonance. In: *Cognitive science* 11 (1987), Nr. 1, S. 23–63

[Har58] Hartley, H.O.: Maximum likelihood estimation from incomplete data. In: *Biometrics* 14 (1958), Nr. 2, S. 174–194

[Har90] Harnad, Stevan: The symbol grounding problem. In: *Physica D: Nonlinear Phenomena* 42 (1990), Nr. 1, S. 335–346

[HCD+05] Han, Jiawei ; Chen, Yixin ; Dong, Guozhu ; Pei, Jian ; Wah, Benjamin W. ; Wang, Jianyong ; Cai, Y D.: Stream cube: An architecture for multi-dimensional analysis of data streams. In: *Distributed and Parallel Databases* 18 (2005), Nr. 2, S. 173–197

[HF08] Holdstein, Yaron ; Fischer, Anath: Three-dimensional surface reconstruction using meshing growing neural gas (MGNG). In: *The Visual Computer* 24 (2008), Nr. 4, S. 295–302

[HV02] Hammer, Barbara ; Villmann, Thomas: Generalized relevance learning vector quantization. In: *Neural Networks* 15 (2002), Nr. 8, S. 1059–1068

[Hyo96] Hyotyniemi, H.: Text document classification with self-organizing maps. In: *Proc. of the Finnish Artificial Intelligence Conf. (STeP)* (1996), S. 64–72

[JA07] Jirayusakul, Apirak ; Auwatanamongkol, Surapong: A supervised growing neural gas algorithm for cluster analysis. In: *International Journal of Hybrid Intelligent Systems* 4 (2007), Nr. 2, S. 129–141

[Jae02] Jaeger, Herbert: Adaptive nonlinear system identification with echo state networks. In: *Advances in neural information processing systems (NIPS)*, 2002, S. 593–600

[Joa] Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines.

[Joh67] Johnson, S.C.: Hierarchical clustering schemes. In: *Psychometrika* 32 (1967), Nr. 3, S. 241–254

[Jor86] Jordan, M.I.: Attractor dynamics and parallelism in a connectionist sequential machine. In: *Proc. Eighth Annual Conf. of the Cognitive Science Society* (1986), S. 531–546

[JTC11] Jie, Luo ; Tommasi, Tatiana ; Caputo, Barbara: Multiclass transfer learning from unconstrained priors. In: *Proc. of the Int. Conference on Computer Vision* (2011), S. 1863–1870

[KKK12] Kasper, Harriet ; Koleva, Iva ; Kett, Holger: Social Media Matrix. In: *Common Value Management* (2012), S. 9

[KLR08] Kietzmann, Tim C. ; Lange, Sascha ; Riedmiller, Martin: Incremental GRLVQ: Learning relevant features for 3D object recognition. In: *Neurocomputing* 71 (2008), Nr. 13, S. 2868–2879

[Koh82] Kohonen, T.: Self-organized formation of topologically correct feature maps. In: *Biological cybernetics* 43 (1982), Nr. 1, S. 59–69

[Koh90] Kohonen, Teuvo: The self-organizing map. In: *Proceedings of the IEEE* 78 (1990), Nr. 9, S. 1464–1480

[Koh95] Kohonen, T.: Learning vector quantization. In: *The Handbook of Brain Theory and Neural Networks* (1995), S. 537–540

[KT00] Kohonen T., Lagus K. Kaski S. S. Kaski S.: Self organization of a massive document collection. In: *IEEE Transactions on Neural Networks* 11 (2000), Nr. 3, S. 574–585

[KWK05] Kirstein, Stephan ; Wersing, Heiko ; Körner, Edgar: Rapid online learning of objects in a biologically motivated recognition architecture. (2005), S. 301–308

[Lap05] Laptev, Ivan: On space-time interest points. In: *International Journal of Computer Vision* 64 (2005), Nr. 2-3, S. 107–123

[Las02] Last, Mark: Online classification of nonstationary data streams. In: *Intelligent Data Analysis* 6 (2002), Nr. 2, S. 129–147

[LG08] Lefebvre G., Garcia C.: A probabilistic self-organizing map for facial recognition. In: *Proc. of the Int. Conf. on Pattern Recognition (ICPR)* (2008), S. 1–4

[LK99] Lagus K., Kaski S.: Keyword selection method for characterizing text document maps. In: *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)* (1999), S. 371–376

[LK06] Lau K., Hubbard S. Yin H. H. Yin H.: Kernel self-organising maps for classification. In: *Neurocomputing* 69 (2006), Nr. 16, S. 2033–2040

[LMSR08] Laptev, Ivan ; Marszalek, Marcin ; Schmid, Cordelia ; Rozenfeld, Benjamin: Learning realistic human actions from movies. In: *Proc. of the Int. Conference on Computer Vision and Pattern Recognition* (2008), Juni, S. 1–8

[LSH+04] Lal, Thomas N. ; Schroder, Michael ; Hinterberger, Thilo ; Weston, Jason ; Bogdan, Martin ; Birbaumer, Niels ; Scholkopf, Bernhard: Support vector channel selection in BCI. In: *Biomedical Engineering, IEEE Transactions on* 51 (2004), Nr. 6, S. 1003–1010

[LYRL04] Lewis, David D. ; Yang, Yiming ; Rose, Tony G. ; Li, Fan: Rcv1: A new benchmark collection for text categorization research. In: *The Journal of Machine Learning Research* 5 (2004), S. 361–397

[Mar91] Martinetz, Schluten K. T.: A neural-gas network learns topologies. In: *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)* (1991), S. 397–402

[Mar93] Martinetz, T. M.: Competitive Hebbian learning rule froms perfectly topology preserving maps. In: *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)* (1993), S. 427–434

[MG08] Mandic, Vayanos P. Chen M. D.P. ; Goh, S.L.: Online Detection of the Modality of Complex Valued Real World Signals. In: *International Journal of Neural Systems* 18 (2008), Nr. 2, S. 67–74

[NBKL06] Nehmer, Jürgen ; Becker, Martin ; Karshmer, Arthur ; Lamm, Rosemarie: Living assistance systems: an ambient intelligence approach. In: *Proceedings of the 28th international conference on Software engineering* ACM, 2006, S. 43–50

[NG00] Nigam, Kamal ; Ghani, Rayid: Analyzing the effectiveness and applicability of co-training. (2000), S. 86–93

[Nig00] Nigam, A. & Thrun S. & Mitchell T. K. & McCallum M. K. & McCallum: Text Classification From Labeled and Unlabeled Documents using EM. In: *Machine Learning* 39 (2000), Nr. 2, S. 103–134

[NNM96] Nene, Sammeer A. ; Nayar, Shree K. ; Murase, Hiroshi: Columbia object image library (COIL-20). In: *Dept. Comput. Sci., Columbia Univ., New York.[Online] http://www. cs. columbia. edu/CAVE/coil-20. html* 62 (1996)

[Nob03] Noble, Weston J. & Leslie C.S. & Ie E. & Zhou D. & Elisseeff A. & Stafford W.: Semi-supervised protein classification using cluster kernels. In: *Bioinformatics* 21 (2003), Nr. 15, S. 3241–3247

[PBC13] Panzner, M. ; Beyer, O. ; Cimiano, P.: Human Activity Classification with Online Growing Neural Gas. In: *Workshop on New Challenges in Neural Computation (NC2)* (2013), S. 106–113

[PE05] Prudent, Yann ; Ennaji, Abdellatif: An incremental growing neural gas learns topologies. In: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on* Bd. 2 IEEE, 2005, S. 1211–1216

[Pea01] Pearson, Karl: LIII. On lines and planes of closest fit to systems of points in space. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (1901), Nr. 11, S. 559–572

[Pro95] Provot, Xavier: Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In: *Graphics interface* (1995), S. 147–147

[PST07] Pauwels, Eric J. ; Salah, Albert A. ; Tavenard, Romain: Sensor networks for ambient intelligence. In: *Proc. of the Workshop on Multimedia Signal Processing* (2007), S. 13–16

[QS04] Qin, A K. ; Suganthan, Ponnuthurai N.: Robust growing neural gas algorithm with application in cluster analysis. In: *Neural Networks* 17 (2004), Nr. 8, S. 1135–1148

[Rau99] Rauber, A.: LabelSOM: On the labeling of self-organizing maps. In: *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN* (1999), S. 3527–3532

[RG11] Rolf, Steil J. J. M. ; Gienger, M.: Online Goal Babbling for rapid bootstrapping of inverse models in high dimensions. In: *IEEE Int. Conf. Development and Learning (ICDL)* (2011), S. 1–8

[RM09] Ren, Jiadong ; Ma, Ruiqing: Density-based data streams clustering over sliding windows. In: *Proc. of the Int. Conf. on Fuzzy Systems and Knowledge Discovery* 5 (2009), S. 248–252

[RP02] Roy, Deb K. ; Pentland, Alex P.: Learning words from sights and sounds: A computational model. In: *Cognitive science* 26 (2002), Nr. 1, S. 113–146

[Sam94] Samaria, F.S.: Parametrization of a stochastic model for human face identification. In: *Proc. of the IEEE Workshop on Applications of Computer Vision* (1994), S. 138–142

[SF10] Shen F, Sakurai K Hasegawa O. Yu H H. Yu H: An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network. In: *Neural Computing and Applications* (2010)

[SJ06] Steil J.J., Wersing H.: Recent trends in online learning for cognitive robotics. In: *Proc. of the European Symposium on Artificial Neural Networks (ESANN)* (2006), S. 77–87

[SLC04] Schuldt, Christian ; Laptev, Ivan ; Caputo, Barbara: Recognizing human actions: a local SVM approach. In: *Proceedings of the 17th International Conference on Pattern Recognition* 3 (2004), S. 32–36

[Sne73] Sneath, Sokal R. P.H.A.: *Numerical taxonomy: The principles and practice of numerical classification.* 1973

[Ste07] Steil, Ritter H. Körner E. J.: Online Learning of Objects in a Biologically Motivated Visual Architecture. In: *International Journal of Neural Systems* 17 (2007), Nr. 4, S. 219–230

[VV05] Valera, M ; Velastin, SA: Intelligent distributed surveillance systems: a review. In: *Proc. of the Conference on Vision, Image and Signal Processing* 152 (2005), Nr. 2, S. 192–204

[Wai87] Wain, Kenneth: *Philosophy of lifelong education*. Croom Helm London, 1987

[WFYH03] Wang, Haixun ; Fan, Wei ; Yu, Philip S. ; Han, Jiawei: Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), S. 226–235

[WKCCRS01] Wagstaff K. & Cardie C. & Rogers S, Schrödl S.: Constrained k-means clustering with background knowledge. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)* (2001), S. 577–584

[WLVT06] Walchshäusl, Leonhard ; Lindl, Rudi ; Vogel, Katrin ; Tatschke, Thomas: Detection of road users in fused sensor data streams for collision mitigation. In: *Advanced Microsystems for Automotive Applications 2006*. Springer, 2006, S. 53–65

[XFHZ09] Xu, Ye ; Furao, Shen ; Hasegawa, Osamu ; Zhao, Jinxi: An online incremental learning vector quantization. In: *Advances in Knowledge Discovery and Data Mining* (2009), S. 1046–1053

[YFSW12] Yang, Hang ; Fong, Simon ; Sun, Guangmin ; Wong, Raymond: A very fast decision tree algorithm for real-time data mining of imperfect data streams in a distributed wireless sensor network. In: *International Journal of Distributed Sensor Networks* 2012 (2012)

[YS05] Yilmaz, Alper ; Shah, Mubarak: Actions sketch: A novel action representation. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2005), S. 984–989

[Zak08] Zaki, H S.M. & Y. S.M. & Yin: A Semi-supervised Learning Algorithm for Growing Neural Gas in Face Recognition. In: *Journal of Mathematical Modelling and Algorithms* 7 (2008), Nr. 4, S. 425–435

[ZFS01] Zhou, Feng ; Feng, Ju F. ; Shi, Qing Y.: Texture feature based on local Fourier transform. In: *Proc. of the Int. Conf. on Image Processing* 2 (2001), S. 610–613

[ZHM+08] Zha, Zheng-Jun ; Hua, Xian-Sheng ; Mei, Tao ; Wang, Jingdong ; Qi, Guo-Jun ; Wang, Zengfu: Joint multi-label multi-instance learning for image classification. In: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition* (2008), S. 1–8

[ZMRA13] Zepeda-Mendoza, Marie L. ; Resendis-Antonio, Osbaldo: Hierarchical Agglomerative Clustering. (2013), S. 886–887

[ZRL96] Zhang, Tian ; Ramakrishnan, Raghu ; Livny, Miron: BIRCH: an efficient data clustering method for very large databases. In: *ACM SIGMOD Record* 25 (1996), Nr. 2, S. 103–114

[ZSH13] Zhu, Xibin ; Schleif, Frank-Michael ; Hammer, Barbara: Secure Semi-supervised Vector Quantization for Dissimilarity Data. In: *Proc. of the Int. Work Conference on Artificial Neural Networks* (2013), S. 347–356