Dissertation

Learning action models by curiosity driven self exploration and language guided generalization

Maximilian Panzner
Bielefeld University

December 2019

Submitted in partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

Reviewer:
Prof. Dr. Philipp Cimiano
Prof. Dr. Angelo Cangelosi
Dr.-Ing. Hendrik Buschmeier

Contents

List of Figures

List of Tables

1 Introduction

Picture of a human and a robot walking hand in hand. Source: CITEC

Action learning is a vital requirement for many intelligent systems that have to act in their environment either alone or together with a human partner in order to achieve common goals. Especially with human partners in the loop language is of particular importance to plan and synchronize the execution of actions in an interactive setting.

Learning of action capabilities and communicative skills in the context of autonomous systems is often approached using statistical machine learning techniques, that relate the frequency of an experience with the associations that can be learned, such as an object's name. In general, most of these machine learning algorithms learn by inducing categories. These categories capture the essence of what is characteristic for instances of that particular category along with the acceptable variation.

Learning of language is closely tied to the learning of cognitive categories the linguistic constructions refer to. Intentional goal-directed actions are a special kind of categories that can be seen as a characteristic sequence of physical states of objects involved in the action and the spatial and semantic relations between them along with pre- and post-conditions that constitute the goal state.

For example, an action where an athlete is jumping over a hurdle could manifest in the following typical sequence: The athlete (A) approaches the hurdle (H), A converges to H and begins to jump, A is above H, A touched grounds again and continues to run.

In language, the respective action is referred to as "jumping over" the hurdle and could be represented by a general syntactic construction `<ACTOR> jumps over <OBJECT>`. However, the linguistic category *jumps over* does not only refer to the characteristic sequence of physical states but also refers to the goal semantics as pre- and post-conditions. First there is an actor that is in front of an obstacle and after the action is successfully performed the actor is behind the obstacle. The meaning of the symbols *in front of* and *behind* in this example is dependent on some notion of direction which is grounded in a concept of the physical world, in this example the direction of the actor's velocity vector.

Learning such categories involves inducing compact representations of characteristic sequences of states in the physical domain that capture the essence of the *jumps over* action along with the acceptable variation in the execution of the action to separate instances within that particular class from instances that belong to different classes. Learning categorical representations of action is complicated by the circumstance that the observation of a given action is often not complete within that single modality. For example, in the physical domain, there is no representation of the intentional goal that was to be reached. Linguistic representations of the same concept often suffer from ambiguities in the usage of language that cannot be resolved without involving the physical meaning that is represented by some linguistic symbol. For example, the sentences `<ACTOR> jumps over <OBJECT>` and `<ACTOR> hops over <OBJECT>` are likely to describe the same concept, while `<ACTOR> jumps on <OBJECT>` refers to a different category of action. While the syntactic constructions look very similar, the physical observations differ substantially.

In this thesis we propose a cross-modal approach to learn linguistic constructions together with their meaning in the physical domain with the potential to resolve synonym relations between surface forms at the linguistic level (e.g. "jumps over" and "hops over") by exploiting similarities in the coupled physical realization of the corresponding action, while also preventing possible over generalization, for example falsely grouping "jumps over" and "jumps on", that differ in a single word only but refer to different actions. The proposed model consists of three components: A component for action learning (see chapter 3), a component to learn linguistic constructions (see section 2.6) and the generalization coordinator that integrates information from both learning components to drive the learning and generalization process. Figure 1.1 illustrates how the model would generalize two coupled examples of action and corresponding sentences.

A suitable component for action knowledge does not exist to the best of our knowledge. The component for language learning is based on an existing model (Gaspers and Cimiano 2014) (introduced in section 2.6) and was extended according to the requirements of multi-modal grounded learning (see section 4.2). Many technological advances are inspired by natural archetypes. Also many advances in machine learning were inspired by the human brain and human cognition, such as the neural networks that model interactions between virtual neurons based on a very simple abstraction of the interaction of neurons in the human brain. We approach the multi-modal learning problem by adopting theories of early language acquisition in children and transferring them also to the domain of action.

Children tackle the problem of learning how to actively shape their environment through goal directed action mostly by imitating actions presented by parents or other tutors. According to Tomasello et al. (1993), there are three main approaches children take to learn actions from observations. First, they may start by simply

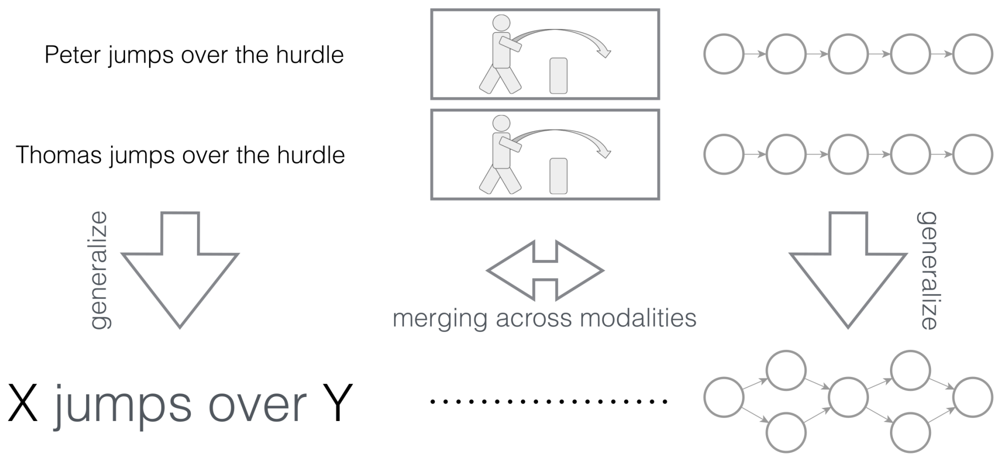

Peter jumps over the hurdle
Thomas jumps over the hurdle
generalize
merging across modalities
generalize
X jumps over Y

Figure 1.1: The model for multi modal learning learns from coupled examples of language and action given as a trajectory of objects involved in the action together with a sentence that describes the action verbally. The model consists of three components, a component for action learning, a component for language learning and a component that coordinates the learning process and drives the cross-modal induction of general categorical representations through merging similar structure. The figure shows how concrete examples are generalized into abstract linguistic constructions such as “X jumps over Y” and a generalized graphical model that probabilistically represents the two constituting action trajectories.

mimicking the movements of other actors without paying attention to the goal of the action. Second, they may establish generalized action-reaction patterns, e.g. realizing that some event causes changes in the environment in reaction to something the actor has done. And third, they combine both, the attention to action-reaction patterns and the goals an actor is trying to achieve, engaging in imitative learning of the intentional action as a whole.

According to Huttenlocher et al. (1983), children start with categories that are specific to either *self action* or *observed action*. The first category refers to actions children perform on their own and observe change in the environment induced by their actions. The second category refers to actions performed by others that are passively observed, in the case of adult tutors often accompanied by a verbal explanation of what is done. When children develop the ability to note parallels between self and observed action they extend these categories to encompass both kinds of instances. Briefly put children develop early action concepts where they are either involved as subjects or as observers and later merge concepts from both categories.

However, the detailed mechanisms that are involved in the co-emergence of linguistic and conceptual structures have not received prominent attention so far. Computational models can contribute to enhance our understanding of such processes by providing an implementable and thus explicit theory that can account for the phenomenon of co-emergence of representations across different modalities. Compared to descriptive models, computational models can be easily evaluated on experimental data, providing insights into both, computational learning and theories of early language acquisition.

We propose a computational model and thus an implemented theory of multi-modal action and language learning that partly emulates the learning through simulated self-action and observed-action with language guided generalization. The model observes actions performed by others together with a sentence describing the same action in natural language and incrementally develops generalized and grounded multi-modal categorical representations across both modalities. The generalization process builds on the knowledge acquired by observation and gradually generalizes specific representations into more general ones through simulated self-action. The mental simulation of action is based on already acquired categorical knowledge and simulates prototypical actions and utterances for existing categories based on which the model proposes to merge categories that are similar in simulation. Graphical models have recently been proposed to partially explain the nature of human cognition in a unified framework (Danks 2014). The usefulness of this approach has been demonstrated by reinterpreting a variety of cognitive theories in terms of graphical models. Our proposed model is inspired by this framework in that it is based on probabilistic graphical models to support mental simulation and merging of categories.

We focus in particular on the case of action verbs and develop the model such that it can learn the grounded meaning of a verbal construction without assuming the prior existence of a corresponding concept. Our model spells out how a learner can distill the essence of the meaning of a verbal construction as a process of incremental generalization of the meaning of action verbs, starting from a meaning that is specific to a certain context in which the verb has been encountered. We understand the meaning of verbs as evoking a simulation rather than a static concept and propose to capture the meaning of verbs via generative probabilistic graphical models. The generative action models represent the essence of the verb's meaning and can capture linguistic variation at the surface level to account for variation in action performance.

The developed model is further inspired by *usage-based* theories of language acquisition that assume that language learning proceeds from specific to general with specific constructions being incrementally generalized and entrenched, leading to different levels of generalization for different words (Behrens 2017). This is empirically backed up by findings demonstrating different levels of generalization in development within the same part-of-speech, e.g. for verbs (Tomasello 2000) or for determiners (Pine et al. 1997). The level of generalization is thus word-specific rather than category-specific. We attempt to carry this idea over to the conceptual domain to yield comparable principles describing linguistic and conceptual development and how they interact with each other.

This is reflected in that our component for action learning also implements a usage-based approach to learning actions in the sense that action learning starts with specific items that are incrementally generalized, yielding gradual abstraction over similar action trajectories. The component for action learning builds on the Hidden Markov models based representation of actions that is introduced in chapter 3. The action models develop in an incremental merging procedure to yield more general models. We hypothesize that linguistic and conceptual development might rely on akin principles, i.e. the incremental merging of specific models to yield more general models or concepts. In the case of action models this generalization is driven by the desire of a learner to yield a compact description of these domains while not losing too much descriptional accuracy.

The desire to yield compact representations corresponds to Ockham‘s razor principle and is implemented in our model as a prior that prefers simpler models. Maintaining descriptional accuracy is implemented through the model merging procedure that merges specific models guided by the desire to yield generalizable models while at the same time maximizing the likelihood of generating the observed action sequences under the generalized model in order to avoid over-generalization (see chapter 3).

The principles of *usage-based* learning are reflected in the component for language learning in the sense that structures of small complexity are learned first, emerging into more complex structure as learning proceeds. In language, structure of low complexity can for example be sequences of words associated with their corresponding meanings. As learning proceeds the model develops specific sequences of words into more abstract productive slot and frame patterns. The component to induce linguistic construction networks is based on the model published in Gaspers and Cimiano 2014 but was extended to represent the grounded meaning of verbal constructions in terms of the previously developed action model instead of the originally proposed purely symbolic representation based on predicate logic. The linguistic model in its original form is introduced in section 2.6. The extensions to the model are detailed in section 4.2.

Many computational models and theories of language acquisition so far assumed that concepts are available prior to learning the meaning of a certain linguistic construction and that novel words or constructions are merely mapped onto these existing or innate concepts. This so called *mapping paradigm* is a large simplification when learning the grounded meaning of verbal constructions. In contrast, our proposed model attempts to develop generalized categories in a *bottom-up* fashion, starting with concrete examples that are incrementally generalized without assuming prior categorical knowledge to exist. However, the model assumes the learner to have basic perceptual capabilities. For the learning of action this is the ability to recognize objects and their positions in the environment and for language we assume that the system can decompose utterances to the word level.

To our knowledge, our model is the first model that explains how linguistic verbal constructions and the action concepts they represent co-emerge, following similar principles relying on incremental generalization driven by the desire to yield more compact models that maintain predictive accuracy. Our model spells out these mechanisms in detail and thus provides a detailed implemented theory modeling how linguistic and conceptual development go hand in hand. Further, we provide a model that can both be used to 'understand' but also to 'generate' language. Our model allows a learner both to 'talk' about observed actions, being able to categorize actions and verbalize them, but also to 'simulate' an action given an (input) utterance that describes the action. In this sense our model is one of the few (cognitive) models of language acquisition bringing also comprehension and generation together in the sense of Pickering and Garrod (Pickering et al. 2013). Further, it is to the best of our knowledge the first model that explains how synonyms emerge as a byproduct of grouping similar action models.

In a second experiment we propose the *desire to induce change* as a primary driving force for a learner to develop basic perceptual capabilities as those required by the model for multi-modal learning. We demonstrate exemplarily how an agent can co-develop basic perceptual and motor skills during the autonomous exploration of affordances of a complex object, guided by the desire to induce change in the configuration of the object. To do so we implemented a reinforcement learning based model that interacts with a simulated environment. The model is introduced and evaluated in chapter 5.

In the following we define the learning problems for the multi-modal and the reinforcement learning based model and structure the development of both models along research questions.

1.1 Learning grounded representations from coupled examples of action and language

The learning problem for the multi-modal model can be stated as follows: Presented with coupled examples of language and observed action, the task of the learning system is to incrementally develop cross-modal categorical representations of action and language grounded in each other. In the example of the athlete, the input given to the system would be presented as a sentence describing the action in natural language and the perception of the scene in terms of the classified objects and their trajectories given as sequences of coordinates, such as in the following example:

Language	The athlete jumps over the hurdle	
Object 1	*athlete*	$[(x_1^1, y_1^1), \ldots, (x_1^{t_1}, y_1^{t_1})]$
Object 2	*hurdle*	$[(x_2^1, y_2^1), \ldots, (x_2^{t_2}, y_2^{t_2})]$

The sentence is given as a string of characters with no further information. Objects are given as a unique identifier (*athlete, hurdle*) and a sequence of their x and y coordinates sampled during the action recording. The resulting representations of action and language should endow the system with the capacity to reliably classify examples given in language or as action trajectories as well as generate prototypical examples of both modalities given either a class or an example of the concept in one of the modalities.

As the model is conceptually based on *emergentist* and *usage-based* theories of language acquisition in the special case of action verbs, the model should, in addition to several technical challenges, be compatible with the requirements of this theoretical foundation, which is particularly reflected in the way the model develops abstract representations of data and the inferential possibilities offered by those representations.

The multi-modal model for joint action and language learning consists of components that model action, language and a third component that coordinates the cross-modal generalization. The component for language is based on an existing model that was build around *emergentist* and *usage-based* theories of language acquisition. A suitable model for action does not exist to the best of our knowledge.

The design and implementation of the envisioned multi-modal model can particularly be structured along the lines of the following research questions:

1. How can categorical knowledge of action be represented under the prerequisite that those representations should be *emergent* and learned in an *item-based* manner?
2. Can general and simple concepts like simplicity and likelihood be applied to guide the incremental model-merging process in action learning and how do the resulting models compare to models that are learned in batch mode?
3. How can cross-modal learning of grounded action concepts and linguistic constructions be orchestrated in an emergentist learning process?

1.2 Learning perceptual and movement primitives by the desire to induce change

The learning problem for the second model is fundamentally different from the previous model as it receives no direct supervision in terms of training examples. The model learns by taking action in a simulated environment and observing change induced as a reaction to its own actions. The environment consists of a simulated toy clock and a simulated fingertip the the learning system can use to interact with the environment. The task of the system is to induce change in the environment, e.g. rotating the hands of the simulated clock with the fingertip.

Initially the system starts without knowledge about the environment, the clock or it's capacity to interact with the environment. In order to induce change in a targeted manner the system has to learn how to interpret observations from the environment and on that basis how to move the fingertip towards the clock hands. Learning occurs when the system receives a reward at certain rare occasions and has to attribute the reward to specific actions taken in the past in order to develop a predictive model that it can later be used to estimate the utility of its actions given the current situation.

The design and implementation of the model is structured along the following research questions:

1. How can a model be designed in the framework of reinforcement learning that exemplifies a possible path to how basic perceptual and movement primitives can develop without complex assumptions to begin with?
2. Is simulated curiosity or more specifically the desire to induce change in the environment sufficient to guide a learning system that is primarily based on random parallel exploration and exploitation of knowledge that was gained in the process?
3. What are possible co-factors that influence the exploration or learning process?

1.3 Contributions

The contributions of the thesis are as follows:

- We propose a model for action learning based on Hidden Markov Models that is in line with recent approaches to understand human cognition as being fundamentally explainable by graphical models (Danks 2014). The proposed model is based on graphical models that are incrementally learned from experience starting with very simple models that gradually evolve into more general models to represent a whole class of actions. The proposed model is incremental, supports simulation and allows to categorize observed actions very early in the progression of the observation sequence.
- The graphical model approach to action modeling is contrasted with an action model based on a Long-Short Term Memory recurrent neural network. It is shown that the HMM based approach is not only favorable because of the aforementioned properties but also performs comparably to the LSTM based approach in experiments that evaluate different aspects that are highly relevant in intelligent systems.
- We propose a cross-modal model for language guided action learning and categorization by observation that models the joint emergence of linguistic constructions and conceptual categories guided by the desire to produce compact

models of both modalities while not reducing the accuracy of the predictions made by the model on already observed sequences. We show that cross-modal learning can provide an advantage over uni-modal models especially when observations are ambiguous and hard to differentiate.

- The proposed model for cross-modal grounded language learning serves as an implemented theory that accounts for *emergentist* and *usage-based* hypothesis of the co-emergence of linguistic and conceptual structure. As a deterministic computational model it can be refined and tested on the basis of different datasets or replicable experiments.
- We propose a model for curiosity driven exploration of object affordances and properties. The model is a reinforcement learning approach that is similar to deep Q-networks but differs in aspects regarding strategies to explore the state-action space and a network layout optimized for learning settings with very sparse goal corresponding reward signals. We investigate several approaches to guide the exploration process through reward signals and probabilistic exploration strategies.

1.4 Outline

The remainder of this thesis is structured as follows: In the next chapter we present background information and introduce existing techniques and models that are applied in this thesis. First we introduce the concept of probabilistic time-series modelling and Hidden Markov models that are used to model action sequences in the component for action learning (chapter 3). We introduce model-merging as a technique to incrementally develop Hidden Markov models and present Long Short-Term memory networks as a state of the art baseline model for comparison (subsection 3.4.1). The Qualitative Trajectory Calculus is introduced as a basic scheme that symbolically encodes the relation between two moving objects in the component for action learning. section 2.6 introduces a model for the item-based induction of linguistic constructions that serves as the basis for the component for language learning in the multi-modal learning model that we develop in chapter 4. In a second model (chapter 5) we demonstrate exemplarily how an autonomous agent can develop basic perceptual an motor skills such as those that are presupposed by the multi-modal learning system. The thesis concludes with a summary of the proposed models and gives an outlook towards future work.

Publications

Parts of this thesis have been published earlier in the following peer reviewed articles and papers:

Journal articles:
M. Panzner, J. Gaspers, and P. Cimiano, "Modelling the co-emergence of linguistic constructions and action concepts: the case of action verbs", IEEE Transactions on Cognitive and Developmental Systems, 2019.

Conference papers:
M. Panzner, J. Gaspers, and P. Cimiano, "Learning linguistic constructions grounded in qualitative action models", IEEE International Symposium on Robot and Human Interactive Communication, 2015.

M. Panzner and P. Cimiano, "Incremental learning of action models as HMMs over qualitative trajectory representations", Presented at the Workshop on New Challenges in Neural Computation (NC2), Aachen, 2015.

M. Panzner and P. Cimiano, "Comparing Hidden Markov Models and Long Short Term Memory Neural Networks for Learning Action Representations", Machine Learning, Optimization, and Big Data : Second International Workshop, MOD 2016, Volterra, Italy, August 26-29, 2016. Revised Selected Papers, P.M. Pardalos, et al., eds., Lecture Notes in Computer Science, vol. 10122.

M. Panzner and P. Cimiano, "A Deep Reinforcement Learning Based Model Supporting Object Familiarization", Proceedings of the 7th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics, Lisbon, Portugal: 2017.

2 Preliminaries

In this chapter we establish the foundation the proposed models are based on in terms of algorithms and notations. The chapter begins with a brief introduction to time-series modelling and its mathematical formulation and based on that introduces Hidden Markov models that are used to model action dynamics and the traditional expectation maximization parameter optimization procedure. This is followed by an introduction to Long Short-Term Memory that provide a state-of-the-art baseline model from the recurrent neural networks family of models. The representation of the semantic relation between two objects is based on the Qualitative Trajectory Calculus (QTC) which is introduced in section 2.5. The model for joint action and language learning is based on an existing model for the *item-based* induction of linguistic constructions whose original formulation is introduced in section 2.6.

The second model proposed for learning perceptual and movement primitives by the desire to induce change is based on deep Q-networks, a neural networks implementation of the classical Q-learning approach, which is introduced in section 2.7.

2.1 Probabilistic time-series modeling

Time-series data, as a special case of sequential data, is given as an ordered series of observations $o_1^t = \{o_1, o_2, ...o_t\}$, from the first observation in the sequence to the current observation at time t. More precisely time-series are a set of $(time, observation)$ tuples, that are ordered in time in ascending order, but as most time-series data is sampled in fixed time intervals (e.g. temperature values measured every second) the time information is implicitly given and can be omitted. Time-series data is often highly structured in the sense that observations arriving in a certain sequence are usually strongly correlated to subsequent observations. Although not all sequential data is time-series data, e.g. gene sequences, the time-series notation and perspective is used throughout this thesis as all of the action data is time-indexed data.

There are several algorithmic options for the classification of sequence data, which can (among other categorizations) be divided into three major categories (Xing et al. 2010):

- **Feature based classification** transforms the sequence into a (fixed sized) vector and then applies conventional classification methods. For example Bag of Words methods condense a natural language text of variable length into a frequency distribution over individual words or tokens. A recent method for feature based classification is to use a recurrent neural network to digest a sequence into a fixed size vectorial representation[1] that is then used for classification in subsequent feedforward layers.

- **Distance based classification** derives a distance measure between sequences which is then used to classify the sequences. A prominent representative of this class of algorithms is the *Dynamic Time Warping* (DTW) (Keogh et al. 2000), which is used to measure the similarity of sequences which mainly vary in speed. The similarity measure is then for example used in a distance-based classifier (e.g. *nearest neighbor* classifier).
- **Model based classification** is often implemented using generative models, such as Hidden Markov models (HMM) (see section 2.2), which assume that the sequences belonging to a certain class are generated by an underlying process that can be approximated by the model. Recurrent neural networks, although they do not model a generative distribution directly, are also often used to infer a generative distribution.

The approach taken in this thesis is to represent time-indexed action sequences in probabilistic sequential models, more specifically Hidden Markov models.

The most basic approach to probabilistic time-series modeling is modeling the probability of the current observation given all past observations $p(o_t|o_{t-1}, o_{t-2}, ...)$. The joint probability distribution for a given sequence can be factored as:

$$P(o_1^t) = P(o_1) \prod_{t=2}^{T} P(o_t|o_1^{t-1}) \tag{2.1}$$

The complexity of a model that depends on the complete history of past events or observations o_1^{t-1} will grow arbitrarily large for longer sequences therefore it is necessary to limit the conditional horizon of the model to keep it computationally tractable. Different computational models employ different approaches. Two of the most prominent approaches are 1) to simply truncate the conditional horizon to the last n observations or 2) assume that the statistical process that generates the observations has an underlying internal hidden state that is not directly visible to the observer but contains all information necessary to predict future observations in the sequence. Both approaches are introduced in the following:

[1] A common approach to yield such a representation is to use a RNN to summarize the information of a given sequence into a low dimensional fixed-size vector and let a second RNN decode the fixed-size representation into the original sequence. This scheme is often referred to as autoencoders with the first RNN being the encoder and the second the decoder. In case of a space constrained representational vector in the middle the encoder has to learn a compression that captures the essence of the input sequence such that the decoder can reconstuct the original sequence from the compressed fixed-size representation.

Truncate the conditional horizon

When limiting the number of observations to consider to the previous time-step o_{t-1} we would model the density $p(o_t|o_{t-1})$, which is often referred to as a first-order Markov Model. A Markov Model represents a stochastic process that is characterized by the property that knowing the full history of a process does not necessarily result in better predictions about the future state of the process than knowing only the present state of the process. This is known as the Markov assumption, sometimes also referred to as the the "memoryless" property, because the lower the order of the Markov Model the less memory of past events the process has to predict the future state.

The Markov assumption can be formalized as the next state of a system X is independent from the previous states the system was in $X^{0:(t-1)}$ conditional on the current state X^t:

$$X^{t+1} \perp X^{0:(t-1)} | X^t \tag{2.2}$$

Markov models rely on two fundamental assumptions:

> **Limited Horizon**: The probability of the next state Q_{t+1} being s_i only depends on the previous state s_{i-1}.
>
> $$P(Q_{t+1} = s_i | Q_1, \ldots, Q_t) = P(Q_{t+1} = s_i | Q_t)$$
>
> **Temporal invariance**: The probability of the model being in state s_i is independent of the time t.
>
> $$P(Q_2 = s_i | Q_1 = s_j) = P(Q_{t+1} = s_i | Q_t = s_j); \forall t, i, j$$

The first exploits to the Markov assumption and the second refers to the inability of the model to follow concept drift that is not explicitly represented in the data.

Higher order Markov Models seem favorable as they can *remember* more history, with potentially predictive value, but the number of parameters grow exponentially with the order n, which is undesirable when the parameters are to be estimated from data[2]. Although it is generally possible to decompose an $n-th$ order Markov Model over the Alphabet $\mathcal{A}$ into a first order Markov Model by concatenating symbols into n-tuples this approach is also not generally applicable as the size of the alphabet would grow exponentially with the order n of the model.

Furthermore the Markov property represents the rather strong assumption that events outside of the conditional window do not influence the distribution of future observations, which could be unrealistic especially when the stochastic process that generates the observations is non-stationary (changing over time).

[2]The more parameters are to be estimated from a given dataset the less reliable the parameter estimates tend to be be as there are less independent examples for each individual parameter

Modeling a statistical process with an underlying hidden state structure

Another approach to make the learning problem tractable without explicitly limiting the conditional horizon is to introduce latent 'hidden' variables into the model and assume that the observations have been generated from a statistical process that can be decomposed into a sequence of subprocesses which are modeled as underlying hidden states. This assumption does not always hold for every statistical process but it is often sufficient to assume that the past sequence can be summarized concisely into a hidden state variable, which carries all the information from o_1^{t-1} that is relevant to describe the distribution of the next observation o_t (S. Bengio et al. 1996). This is the fundamental assumption in Hidden Markov Models that are introduced in the following.

2.2 Hidden Markov Models

Hidden Markov Models are a popular method for modeling time series and sequential stochastic processes with an (assumend) underlying finite-state structure and found widespread application in part-of-speech tagging and probabilistic modeling of genome sequences, as well as diverse applications in other fields. Hidden Markov Models represent probability distributions over sequences of observations under the assumption that the data generating process can be decomposed into subprocesses with an underlying finite state structure.

Hidden Markov Models can be viewed from at least two different perspectives. From the first perspective they can be considered a simple instance of dynamic bayesian networks where the network layout is simplified by the assumption that an event can only cause other events in the future but not vice versa. They can also be considered a probabilistic generalization of nondeterministic finite automata (NFA) (Stolcke et al. 1993), because as NFAs they generate or accept strings over the output alphabet by performing nondeterministic walks over the graphical structure of the model. HMMS are generative models in a sense that they can be used to generate typical data according to the model (Danks 2014, p44).

The mathematical foundation of Hidden Markov Models was developed by Baum et al. (1970) but it was Andrei Markov who gave his name to stochastic processes satisfying the Markov property, the assumption that knowing the full history of the process provides no more information about the future state of the system than knowing just the present state. The future state of a Hidden Markov Model thus only depends on the current state the model is in (see Figure 2.1).

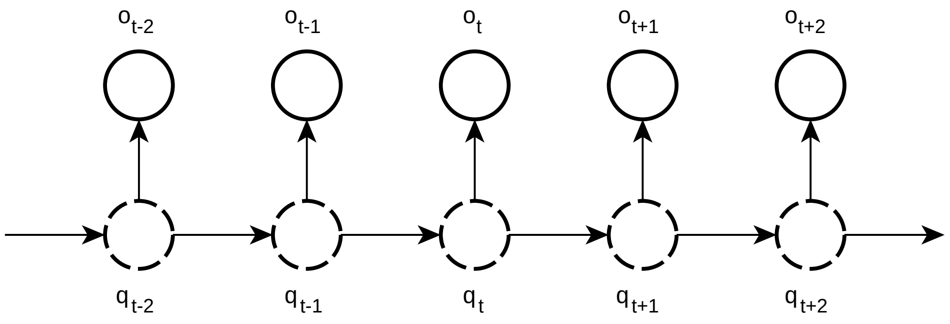


Figure 2.1: Bayesian network representing the independence assumptions of a first order Hidden Markov Model. The hidden markov process composed of the sequence of hidden states is denoted by $q_1, \ldots, q_t, \ldots$ and the “visible” observations $o_1, \ldots, o_t, \ldots$ generated while being in the respective state.

2.2.1 Definition

Hidden Markov Models are defined as a tuple

$$\lambda = (A, B, \pi) \tag{2.3}$$

with the discrete set of states S and the observation alphabet V:

$$S = (s_1, s_2, \ldots, s_N) \tag{2.4}$$
$$V = (v_1, v_2, \ldots, v_M) \tag{2.5}$$

We define Q to be a state sequence of length T and the corresponding sequence of observations O:

$$Q = q_1, q_2, \ldots, q_T \tag{2.6}$$
$$O = o_1, o_2, \ldots, o_T \tag{2.7}$$

The transition matrix is given as $A \in \mathbb{R}^{n \times n}$, where a_{ij} denotes the probability to transition from state i to state j:

$$A = [a_{ij}], a_{ij} = P(q_t = s_j | q_{t-1} = s_i) \tag{2.8}$$

$B \in \mathbb{R}^{n \times m}$ is the observation probability matrix, where $b_i(o)$ specifies the probability to make a certain observation $o \in V$ while being in state q_i:

$$B = [b_i(o)], b_i(o) = P(o = v_j | q_t = s_i) \tag{2.9}$$

The probability of the initial state q_1 being s_i is given by:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i) \tag{2.10}$$

As a first-order Markov model HMMs satisfy the Markov assumption that the current state is only dependent on the previous state:

$$P(q_t|q_1^{t-1}) = P(q_t|q_{t-1}) \tag{2.11}$$

where q_1^{t-1} denotes state sequence $q_1, \ldots, q_{t-1}$. Along with the states the observations also satisfy an independence assumption that states that the observation at time t is only dependent on the current state q_t and independent from past states and observations:

$$P(o_t|o_1^{t-1}, q_1^{t-1}) = P(o_t|q_t) \tag{2.12}$$

Observations can be either discrete or continuous. Discrete observation densities could be modeled as parametric densities or by simple normalized histograms. Continuous densities can be represented as gaussian densities or even as neural networks (Y. Bengio, De Mori, et al. 1992) that model the observation density $p(o|s)$ directly .

The matrices π, A and B are row stochastic, which means that every element is a probability and each row represents a distribution.

Many variants of Hidden Markov Models have been proposed that can be obtained by imposing additional constraints on the transition model A. For example the left-to-right HMM, which is often applied to speech recognition, can be obtained by constraining the parameters A_{ij} to be zero if $j < i$.

In the following we introduce definitions and solutions to some of the common inference problems Hidden Markov models are often applied for:

2.2.2 Probabilistic Inference in HMMs

There are three fundamental questions an HMM is often used to answer (Koller et al. 2009; Manning et al. 1999; Stamp 2004):

- **Evaluation:** Estimate the probability of a given observation sequence when the sequence of hidden states of the underlying statistical process cannot be observed.
- **Decoding:** Given an observation sequence Q and the model λ, find the most likely sequence of hidden states that best explains a sequence of observations.
- **Learning:** Given the observation sequence and the space of possible model parameters $\lambda = (A, B, \pi)$, find the parameters that maximize the likelihood of the observations under the model.

In the following we introduce common algorithms to find solutions to those questions.

Computing the probability of a sequence of observations

The process of computing the probability of a sequence of observations is often referred to as *evaluation*, because this problem can be viewed as evaluating how well a given observation sequence is predicted by the model. The problem can be formalized similar to (Manning et al. 1999; Stamp 2004) as follows: Let $O = o_1, \dots, o_T$ be the sequence of observations and $\lambda = (A, B, \pi)$ a given model, compute the probability $P(O|\lambda)$.

For a given state sequence $Q = q_1, \dots, q_T$ we can calculate the probability of the observation given a known sequence of states Q and the model λ:

$$\begin{aligned} P(O|Q,\lambda) &= \prod_{t=1}^{T} P(o_t|Q_t, Q_{t+1}, \lambda) \\ &= b_{q_1}(o_1) \dots b_{q_T}(o_T) \end{aligned} \tag{2.13}$$

using the definition of the observation probability matrix B. And the probability of the state sequence Q under the model λ:

$$P(Q|\lambda) = \pi(q_1)a(q_1 \to q_2), \dots, a(q_{T-1} \to q_T) \tag{2.14}$$

where $\pi(q_i)$ is the probability of starting the sequence in state q_i and $a(q_i \to q_j)$ is the probability to transition from state q_i to q_j. The joint probability of the observation sequence and the hidden state sequence can now be written as:

$$P(O, Q|\lambda) = P(O|Q, \lambda)P(Q|\lambda) \tag{2.15}$$

and therefore

$$\begin{aligned} P(O|\lambda) &= \sum_{Q} P(O|Q,\lambda)P(Q|\lambda) \\ &= \sum_{Q_1,\dots,Q_T} \pi(q_1) \prod_{t=1}^{T} a(q_t \to q_{t+1}) b_{q_t}(o_t) \end{aligned} \tag{2.16}$$

This derivation is straightforward but generally infeasible to calculate as it requires $(2T + 1)N^{T+1}$ calculations (Stamp 2004) [3]. Fortunally most of these calculations consist of partial results that have been calculated before. The key to making this calculations tractable is thus memorizing these partial results instead of repeatedly recomputing them. The general technique to avoid this complexity is known as *dynamic programming* (see Cormen et al. 1990 for a in depth introduction). In the context of HMMs the dynamic programming problem is often formulated in terms of trellises or lattices, which in this case is a matrix of states versus time. Each entry represents the probability of being in a certain state at a certain instant in time.

[3] There are N^{T+1} different state sequences of length N. Calculating the probability for the observation sequence O given a sequence of states requires $2T + 1$ multiplications, 2 for each time-step (transition times emission probability) plus one multiplications for the initial state.

The forward algorithm

The forward algorithm calculates the forward probability, the probability of ending up in state s_i at time t given the sequence of observations $o_1, \ldots, o_{t-1}$.

$$\alpha_i(t) = P(o_1, \ldots, o_{t-1}, Q_t = i, \lambda) \tag{2.17}$$

The forward variable $\alpha_i(t)$ is stored in the trellis at (s_i, t) and expresses the total probability of ending up in state s_i at time t. The value is calculated by summing all incoming arcs at the corresponding trellis node. The forward algorithm begins with initializing the α array at the first time step $t = 1$ with the probabilities to start the sequence at a specific state:

$$\alpha_i(t = 1) = \pi_i, \quad 1 \leq i \leq N. \tag{2.18}$$

and then recursively calculates

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_i(t) a(q_i \to q_j) b_{q_i}(o_t), \quad 1 \leq t \leq T, i \leq j \leq N. \tag{2.19}$$

The probability to reach state s_j at time $t+1$ is defined as the sum of the probabilities to come from an arbitrary state s_i to s_j while observing o_t. The total probability is then

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T+1) \tag{2.20}$$

This recursive formulation is computationally much more tractable as it only requires $2N^2T$ calculations.

The backward algorithm

The backwards algorithm computes the backward variables which represent the probability of observing a certain sequence given that we are already in state s_i at time t. The backwards variables are defined as follows:

$$\beta_i(t) = P(o_t, \ldots, o_T | Q_t = i, \lambda). \tag{2.21}$$

Then the variables are calculated backwards working from right to left through the trellis by first initializing the last step of the calculation

$$\beta_i(T+1) = 1, \quad 1 \leq i \leq N \tag{2.22}$$

Recursion

$$\beta_i(t) = \sum_{j=1}^{N} a(q_i \to q_j) b_i(o_t) \beta_j(t+1), \quad 1 \leq t \leq T, 1 \leq i \leq N \tag{2.23}$$

Total

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i \beta_i(1) \tag{2.24}$$

With the definition of the forward and backward recursion we can use any combination of these two algorithms to calculate the probability of an observation sequence. The joint probability of observing the sequence O and being in state s_i at time t can be derived as:

$$\begin{aligned} P(O, Q_t = i|\lambda) &= P(o_1 \dots o_T, Q_t = i|\lambda) \\ &= P(o_1 \dots o_{t-1}, Q_t = i, o_t \dots o_T|\lambda) \\ &= P(o_1 \dots o_{t-1}, Q_t = i|\lambda) P(o_t \dots o_T|o_1, \dots, o_{t-1}, Q_t = i, \lambda) \\ &= P(o_1 \dots o_{t-1}, Q_t = i|\lambda) P(o_t \dots o_T|Q_t = i, \lambda) \\ &= \alpha_i(t)\beta_i(t) \end{aligned} \tag{2.25}$$

Therefore we get the general formulation:

$$P(O|\lambda) = \sum_{i=i}^{N} \alpha_i(t)\beta_i(t), \quad 1 \le t \le T+1 \tag{2.26}$$

Estimating the optimal state sequence to explain a given sequence of observations

The process of estimating the best state sequence to explain a given sequence of observations is often also referred to as *decoding* because it decodes the hidden state sequence that was most likely to have produced the given sequence of observations. The problem can be formalized as: Let $\lambda = (A, B, \pi)$ be a hidden Markov model and $O = o_1, \dots, o_T$ be a sequence of observations, find the best sequence of states $Q = q_i, \dots, q_T$ that maximizes $P(Q|O, \lambda)$.

The most common approach to this problem is the Viterbi algorithm, which is another trellis algorithm very similar to the forward algorithm, except that instead of summing, the transition probabilities are maximized at each iteration.

Viterbi algorithm

The Viterbi algorithm defines

$$\delta_j(t) = \max_{q_1 \cdots q_{t-1}} P(q_1 \dots q_{t-1}, o_1 \dots o_{t-1}, q_t = s_j|\lambda) \tag{2.27}$$

as the probability of the most probable state sequence path for the given observation sequence. The algorithm stores for each point in the trellis the probability of the most probable path that leads to that particular node. The corresponding variable ψ

stores the backtrace, the node of the incoming arc that leads to this most probable path.
The algorithm is defined as follows:

1. Initialization:

$$\delta_j(1) = \pi_j, \quad 1 \leq j \leq N. \tag{2.28}$$

2 . Recursion:

$$\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) a(q_i \Rightarrow q_j) b(q_i \Uparrow o_t), \quad 1 \leq j \leq N \tag{2.29}$$

Store the backtrace

$$\psi_j(t+1) = \operatorname*{argmax}_{1 \leq i \leq N} \delta_i(t) a(q_i \Rightarrow q_j) b(q_i \Uparrow o_t), \quad 1 \leq j \leq N. \tag{2.30}$$

3. Termination:

$$\begin{aligned} \hat{Q}_{T+1} &= \operatorname*{argmax}_{1 \leq i \leq N} \delta_i(T+1) \\ \hat{Q}_t &= \psi_{\hat{Q}_{T+1}}(t+1) \\ P(\hat{Q}) &= \max_{1 \leq i \leq N} \delta_i(T+1) \end{aligned} \tag{2.31}$$

When multiple paths have the same probability we choose one of these paths at random. It is possible to extend this algorithm to not only determine the best state sequence but to determine the n-best sequences by storing the $m < n$ best previous states at a node.

Learning by optimizing the model parameters

So far we discussed the various inference problems Hidden Markov models are frequently applied to answer. However before the model can be applied, its parameters have to be estimated from data, in machine learning terminology often referred to as learning.

In case of Hidden Markov models learning from sequential data $\mathcal{D}$ is composed of two subproblems. Picking a model structure $\mathcal{M}$ (network layout) and optimizing the parameters $\theta = (A, B, \pi)$ which comprise the transition and emission densities as well as the initial state probabilities.

The Baum Welch Algorithm (Baum et al. 1970; Welch 2003) is a frequently applied iterative algorithm for calculating a maximum likelihood [4] estimate of the parameters

[4] Optimizing the likelihood of the model is probably the most common approach but there are also other approaches that optimize based on other quantities, e.g. gradient based methods (S. Bengio et al. 1996) that are frequently applied when observation distributions are modeled based on neural networks.

of a hidden Markov model. More specifically given an HMM $\lambda = (...)$ and a set of observation sequences $\mathcal{D} = \{O_1, \ldots, O_k\}$ suggest a new HMM $\lambda\prime$ that explains the observations in $\mathcal{D}$ at least as good as the the previous one such that:

$$P(\mathcal{D}|\lambda\prime) >= P(\mathcal{D}|\lambda). \tag{2.32}$$

The algorithm was first described in the late 1960s in a series of articles by L. Baum and was later formalized as an instance of the Expectation Maximization (EM) family of algorithms.

The initial set of parameters $\theta = (A, B, \lambda)$ is often chosen randomly, however it is possible to use the k-means algorithm to find initial model parameters that are closer to the maximum likelihood estimate than randomly selected parameters are for most applications (Juang et al. 1990). A good initial estimate for the parameters can drastically reduce the number of required EM iterations to converge and thus speed up the learning process.

The result of the iterative procedure would ideally be the model that maximizes $P(\mathcal{D}|\lambda)$ but practically Baum-Welch often results in a solution that represents a local minimum. To reach a global optimum the algorithm depends on having a good initial starting point in a region of parameter space that is already close to the desired optimal solution. To mitigate the effect of the initialization the algorithm is often run multiple times and the best solution is picked afterwards. In practice the emission densities B are found to be more sensitive to the initialization than the A and π parameters for which different starting points often result in less variance in the found soulution (Manning et al. 1999).

Baum-Welch algorithm

The basic idea of the algorithm is to keep track of the transitions and emissions that were most frequently used in calculating the probability of the observation sequence to construct a revised model in the subsequent iteration that assigns larger parts of the probability mass to frequent transitions and emissions and thus increases the overall probability of accepting the observation sequences under the revised model.

The learning problem is stated as follows: Given a HMM λ and N sequences of observations $O = \{[o_1, \ldots, o_{T(1)}], [o_1^n, \ldots, o_T^n(N)]\}$, find a model λ that maximizes the likelihood.

$$P(O|\lambda) = \prod_{n=1}^{N} p(o^n|\lambda) = \mathcal{L}(\lambda|O) \tag{2.33}$$

The process begins with a preselected (e.g. by k-means initialization) or randomly chosen model λ and calculates expectations on the transitions between states and updating model parameters using those estimates. The process is repeated until it converges, re-estimating the parameters in each iteration as follows:

Estimation-Step In the expectation step the algorithm calculates the following quantities given a sequence of observations O by summing over time:

$$\sum_{t=1}^{T} \gamma_i(t) = \text{expected number of transitions from state } i \text{ while emitting } O \quad (2.34)$$

$$\sum_{t=1}^{T} p_t(i,j) = \text{expected number of transitions from state } i \text{ to } j \text{ while emitting } O \quad (2.35)$$

with $p_t(i,j)$ and $\gamma_i(t)$ being defined as follows:

Let $p_t(i,j), 1 \leq t \leq T, 1 \leq i,j \leq N$ be the probability of transitioning from state i to j at time t given the observation sequence O:

$$\begin{aligned} p_t(i,j) &= P(Q_t = i, Q_{t+1} = j | O, \lambda) \\ &= \frac{P(Q_t = i, Q_{t+1} = j, O | \lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_i(t) a(q_i \Rightarrow q_j) b(q_i \Uparrow o_t) \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t) \beta_m(t)} \\ &= \frac{\alpha_i(t) a(q_i \Rightarrow q_j) b(q_i \Uparrow o_t) \beta_j(t+1)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m(t) a(q_m n \Rightarrow q_j) b(q_m \Uparrow o_t) \beta_n(t+1)} \end{aligned} \quad (2.36)$$

and

$$\gamma_i(t) = \sum_{j=1}^{N} p_t(i,j) \quad (2.37)$$

denotes the probability to reach state i at time t with the symbol observed at that time being O.

Maximization-Step In the maximization step the algorithm recalculates the model parameters based on previously calculated expectations.

$$\begin{aligned}\hat{\pi}_i &= \text{ expected frequency in state } i \text{ at time } t = 1 \\ &= \gamma_i(1)\end{aligned} \tag{2.38}$$

$$\begin{aligned}\hat{a}(q_i \Rightarrow q_j) &= \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \gamma_i(t)}\end{aligned} \tag{2.39}$$

$$\hat{b}(q_i \Uparrow o_k) = \frac{\text{expected number of observations of } o_k \text{ while in } q_i}{\text{expected number of visits to } q_i} \tag{2.40}$$

$$= \frac{\sum_{\{t:o_t=k, 1 \le t \le T\}} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{2.41}$$

The algorithm proceeds iteration between the expectation and the maximization steps until convergence.

2.3 Bayesian model merging for Hidden Markov model induction

In this section we introduce the model merging approach to inducing Hidden Markov models from data that constitutes the basis for the incremental learning procedure in the component for action learning. Model merging is an active field of research and not limited to Hidden Markov models but applicable to a variety of complex model types that can be created by merging simpler ones. There is ongoing research to unify model merging approaches, e.g. Brunet et al. (2006) propose unified operators for different model types such as Entity Relationship Models and State Machines, as a basis for a more general formalization of the model merging paradigm.

Here we follow the best-first bayesian model merging approach (Omohundro 1993; Stolcke et al. 1993) that is mainly characterized by the greedy optimization that drives the optimization using a bayesian criterion. In contrast to the Baum-Welch algorithm (section 2.2.2), which optimizes model parameters given a fixed model structure, model merging optimizes the model topology along with the model parameters to fit the available training data. In essence, the bayesian model merging proceeds in a bottom up fashion from specific models representing concrete examples to general models that abstract from concrete examples.

Training examples are first modeled as individual HMMs that are maximally specific to their respective training instance. The individual models can be merged to a single large model, which then consists of maximally specific sub-path that correspond to the individual training examples. The resulting models are thus unable to generate or accept any other sequence, that was not in the constituting data. In order to yield general models, the approach searches for similar structure within the model that can be collapsed into a more abstract and thus general sub-structure. Learning, as abstraction from concrete examples, results from structure merging in contrast to directly optimizing a fixed set of model parameters as in the Baum-Welch algorithm. In the following we introduce the intuition and concrete implementation of the bayesian model merging approach in detail.

HMM induction as guided search through the space of possible models

Learning a Hidden Markov model typically involves first choosing the model structure and second the estimation of optimal model parameters given an optimization criterion based on the chosen fixed structure. A good choice of structural and model parameters usually yields a good model to represent the distribution or process assumed to have generated the data. Model merging makes the choice of structure an integral part of the learning and optimization process. Traditional training procedures perform a search through the parameter space of the chosen model along the optimization landscape that is defined by the optimization criterion. The model merging approach on the other hand induces complex models by gradually merging simpler models and thus performing a search through the space of possible model structures in addition to the space of possible model parameters within the given family of models. By merging two simple models and collapsing similar structure within the resulting model, the approach automatically adapts the representational power of the model to the characteristics of the underlying data and is thus less prone to overfitting, without requiring explicit regularization other than the optimization criterion.

In essence, model merging can be considered a guided search through model space starting with very concrete models and proceeding to more general models that represent a concept on a high level of abstraction. The search has no intrinsic stopping criterion in the sense that there is a single best model to be found as a result to the search. Instead there is a set of models that fits the optimality criterion in between highly specialized item-based (specific to a single item) models that are prone to overfitting and very general models that could probably underfit the real data distribution. A proper criterion is crucial to guide the search for an optimal solution. Since every merging step is a potential generalization step in the model search, the optimization criterion effectively determines the degree of generalization in the resulting models.

There is no principled limitation on the methodology the stopping criterion is based on. It could be for example based on the global characteristics of the target model, e.g. the model size or the complexity of the resulting models. Other options include measures that determine the fit of the data given the model in terms of a loss measure, e.g. squared error or the negative log likelihood. It is also possible to use simple heuristic based approaches like thresholding which impose fixed bounds on the loss.

The model merging method for Hidden Markov models relies on five major elements:

1. A method to construct initial HMMs from data.
2. A method to merge sub-structure within the model that has the potential to represent a more general construct when collapsed.
3. An error measure that ranks the utility of several candidate sub-structures to be merged and limits the generalization.
4. A search strategy that selects sub-structure to be merged and thus guides the search for the optimal model in the space of all possible models.

In line with Stolcke (Stolcke et al. 1993) we translate these elements to the domain of Hidden Markov models as follows:

1. Each new sample sequence is translated to a Hidden Markov model where each symbol in the sequence is represented by a dedicated state in the model such that the resulting model generates all and only the symbols in the sequence.
2. Merging sub-structure within HMMs can essentially be reduced to merging two individual states. Merging two states gives the resulting merged state emission and transition characteristics that are a weighted average of the corresponding distributions from the states which have been merged.
3. While there is no principled limitation on the shape of the error measure, we follow Omohundro (1993) and formalize the error in terms of a bayesian probability that provides a flexible basis to limit generalization based on cognitively plausible considerations.
4. We employ a search strategy based on the greedy *best-first* approach that at each step merges the states with the highest score according to an evaluation function. We extend this strategy with a lookahaed mechanism as described in section 3.3.3.

The concrete implementation of how initial models for individual training examples are constructed is further explained in section 3.3.3. Merging of states within a given model relies on the concrete distributions that govern the transition and emission within the model to be mergeable as well. The concrete implementation of the state merging procedure is detailed in section 3.3.3. The implementation of the search strategy in the component for action learning is based on the *best-first* strategy as originally proposed but differs in a *lookahead* strategy that continues the search for a few more steps even when the originally proposed strategy would stop the search.

Generalization and inductive bias

Bayesian model merging as it is employed in this thesis can be seen as a guided search for the optimal model in the space of all possible models. The search is guided by a criterion that assesses the goodness or optimality of found solutions.

Optimizing exclusively using a criterion that corresponds to the likelihood of the found solution given the data (maximum likelihood) would not necessarily result in a model that generalizes well because the Hidden Markov models in our approach are conceptually unrestricted in size while being expressive enough to simply duplicate the training data. Duplication of training data generally leads to models that merely cover examples which are very close to the memorized examples but the model would be unable to generalize from the data by abstracting common properties that comprise the essence of what defines the training examples conceptually. This corresponds to the well known *over-fitting* effect. On the other hand, generalizing too much would lead to *under-fitting* the data, which in the extreme, results in models that would take their decisions independent from evidence.

Generalization in machine learning is often achieved by introducing a set of assumptions on the relation of input to output values. This is often referred to as the inductive bias or learning bias of a learning algorithm. In essence these assumptions prefer one generalization over the other (Mitchell 1980). There are many classical examples for inductive bias applied in a variety of models and learning algorithms. The nearest neighbor classifier for example relies on the assumption that examples that are similar to each other according to some metric tend to belong to the same class. The support vector machine on the other hand relies on the assumption that distinct classes tend to be separable by large margin boundaries.

Following Omohundro (1993) and Stolcke et al. (1993) we take the bayesian perspective, which can be considered a joint approach combining the data dependent likelihood with a prior that constitutes a global data independent measure.

Bayesian model merging

Bayesian model merging (Omohundro 1993; Stolcke et al. 1993) is an instance of model merging that is based on a bayesian inductive bias as the optimization criterion to guide the optimization process. In general, learning from data is generalizing from it and the inductive bias is the measure that separates wanted from unwanted generalization, usually by introducing a set of assumptions to the learning algorithm.

A simple maximum likelihood approach to identify the optimal model in the space of all possible models (including structural parameters) would result in a model which could simply duplicate the training data as the class of models is usually rich enough and leaving the model structure open to the optimization process grants enough model capacity in terms of free parameters to perform rote learning. This is different from traditional optimization with fixed model structure which imposes a constraint on the form of the maximum likelihood solution by forcing it to abstract from data in order to represent complex knowledge in a constrained set of parameters. Richer models with higher model capacity often result in *over-fitting* models that essentially perform rote-learning by explicitly memorizing parts of the training data. Without this implicit bias towards more general models (e.g. by having a fixed model size) the model merging approach requires an explicitly given bias that drives the optimization towards more general models while effectively limiting over-generalization.

In bayesian statistics this bias can elegantly expressed as a prior that favors particular models over the others without considering actual evidence in addition to optimizing the likelihood alone.

Each optimization step then maximizes the model's posterior according to the bayesian definition:

$$posterior \propto likelihood \times prior. \tag{2.42}$$

The model merging approach can be seen as a search through the space of all possible models, in this figure the goal of the search is finding a model for which the posterior probability reaches its maximum. The bayesian perspective offers an elegant solution to expressing the wanted degree of generalization as prior and likelihood can be chosen such that they constitute two opposing forces, with the likelihood having its maximum at the initial model comprising only the maximally specific sub-paths that represent concrete training examples. As this model by definition would maximally *over-fit*, the role for the prior is then to drive the generalization process towards more abstract models.

The bayesian perspective also simplifies the task to stop the generalization process at the optimal degree of generalization and thus to effectively prevent *under-fitting* of data. The generalization process can simply be stopped when no further increase in posterior is possible through any of the available merging steps, as this essentially corresponds to the equilibrium between prior and likelihood or, assuming a good choice of the prior, between *over-* and *under-fitting*.

A good prior results in models that constitute a good compromise between over- and under-generalization and in addition are compatible with *emergentist* theories of concept acquisition. The latter also demands not to introduce any strong biases that can not be justified when modeling early language acquisition. In the following we discuss the relation between complexity and generalization as these concepts are helpful tools in defining such a prior.

Minimum description length The notion of minimum description length gives an alternative interpretation of bayesian inference based on information theoretic concepts. Maximizing the joint probability of having a model M to explain the data X is

$$P(M, X) = P(M)P(X|M) \tag{2.43}$$

is equivalent to minimizing

$$-\log P(M, X) = -\log P(M) - \log(P(X|M)). \tag{2.44}$$

In information theory the negative logarithm of the probability of a discrete event E can be interpreted as the optimal code word length for communicating instances of E. The terms in the above equation can be interpreted accordingly, which leads to interpreting $-\log P(M)$ as the description length of the model under the prior distribution (of possible models) and $-\log P(X|M)$ as the description length of the data when using a code that is optimal for M. The minimization of the description length can be interpreted as compressing the training data, describing it using fewer symbols by exploiting regularities in the data without sacrificing accuracy compared to describing the data literally (lossless encoding).

This interpretation opens up the correspondence to cognitive theories of learning. A complexity and space limited learning system such as the human brain has to inevitably seek for the most compressed representations of concepts possible in order to cover as many diverse concepts as possible while still being so specific to allow us to make good decisions in specific situations. The description length essentially describes how much information a given message or event actually conveys, which quantifies the degree of surprise or unexpectedness.

Kolmogorov complexity There is another popular approach towards the definition of complexity or simplicity which is highly related to the description length principle, the so called *algorithmic complexity* also known as the *Kolmogorov complexity*. The main idea of this complexity measure is that the complexity of a string of characters reflects the degree of incompressibility, which can be measured by the length of an algorithmic expression or computer program that can generate that string. Simple strings that bear a high degree of regularity can be expressed by very simple programs while strings without any regularities have to be printed verbatim by the program.

Shannon information Both interpretations relate also to Shannon (1949), who defines complexity as surprisingness given the model. In this interpretation having a model that precisely predicts future outcomes of a statistical process reduces the surprisingness of new data, because the model can already accurately predict what data would be seen next.

Mutual information and sufficient statistics Kolmogorov and Shannon have a different view on information content of a single object. For Shannon information can be represented in terms of a random variable. Kolmogorov puts an individual sequence as the conveyer of information. Both interpret information content (and thus complexity) in terms of mutual information, the information that one object gives about the other. Another concept that is related to mutual information is the notion of sufficient statistics, which in an algorithmic setting describes the *meaningful* information that can be extracted from data, leaving the rest of the data as noise.

Optimizing a model for maximum likelihood with unconstrained model complexity likely results in complex models that, in the worst case, simply duplicate their underlying training data. Intuitively, having a prior based on a notion that strives towards models that are smaller and simpler in general would be a good choice as it effectively limits the tendency towards over-fitting in maximum likelihood optimization by penalizing overly complex models.

With a simplicity prior, the product of prior and likelihood $P(M)*P(X|M)$ represents that compromise between the simplicity that is embodied in the prior and the fit to the data that is reflected in a high model likelihood that provides the model with good coverage of the target domain.

The information theoretic viewpoint provides a good basis to derive a notion of complexity from. The choice of the prior for the component for action learning is further discussed in section 3.3.3.

Likelihood computation The likelihood can be exactly calculated relative to a given dataset X as the product of the individual sample probabilities:

$$P(X|M) = \prod_{x \in X} P(x|M) \tag{2.45}$$

with

$$P(x|M) = \sum_{q_1 \dots q_l \in Q^l} p(q_I \to q_1) p(q_1 \uparrow x_1) \dots p(q_l \uparrow x_l) p(q_l \to q_F) \tag{2.46}$$

where l is the length of the sample and $q_1 \dots q_l$ denotes a path through the model with q_I, q_F being the initial and final states.

The calculation of the exact likelihood requires iterating over the complete dataset X. Omohundro (1993) suggests to simplify this calculation using the Viterbi approximation, the assumption that the generation probability of a given sample is largely dominated by a single generation path in the HMM. In model merging this assumption is justified by the process that develops complex models from simple models that are maximally specific to a given training instance $x \in X$ and thus constitute Viterbi paths in themselves. Furthermore, the likelihood computation is only used to compare models before and after merge operations, which makes the compared models structurally highly similar. Applying the Viterbi approximation replaces the inner summation with the term with the highest contribution. The likelihood can now be calculated as:

$$P(X|M) \approx \prod_{x \in X} \max_{q_1 \cdots q_l} p(q_I \to q_1) p(q_1 \uparrow x_1) \dots p(q_l \uparrow x_l) p(q_l \to q_F) \tag{2.47}$$

Grouping the terms in this expression by states, leads to the form:

$$P(X|M) \approx \prod_{q \in Q} \left(\prod_{q\prime \in Q} p(q \to q\prime)^{c(q \to q\prime)} \prod_{\sigma \in \Sigma} p(q \uparrow \sigma)^{c(q \uparrow \sigma)} \right) \tag{2.48}$$

where $c(q \to q\prime)$ and $c(q \uparrow \sigma)$ correspond the total counts of transitions and emissions occurring along the Viterbi path associated with the samples in the underlying dataset. The emission and transition counts along the viterbi paths (viterbi counts) can be tracked implicitly without having to reparse the complete dataset after each merge operation. Assuming that merges preserve the viterbi paths, viterbi counts can be optimistically updated as follows: When initial models are created, the viterbi counts are set to one, corresponding to the single sample any initial path was created for. After merging any two states q_1 and q_2 the corresponding counts are simply added and recorded for the resulting merged state. For example given the transition counts $c(q_1 \to q')$ and $c(q_2 \to q')$ in the current model, the resulting merged state q_3 would be assigned

$$c(q_3 \to q') = c(q_1 \to q') + c(q_2 \to q') \tag{2.49}$$

as the new transition counts. Emission counts are updated accordingly. This optimistic update is correct as long as all Viterbi paths that include the transitions $q_1 \to q$ and $q_2 \to q'$ retain their most likely paths with the merged states q_1 and q_2 being simply replaced by the resulting state q_3. This update could introduce errors if other samples change their paths to include $q_3 \to q'$ as a result of the merge operation. Incorrect counts will eventually be replaced by counts that are more recent as new data is incorporated into the model.

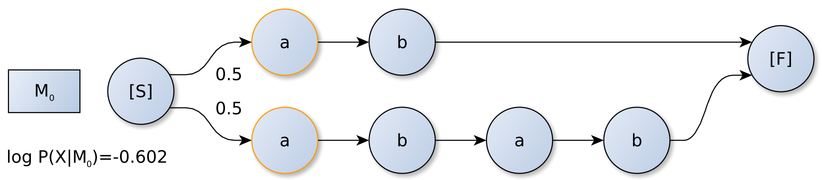
Example

As an example we consider a sequence of models obtained by merging samples from an exemplary language $(ab)^+$. This example was reproduced from Omohundro (1993). All transitions and emissions in the example that are not specifically labelled have a probability of 1.0.

The model induction process starts with a model M_0 that is constructed from two sequences $X = \{ab, abab\}$ obtained by sampling from the regular language $(ab)^+$. These two sequences constitute the data for the example.

To incorporate the sequences into a (initially empty) model we first transform both sequences into maximally specific Markov chains that produce their respective sequences with probability 1.0 and integrate them into a new Markov model by introducing transitions from the start state of the model to the first states in the data sequences. As both sequences are observed once, the transition probability mass of the start state is equally distributed. The last states of the sequences are terminated by transitioning to the final state of the Markov model.

The resulting model M_0 can be seen in the following figure:

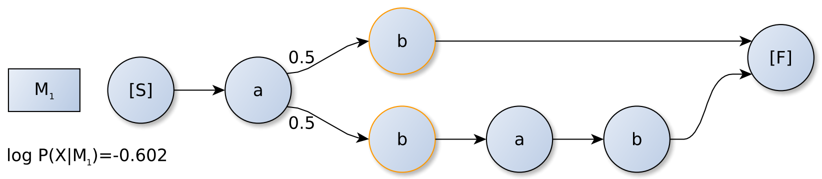


The likelihood of model M_0 can be calculated according to Equation 2.48. Note that all calculations are done in log-space to improve numerical stability (see section 3.3.3). The log likelihood of the initial model can be calculated as

$$\log P(X|M_0) = log(0.5) + log(0.5) = -0.602.$$

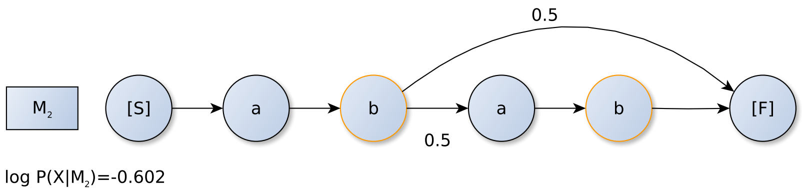
As most transitions have a probability of 1.0, only the two transitions emanating from the initial state contribute to the result.

Starting from the first model the algorithm would select the first two states to be merged according to their emission and transition similarities. This similarity measure can be seen as an approximation to the expected drop in likelihood of the resulting model. Merging both highlighted states (see section 3.3.3) yields model M_1:

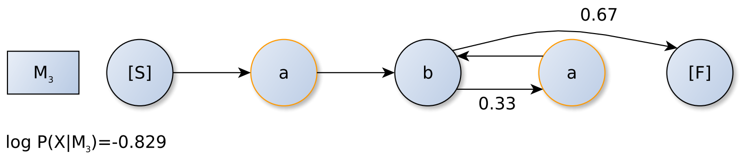


The overall model likelihood does not change, because we have again only two paths contributing to this measure, both having a probability of 0.5.

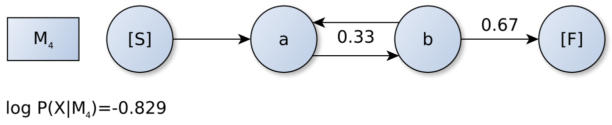
After merging the next two highlighted candidates, both emitting a b symbol, we yield model M_2 again without a drop in model likelihood:



The next merge yields M_3 where we suffered a first drop in likelihood, but as we have a prior favoring less states, M_3 is still more preferable compared to M_0:



This model is now cyclic and able to generate more sequences than the original sequences $ab, abab$ it was created from. In fact being cyclic enables M_3 now to generalize to the whole language $(ab)^+$ which was used to generate the constituting samples in the first place.



The last merge simplifies the model further to the most compact form to generate the example language, again without a drop in likelihood.

2.4 Long Short-Term Memory

Long Short Term Memory networks (LSTM) are generative recurrent neural networks to model sequences of arbitrary length. Recurrent neural networks differ from regular feed-forward networks in that they introduce loops into the network structure which allows information to persist from one step to the other. While loops are a powerful tool to model long term dependencies by connecting previous information to current tasks, traditional RNNs had difficulties learning how to connect that information when the number of intermediate steps was large. The fundamental problem is associated with the *vanishing gradient* effect, that was analyzed in detail in Y. Bengio, Simard, et al. (1994) and Hochreiter (1991).

While Hidden Markov models decompose an assumed statistical process into sub-processes and embed them into a graphical structure that mediates the transition between sub-processes, a neural network models an assumed functional relation between input and output variables as a computational graph that incrementally transforms the input as it flows through the graphical structure by applying basic mathematics operations at each node. The graphical structure thus constitutes a decomposition of a larger functional interrelation between input and output predictions.

Unlike the Hidden Markov Model, LSTMs condition on the complete history of the sequence using an explicit memory mechanism. The amount of memory is limited and fixed but the model can learn which parts of the history are relevant for future predictions and how to efficiently encode the relevant information. For this purpose the model has a tunable gating mechanism to control the flow of information into and out of the memory. This gating mechanism is the reason why LSTMs are much more stable with regard to the *vanishing* or *exploding gradient* effect, which drastically limited the applicability of recurrent networks for many complex problems.

LSTM networks found widespread application in, e.g. recognizing patterns in sequential data, such as text, handwriting, genomes or analysis of numerical time-series. Depending on the task at hand, LSTMs are either applied as standalone predictors or together with other types of neural networks in a layered (deep) architecture.

Several variants of the block architecture of LSTM memory cells have been proposed (Gers et al. 2000; Greff et al. 2015) that mainly differ in the configuration of the multiplicative gates that regulate information flow. An example of a simple recurrent network unit and a LSTM block with peephole connections can be seen in Figure 2.2. The implementation in this thesis is based on the original architecture proposed by Hochreiter and Schmidhuber (1997) with input, output and forget gates.

LSTM network architecture

As a special kind of RNNs, Long-Short Term Memory networks are capable of remembering long-term dependencies. In fact they were originally proposed by Hochreiter and Schmidhuber (1997) to specifically tackle the problem of long-term dependencies that were difficult to learn by traditional RNNs due to the *vanishing gradient* problem. While the latter has a single loopback connection, that carries information from the current time-step over to the next time-step verbatim, LSTMs introduce a sophisticated gating mechanism that enables the network to learn when to carry information over to the next time-step and to what extent. Furthermore, the mechanism allows the network to learn to protect its *memory* from changes and also to completely forget past information. The unit composed of a traditional

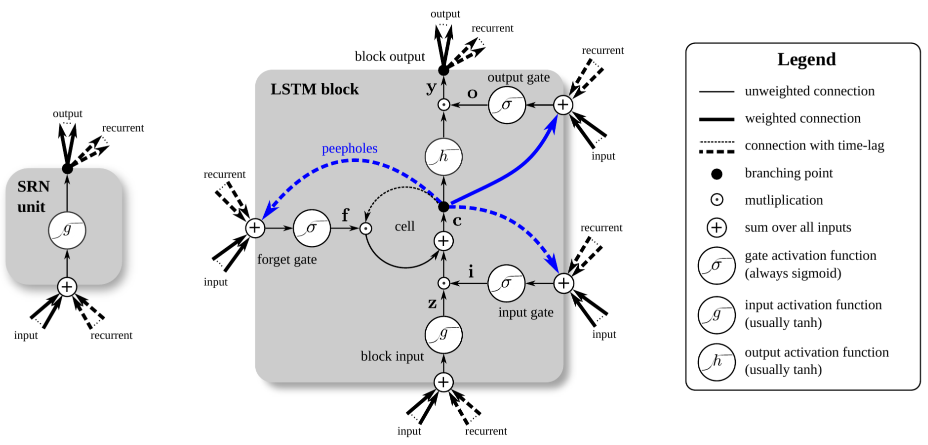


Figure 2.2: This Figure shows a traditional RNN/SRN (simple recurrent network) on the left and a LSTM Block on the right. The Figure originates from Greff et al. (2015). The blue connections refer to the so called *peephole* that are not part of the originally proposed architecture. Information flows from bottom to top through the cell and is, in the case of an LSTM cell, modified and remembered controlled by the gating mechanism.

feedforward sigmoid connection and the gating mechanism is referred to as a *memory cell.*

While information flows through a *memory cell*, the LSTM has the ability to remove or add information to the cell state regulated by the so called *gates*.

Multiplicative gating mechanism

Gates consist of a sigmoid connection followed by a pointwise multiplication operation. As the sigmoid function σ outputs numbers between zero and one, the following pointwise multiplication can essentially decide how much of the original information is allowed to pass the gate. A value of zero means that no information is allowed to pass and a value of one lets information flow unchanged. The original LSTM memory was proposed with three gates to control to *forget* past information and control the *input* of new information into the cell state or the *output* of information to the current prediction. Figure 2.2 depicts a single LSTM cell together with the much simpler traditional RNN unit.

Forward pass

The activations of the input(i_t), forget(f_t) and output(o_t) gates in a forward pass at time t are calculated as:

$$\begin{aligned} i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\ o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o). \end{aligned}$$

Where x_t represents the input at the current timestamp. Each of the gates have their respective tunable weights w_i, w_f, w_o and bias vectors b_i, b_f, b_o.

The output h_t of the LSTM cell is calculated as:

$$\begin{aligned} \hat{c}_t &= \tanh(w_c[h_{t-1}, x_t] + b_c) \\ c_t &= f_t * c_{t-1} + i_i * \hat{c}_t \\ h_t &= o_t * \tanh(c_t) \end{aligned}$$

The calculation proceeds in two steps, first a candidate cell state $\hat{c}_t$ is calculated, which in turn is used to calculate the actual cell state c_t considering the previous cell state weighted by the forget gate activation and the amount of information to input from the current candidate state. How much information is allowed to exit the cell is determined by applying the activation function on the current cell state and multiplying the result with the activation of the output gate.

2.5 Qualitative Trajectory Calculus

The qualitative trajectory calculus (QTC) is a representation scheme to logically express the relation between moving objects with respect to each other [5]. It provides a sound mathematical formulation to reason about moving objects qualitatively by abstracting from raw euclidean coordinates and their temporal dynamics. Different types of spatial QTC have been defined, belonging to two main categories: QTC - Basic (QTC_B) and QTC - Double Cross (QTC_C). The two categories and their subtypes consider pairs of objects (o_1, o_2) and mainly differ in the number and type of physical factors considered (e.g. relative speed, distance, directions, etc.) and in the dimensionality of the underlying data. The two objects are represented as point-objects (e.g. their center of mass) and their physical extent is neglected. The QTC framework defines a state descriptor $(C_1, \ldots, C_n)$ to express the relation between the two objects qualitatively, where each C_i represents a so called constraint with the following interpretation:

[5] There is ongoing effort to extend the QTC family of representations to other domains, such as networks or higher dimensional spaces. In this thesis we consider only QTC types that represent two dimensional spatial relations between two objects.

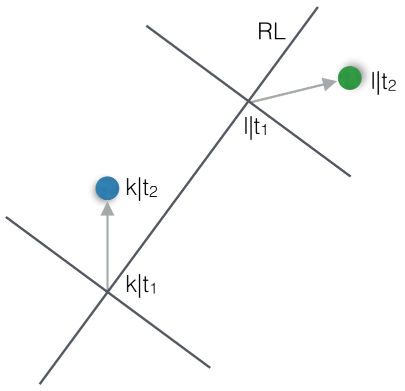


Figure 2.3: This figure shows two moving objects k and l at two different time points t_1 and t_2. In this example k is moving towards l at time t_1 on the left hand side of the reference line RL from k to l, l is moving away from k on the left hand side of the reference line from l to k. The corresponding QTC_{C1} relation is $(-+--)$. Reproduced from (Weghe et al. 2005)

(**A**) Distance constraint: movement of k with respect to l at time t_1:

- - k is moving towards l
- 0 k is not moving relative to l
- $+$ k is moving away from l

(**B**) movement of l with respect to k at time t_1: same as above but with k and l interchanged

(**C**) Speed constraint: Difference in speed of k and l.

- - k is moving slower than l
- 0 k and l move with equal speed
- $+$ k is moving faster than l

(**D**) Side constraint: Movement of k with respect to the reference line RL at time t_1

- - k is moving to the left-hand side of RL
- 0 k is moving along RL or not moving at all
- $+$ k is moving to the right-hand side of RL

(**E**) Side constraint: Movement of l with respect to RL at time t_1: same as above but with k and l swapped

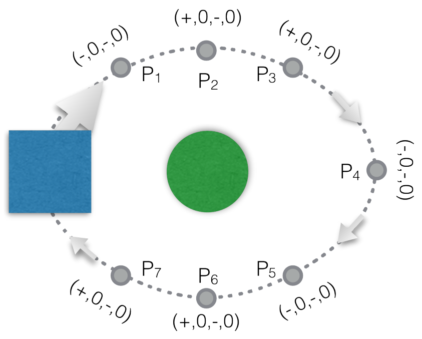


Figure 2.4: Example sequence of a "circles around" action. The blue rectangle circles around the green circle on a smooth elliptic trajectory. The QTC_{C1} relations only change at the marked positions P_2, P_4, P_6 and remain unchanged in between.

(F) Angle constraint: Minimal angle θ_k, θ_l between speed vector of the objects k, l and the reference line RL

- - θ_k is less than θ_l
- 0 θ_k is equal to θ_l
- + θ_k is greater than θ_l

The basic QTC_{B1} type only considers distance and relative speed (difference of distances) and thus can be easily applied also in non-cartesian spaces (e.g. joint space in robotics) where only a distance metric is known. QTC_{C1} also considers the direction of movement with respect to a reference frame between the two objects, which takes the form of a double-cross (Freksa 1991; Freksa 1992b; Freksa and Zimmerman 1992; Zimmermann and Freksa 1993; Zimmermann and Freksa 1996). The double-cross is defined by a reference line RL between the object centers and one line perpendicular to RL ($RL \perp 1$, $RL \perp 1$) for each of the object centers (see Figure 2.3).

As an illustration of the action representation, consider Figure 2.4, which depicts a rectangle circling once around a circle on an elliptic trajectory. At the first marked position, P_1, the square is moving on the top left of the circle, corresponding to the QTC descriptor (-,0,-,0). At P_2, the square is on top of the circle and instead of approaching the circle the rectangle veers away from the circle now, resulting in the first constraint of the QTC relation to change from $-$ to $+$ yielding (+,0,-,0) as the new relation. In this illustration of very smooth trajectory, the QTC relations would only change at the positions P_2, P_4, P_6 and remain unchanged in between, leading to subsequences with many repeated QTC relations in between which are subject to a logarithmic compression scheme that is introduced in section 3.3.1. Trajectories from actions performed by humans however are much more cluttered and thus the sequences of QTC relation contain many additional transitions.

QTC Basic	A	B	C	D	E	F
QTC_{B1}	✓	✓				
QTC_{B2}	✓	✓	✓			
QTC Double-Cross	**A**	**B**	**C**	**D**	**E**	**F**
QTC_{C1}	✓	✓	✓	✓	✓	
QTC_{C2}	✓	✓	✓	✓	✓	✓

Table 2.1: Four variants of spatial QTC as combinations of the six basic constraints

2.6 Item-based induction of linguistic constructions

In the following we introduce a model for the item-based induction of linguistic constructions that serves as the basis for the component for language learning in the proposed multi-modal learning model. In this section we describe the model as it was originally proposed by Gaspers and Cimiano (2014). The extensions to the model that are necessary for the model for multi-modal grounded action and language learning are described in section 4.2.

The model for inducing generalized linguistic constructions acquires syntactic constructions and a lexicon from examples comprised of input across two different input channels: a language channel and a visual channel. The language channel presents utterances as sequences of words to the system while input from the visual channel is in the original version of the system represented as a symbolic description of the visual context. The visual meaning representation (MR) is comprised of a set of actions taking place while the utterance is being uttered. Each action $mr_i \in MR$ is represented by means of a predicate logic formula, comprising a predicate ξ along with a set of arguments. The learning process and an example of a verb-specific construction stored in the network are shown in Figure 2.5.

The model starts with a network that is initially empty and continuously augmented and refined by dynamically updating the network with each new input example. Learning in the model proceeds in a bottom-up fashion, starting with representations of low complexity as sequences of words coupled with a predicate logic representation of their meaning. Linguistic structures of low complexity are gradually abstracted from specific words into partially productive slot and frame patterns that are lexically anchored to yield fully productive constructions when generalization is finished.

In sum, the model is characterized by the following properties:

1. Linguistic knowledge is represented in a network structure that contains linguistic constructions at various levels of abstraction as a result from an incremental data-driven generalization process.
2. The model does not assume linguistic knowledge to exist in the beginning.

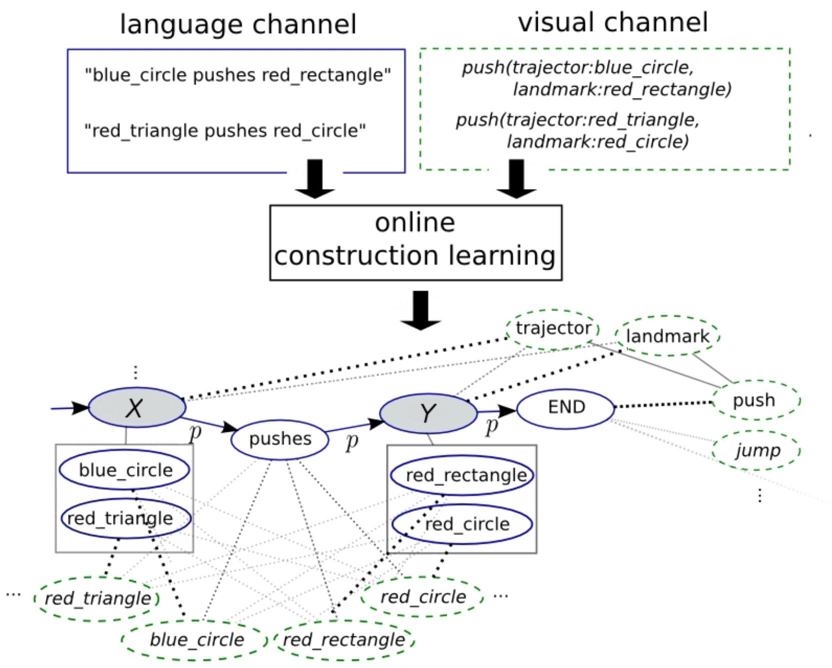

language channel
visual channel
"blue_circle pushes red_rectangle"
"red_triangle pushes red_circle"
push(trajector:blue_circle, landmark:red_rectangle)
push(trajector:red_triangle, landmark:red_circle)
online construction learning
trajector
landmark
X
p
pushes
p
Y
p
END
push
jump
blue_circle
red_triangle
red_rectangle
red_circle
red_triangle
blue_circle
red_rectangle
red_circle

Figure 2.5: Schematic overview showing an example construction learned from the two examples of the "pushes" action given as paired input across the language and visual channel. The figure shows the learned construction stored in the network.

3. Resulting representations in the model are suitable to both language understanding as well as generation of utterances.
4. The model learns *on-line* in an incremental fashion where each new item causes direct updates of the learned knowledge and does not need to be explicitly stored for later consideration.

Having these properties, the model complies with *emergentist* and *item-based* theories of early language acquisition and is thus compatible with the theoretical framework the multi-modal learning model is situated in. Properties required by the theoretical framework include the ability to learn *on-line* in a *bottom-up* fashion starting with representations of low complexity that are gradually abstracted into more general representations. The properties required to be compatible with the theoretical framework together with their technical implications are discussed in detail in section 3.2.

The learned linguistic knowledge in the model is stored in a network structure that evolves over time by continously updating its structure in the light of new data. The network can be roughly divided into a subnetwork representing lexical, and a subnetwork representing syntactic constructions, where the syntactic subnetwork builds on the lexical subnetwork and is divided into two sublayers: a slot-and-frame (S&F) pattern layer and a mapping layer. The **lexical subnetwork** encodes simple phrases, i.e. (short sequences of) words, as nodes together with the associated semantic referents, e.g. the sequence "red triangle" and the corresponding semantic referent *'red_triangle'* in Figure 2.5. The **S&F pattern layer** represents syntactic constructions as sequences of nodes that together constitute a path. Paths can contain variable nodes that represent slots in the syntactic pattern. These slots can be filled with elements contained in specific groupings. This layer also encodes the associated semantic frames. For instance, in Figure 2.5, a syntactic construction is represented as a path p which expresses a pattern "X pushes Y", where X and Y represent syntactic slots in the pattern, which can be filled with groupings of elements such as "blue circle" and "red triangle" in the case of X or "red circle" and "red rectangle" in the case of Y. The semantic frame associated with the pattern is *push(trajector,landmark)*. The **mapping layer** contains networks representing construction-specific argument mappings between syntactic patterns and semantic frames together with mappings of the syntactic arguments to semantic arguments. For example, in Figure 2.5 an individual mapping network captures the correspondences between X and the *trajector* role as well as Y and the *landmark* role.

In general, the model considers two types of constructions stored in the network that correspond to different degrees of generalization:

- Constructions at the word level where the construction corresponds to short sequences of words, e.g. "red triangle" is captured in the word construction network CON_{WORD}.
- Constructions of higher complexity at the level of slot and frame patterns, e.g. "X pushes Y" are stored in the slot-and-frame pattern construction network $CON_{S\&F}$.

Form-meaning mappings as correspondences between linguistic and conceptual entities, are established by capturing their co-occurrence frequencies across observed examples and contexts in associative networks.

The individual subnetworks and the representations they develop during learning are detailed in the following:

Lexical subnetwork The lexical subnetwork encodes simple phrases, i.e. (short sequences of) words along with their associated semantic referents as nodes in the network, i.e. the sequence "red triangle" and the corresponding semantic referent *'red_triangle'* in Figure 2.5. The underlying network is a left-to-right graph, where transitions from node n_i to node n_j are constrained such that there are no transitions with $j < i$. The lexical subnetwork thus directly encodes the order in which words appear in a sentence. The strict left-to-right constraint also ensures the graph is loop free and only sentences of finite length can be generated.

S&F pattern layer The S&F pattern layer represents syntactic constructions as sequences of nodes that together constitute a path. Paths can contain variable nodes that represent slots in the syntactic pattern. These slots can be filled with elements contained in specific groupings. This layer also encodes the associated semantic frames. For instance, in Figure 2.5, a syntactic construction is represented as a path p which expresses a pattern "X pushes Y", where X and Y represent syntactic slots in the pattern, which can be filled with groupings of elements such as "blue circle" and "red triangle" in the case of X or "red circle" and "red rectangle" in the case of Y. The semantic frame associated with the pattern is *push(trajector,landmark).*

Mapping layer The mapping layer contains networks representing construction-specific argument mappings between syntactic patterns and semantic frames together with mappings of the syntactic arguments to semantic arguments. For example, in Figure 2.5 an individual mapping network captures the correspondences between X and the *trajector* role as well as Y and the *landmark* role.

Co-occurence frequencies in hebbian associative networks

Correspondences between form and meaning are represented in associative networks as suggested by Rojas (2013) in a Hebbian learning (Hebb 1949) setting, where connections between simultaneously activated units are strengthened and thus capture the co-occurence between form and meaning. The graph consists of two bipartite layers x and y which are fully connected through a matrix W of learnable weights. Form and meaning are thus organized such that form is represented on one side of the graph and meaning on the other. Learned associations between both can be retrieved by

$$y = Wx \tag{2.50}$$

and

$$x = W^T y \tag{2.51}$$

respectively. Gaspers and Cimiano (2014) suggest to use an adjusted learning rule originally proposed by Schatten (2003):

$$\Delta w_{i,j} = \eta(x_i - x_i^{'})(y_i - y_j^{'}) \tag{2.52}$$

with x_i' and y_j' being the current value of x_i and y_j after processing the current input x and y. The update is mediated by the learning rate η.

In order to learn new mappings when they are observed in the data, the network has to grow over time, introducing new nodes for newly observed associations. When new nodes are introduced into the network new connections between those nodes and the already existing nodes have to be established and their initial weights have to be determined. Possible initialization strategies are for example to initialize new connections with a predetermined fixed initial weight or to calculate the initial weights based on the average of all weights. However, in the case of the associative network that captures the co-occurrence between lexical units $v_{nl} \in V_{NL}$, the authors proposed a different initialization strategy, that provides the model with the ability to quickly associate newly observed words with their meaning on the first occurrence. This *one-shot learning* ability is in the context of language learning often referred to as *fast mapping* observed in children (Susan Carey et al. 1978). The authors suggest an initialization strategy based on the intuition that new lexical units or referents should preferably be associated with referents or lexical units that in the current state of the network are not associated with other units.

All of the associations stored in the associative networks relate surface forms with their corresponding meaning.

In summary, associations are tracked to:

- Establish correspondences between lexical units and simple semantic referents
- Relate sets of slot-filling elements in syntactic patterns to argument slots in semantic frames
- Link semantic frames to syntactic patterns

Learning and Generalization

Learning is organized in an online fashion where each input example causes immediate changes in the network structure. The learning process is roughly divided into two steps: i) update of the lexical layer, where connections between lexical units and semantic referents are established and reinforced, and ii) update of the construction layer, where the model mainly attempts to merge paths, and thus generalizes over specific linguistic and action examples observed.

For generalization, the model exploits type variations at the linguistic level in relation to semantic observations. More specifically, there are two different generalization steps, both of which are applied to each observed input example, i.e. i) a slot-driven generalization step and ii) a syntactic generalization step. In the slot-driven generalization step, the model searches for sentences and (partially generalized) patterns for which linguistic variation in a position yields corresponding semantic variation in a slot in an associated semantic frame. In the syntactic generalization step, the model searches for patterns which show linguistic variation in a position but are associated with the same semantic frame. Thus, syntactic generalization may yield groupings of lexical units which are synonyms.

To illustrate the intuition behind the learning steps, consider the following example: A learner hears "the blue circle jumps" and "the red triangle jumps" in the visual context *jump(trajector:blue_circle)* and *jump(trajector:red_triangle)*, respectively. To learn across situations, during updates of the lexical layer, the model would use its knowledge that the linguistic phrase "red triangle" refers to the semantic entity *red_triangle* and that the phrase "blue circle" refers to the semantic entity *blue_circle*. Such knowledge would, in turn, be applied during updates of the construction layer in the slot-driven generalization step to bootstrap that the type variation in the sentences' first position ("blue circle" vs. "red triangle") reflects the meaning difference in the *trajector* role of *jump*. The model would use its knowledge to acquire the more general pattern shown below, where X = [blue circle $\rightarrow$ *blue_circle*, red triangle $\rightarrow$ *red_triangle*].

1.

Syntactic pattern	X jumps
Semantic frame	*jump(trajector)*
Mapping	$X \rightarrow$ *trajector*

Now let's assume that after observation of some more input examples the model has also acquired the constructions shown in (2), where again X = [red triangle $\rightarrow$ *red_triangle*, blue circle $\rightarrow$ *blue_circle*].

2.

Syntactic pattern	X hops
Semantic frame	*jump(trajector)*
Mapping	$X \rightarrow$ *trajector*

Since the two syntactic patterns show linguistic variation in one position ("jumps" vs. "hops"), but are associated with the same semantic frame, the model would group these two words and assume that both can be used interchangeably (without yielding semantic change). The model would thus use its knowledge to acquire the more general pattern shown in (3), where X = [red triangle $\rightarrow$ *red_triangle*, blue circle $\rightarrow$ *blue_circle*] and SYN_1 = [jumps, hops].

3.

Syntactic pattern	X SYN_1
Semantic frame	*jump(trajector)*
Mapping	$X \rightarrow$ *trajector*

2.7 Deep Q-learning

Reinforcement learning is often applied in learning scenarios where an agent has to learn how to efficiently achieve a certain goal by taking actions and observing the resulting changes in the environment in order to take the next action that brings the agent further to the goal it desires to reach. From time to time the agent receives a

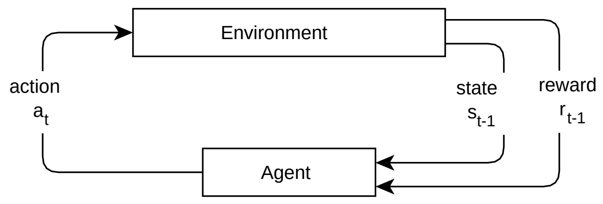


Figure 2.6: The interaction between the agent and the environment is modeled as a markov decision process, where the agent takes an action a_t at the current time t that changes the environment, which, in turn, generates a new state and a reward that is consecutively observed by the agent to estimate the next optimal action to take in order to maximize the cumulative reward over all successive iterations. Reproduced from Richard S Sutton et al. (2017).

so called reward, e.g. when a certain sub-goal is reached, that is usually only sparsely given as a result to a sequence of actions that lead to a sequence of changes in the environment.

The fundamental learning task is now to correlate immediate actions with the delayed rewards they produce. Many reinforcement algorithms have been proposed across several fields of research, including control theory, operations research, game theory and genetic-algorithms. In this section we introduce a variant of the well known Q-learning (Watkins et al. 1992) algorithm which builds on a deep feed-forward neural network to calculate the so called Q function which estimates the long-term return when taking action a in the current state s under the current policy π. The environment is usually modeled as a function that transforms an action taken in the current state into a set of observations describing the next state and a reward that constitutes the weak supervision for the learning process.

Reinforcement learning fundamentally builds on the concepts of agents, environments, states, actions and rewards (see Figure 2.6) which are described in the following:

Agent(A): The agent is the fundamental abstraction of the learner, which, in contrast to supervised learning, is not directly told the best action to take in the current state of the environment, but instead has to explore the state-action space to discover which actions achieve the most reward.

Action(a): To interact with the environment the agent can choose from a set of possible actions to take. Applying an action can result not only in an immediate reward, but can also have an effect on the reward received in sub-sequent iterations.

Environment(E): The environment is the world the agent interacts with. The environment can be modeled as a function that takes an action and the current state of the world and outputs the next state of the world together with the reward the agent receives.

State(s): A state encapsulates all information that comprise the immediate situation the agent is in with regard to the environment. The state can, e.g. contain information about the position of a mobile agent, or it's relation to some reference object. The state is usually not directly observable but instead manifests in a set of observations.

Reward(r): The reward is the feedback signal the agent receives as supervision. Reward can be a measure of success or failure or any other signal that allows the agent to draw conclusions on the utility of the action it has taken in the past. The reward is not required to reflect the immediate effect of an action taken but can rather be delayed or a consequence of a sequence of actions taken in order. The agents goal is to discover the sequence of actions that maximizes the cumulative reward the agent receives.

Policy(π): The policy is the strategy the agent follows to determine the next action to choose based on the current state the environment is in. The policy can usually be seen as a function that takes the current state as an argument and outputs the expected utility of taking any of the applicable actions. The policy is usually complemented with an exploration strategy that decides whether to apply the highest rated action or to explore the environment along paths (sequences of actions) that appear less favorable under the current learned policy.

Value(V): The value function $V_\pi(s)$ represents the expected long-term reward in state s under the current policy π, as opposed to the short-term immediate reward r. Reward expected in the future is usually discounted by a factor γ per iteration, giving precedence to recent over later rewards.

Q-Value (Q): The Q-value $Q(a, s)$ or action-value refers to the long-term reward that is expected when taking action a in current state s and following the optimal action according to the current policy π afterwards.

The environment the agent interacts with can be arbitrarily complex, for example a real physical robot that performs the actions chosen by the learner in the real world. In this example the environment would probably be highly non-deterministic, e.g. when the actions are motor voltages, no sequence of actions would be reproducible in a sense that it results in the same sequence of real-valued observations of torques and positions. While the current state of the environment can be observed, the environment itself is unknown and considered a black box. The task of the agent is now to attempt to approximate the function that describes the environment such that the agent can accurately predict the next state s_{t+1} of the environment together with the received reward t_{t+1} when taking action a_t at the current time t.

The classical Q-learning (Watkins et al. 1992) is a form of temporal difference learning (Richard S. Sutton 1988) and was one of the most popular reinforcement learning algorithms that is applicable to a variety of different learning problems. However the general applicability was limited to domains which are fully observable with low-dimensional input and state spaces or when a set of low-dimensional input observations could be handcrafted. Mnih, Kavukcuoglu, Silver, Rusu, et al. (2015) propose a variant of the classical Q-learning that overcomes this limitation by employing a deep neural network as an estimator to the Q-function. Neural networks with many layers are known to work well even in high-dimensional feature spaces and in many cases develop expressive features for highly structured data. These features are developed by transforming the data in each of the network's layers such that, with each layer, it progressively develops more and more abstract representations with regard to the learning task. To demonstrate the flexibility of the approach, the authors showed that their agent could learn to play 49 different Atari 2600 games on a human level of performance without relying on game specific features. To learn how to derive relevant features that accurately represent the current state of the game from raw pixel input, they employed a deep network with convolutional layers to learn domain specific filter banks. The approach, however, is not limited to convolutional networks and was successfully applied with many different neural networks architectures.

Reformulating the classical Q-learning approach with a neural network as Q-function estimator has the advantage that the state-space can be very large and is not required to be discrete. However, applying (deep) neural networks in reinforcement learning poses some challenges regarding the stability of the training target and the correlation of input observations. These challenges are approached by employing a *target network* that is separate from the currently learned live network and *experience replay* to decorrelate subsequent observations.

Target network

Neural networks are usually trained in a supervised manner. In reinforcement learning, however, the correct labels (long-term reward of taking a certain action in the current state) are usually unknown. In order to derive a training target Y_t (label) for the network, Q-learning builds on it's own estimation of the Q-function.

Let $Q(a, s, \theta)$ be a neural network that represents the Q-function with θ being the learnable weights of the network, the training target takes the following form:

$$Y_t = r_t + \gamma \max_a Q(s', a', \theta_i) \tag{2.53}$$

with r_t being the current reward at time t and a discount factor γ that prefers current over later rewards. The training target consists of r_t as the actual evidence of current reward and the networks own prediction of cumulated expected reward when taking action a in the current state and following the best available actions afterwards. Updating a neural network with its own predictions can lead to severe overestimation of state-action utilities. Mnih, Kavukcuoglu, Silver, Rusu, et al. (2015) propose to mitigate this effect by employing a separate target network θ' which is structurally equivalent to the online network except that its parameters are a periodic copy of the parameters of the online network every τ time-steps. The idea of employing a decoupled target network has further been extended by Hasselt (2010) based on the observation that the max operation in Equation 2.53 introduces a positive bias towards systematically overestimating the expected action value. The authors propose to decorrelate the selection of the best available action from the estimation of the expected cumulative reward. The adapted update target decomposes the max operation by using the online network θ_t for action selection and the target network θ'_i for action evaluation under a greedy policy:

$$Y_t = r_t + \gamma Q(s', \operatorname*{argmax}_a Q(s', a'; \theta_t), \theta'_i). \tag{2.54}$$

Experience replay

In supervised learning, input examples are often assumed to be i.i.d (independent and identically distributed). In reinforcement learning the learner is usually challenged with observations that are observed in successive states of the environment and thus highly correlated. Experience replay is a biologically inspired mechanism that randomizes over data and thereby removes correlations in consecutive observation sequences.

During the interaction with the environment, the agent repeatedly performs actions a_t while the environment is being in a certain state s_t. In reaction to the agent's actions the environment (in it's functional interpretation) transitions to it's next state s_{t+1} and outputs an immediate reward r_{t+1}, that can be the result of any sequence of past interactions. Instead of updating the Q-network after each iteration directly and thus learning form subsequent highly correlated samples, *experience replay* keeps a windowed replay buffer over the last N transitions and updates the network with a batch of randomly sampled transitions stored in the buffer.

Policy iteration and random exploration

Neural networks are usually employed in supervised learning problems where the correct output values are known for a given set of training instances. In deep Q-learning there are no explicit training instances to learn from, instead the agent has to rely on exploration to elicit feedback from the environment in terms of a rarely given reward. Monte Carlo methods that rely on repeated random sampling can be employed to train the estimator. The method is based on the intuition that when randomly exploring the environment for enough iterations, better state-action

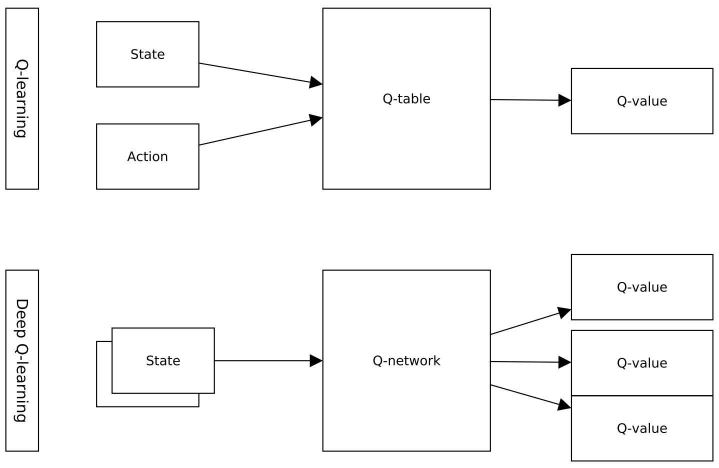


Figure 2.7: In the classical Q-learning formulation the action whose expected reward is to be calculated is given as a parameter of the Q-function. As this approach is not efficient when the Q-function is estimated by a computationally more demanding neural-network, the expected rewards of all actions are calculated at once in a single forward pass.

combinations yield higher rewards on average, due to the law of large numbers. This way the agent can extract some information about the expected value of a state without having any prior knowledge about the environment. To efficiently explore the environment, the agent has to find the optimal trade-off between exploration and exploitation of gained knowledge.

Efficient action value estimation

In order to efficiently model the Q-function as a neural network, the function has to be adapted. The classical Q-function represents the maximum discounted reward that is expected in successive iterations when taking action a_t in the current state s_t and continue with the best actions (receiving the highest awards) afterwards:

$$Q(s_t, a_t) = \max_{r_t,\cdots,r_T} r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^T r_T \tag{2.55}$$

Having the action as a parameter would require a separate forward pass of the neural network for each action. As this is not efficient, the Q-values for all possible actions are output all at once instead (see Figure 2.7). Another difference to the classical approach that relied on hand-engineered features to represent the information necessary to solve the learning problem is that when developing these features as part of the learning problem, some features can not be learned by a single snapshot of the current state. Velocities of objects for example can only be derived from consecutive states containing information about the current position of those objects.

3 Item-based induction of generalized qualitative action models

Movement is the only potential we have to interact with our environment. Even communication between humans through gestures, facial expressions or even speech is fundamentally mediated through body movements based on a common code of communication. This tight coupling between our ability to communicate and our ability to act in the environment suggests that the development of both abilities in infants does not happen in isolation but rather as a joint process.

This coupling does not only apply to the development of action skills and concepts but for conceptual development in general (Bowerman et al. 2001). The theory of linguistic relativism, more widely known as Sapir-Whorf hypothesis, claims that language structures thought and shapes the concepts we acquire (Phillip Wolff et al. 2011).

In contrast, however, many computational models and theories of language acquisition so far assumed that concepts are available prior to learning the meaning of a certain linguistic construction and that novel words or constructions are merely mapped onto these existing or innate concepts. This so called *mapping* paradigm is a large simplification when learning the grounded meaning of verbal constructions. Gleitman (1990) has argued that it is not likely "that learning in this regard is always and only a matter of mapping the words heard onto a preset and immutable set of concepts priorly available to the prelinguistic child. Rather, there is bound to be some degree of interaction between the categories lexicalized in a language and the child's conceptual organization; moreover, that conceptual organization is changing during the period of vocabulary growth, to some degree affecting the nature of lexical entries...".

In this thesis we propose a cross-modal model for the joint acquisition of action concepts and linguistic constructions that develops grounded representations of both modalities in an item-based manner without requiring innate categorical knowledge to exist. Without innate linguistic knowledge to overcome the possible semantic underspecification of language, the model has to rely on cross-modal learning to incorporate categorical learning cues from other modalities. In the model for multi modal learning (chapter 4) we focus on categories of goal directed actions that represent the meaning of action verbs. Actions in general can be defined as repeatable processes that cause change in the environment that is so important and meaningful that we assign a name to it.

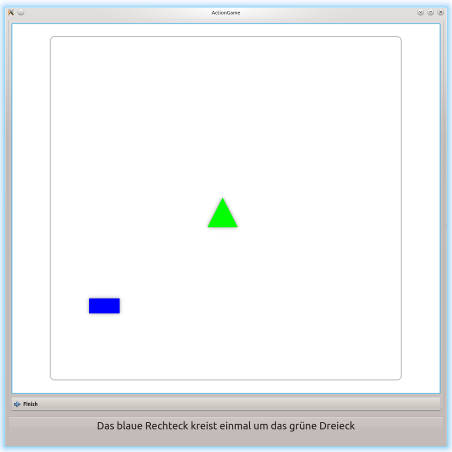
Goal directed action are defined by Huttenlocher et al. (1983) as follows:

> **Definition 1.** Goal directed actions are actions in which an object moves or changes state as a result of the actor's movement, where both the movement and the effects on the objects are observable

In this chapter we develop the component that is responsible for inducing generalized representations of action in the multi-modal learning model that is developed in the next chapter (see chapter 4). As the final model is designed to to be compatible with *emergentist* and *usage-based* theories of language acquisition, the components the model is composed of, are also required to satisfy the requirements that result from that theoretic foundation. Briefly put, *emergentist* and *usage-based* theories of language assume that language can be learned from language use by means of a generalization process that is powerful enough to drive the learning, embedded in a cultural context of other language users, without relying on innate categorical knowledge to exist. We discuss the requirements for the action model that result from this theoretical foundation in section 3.2. The relation between the multi-modal model, the modality-specific learning components it is composed of and the theoretical foundation that frames the general context is discussed in chapter 4.

The question how generalized categories can develop solely from experience is still subject to active research. One of the suggested mechanisms behind emergence is compositionality. More complex structure can be generated from simpler structure by means of composition. Combining simple structure into more complex concepts can also account for the ability of learners to go beyond what is specifically contained in experience. In computational machine learning this is termed generalization, referring to the ability of a learning system to extract the regularities and properties of an underlying process that is not directly specified in the examples the system learns from, as these processes are usually not directly observable. The examples can be considered as the surface form of what the assumed underlying process generates. Human cognition can be considered such an underlying process that generates effects that surface as a variety of different observations, that all manifest as recognizable changes in the environment. Communication manifests for example as sound waves caused by movements of our lungs and vocal cords or as gestures by moving our arms.

The remainder of this chapter is structured as follows: In the next section we introduce the learning problem and the corresponding dataset that is used in the evaluation of the developed action model. Then we investigate the requirements that result from the theoretical foundation of *emergentist* and *usage-based* theories of language and concept acquisition (section 3.2), followed by section 3.3 that describes the developed model technically. In section 3.4 we evaluate the model with regard to



Das blaue Rechteck kreist einmal um das grüne Dreieck

Figure 3.1: Simple game with two geometric objects which can be freely moved on the game field. In this screen, test subjects are asked to revolve the blue rectangle around the green triangle (instruction in the lower part of the screen given in german).

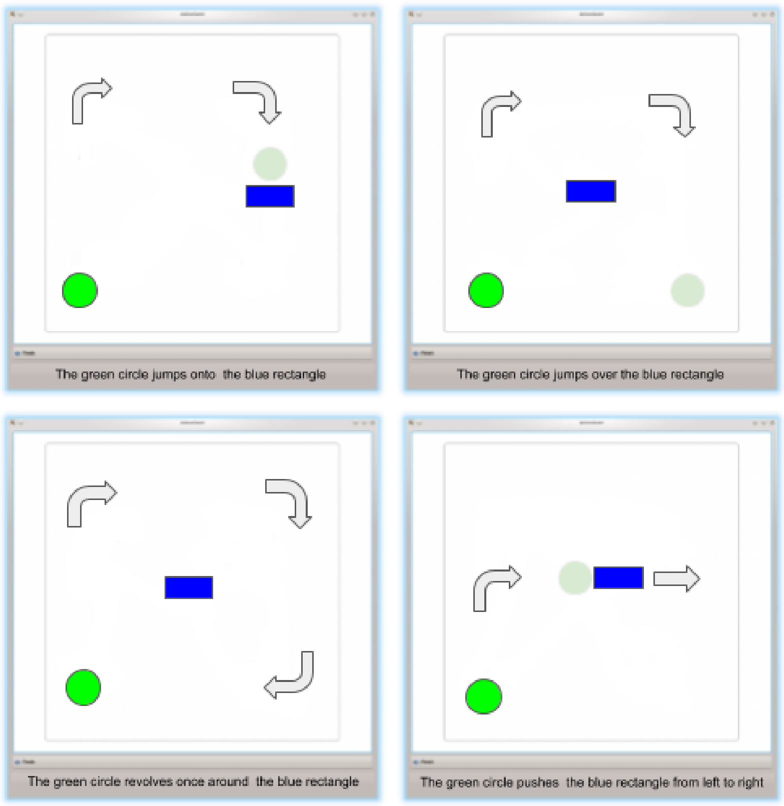
several aspects that are relevant for the multi-modal learning model as well as aspects that are relevant for a general application of the action model, e.g. in developmental robotics.

3.1 Learning problem and dataset

We consider a learning scenario in which the system learns from trajectories of objects corresponding to different actions coupled with utterances describing these actions. We considered the following four actions (see Figure 3.2):

- *jump onto* - The object should be placed on top of a reference object
- *jump over* - The object should move over the reference object
- *revolve around (once)* - The object should circle once around the reference object
- *pushes* - The object should push the other object

These actions were chosen because they can be performed easily in a 2D-scenario regardless of the types of objects involved and because they also provide some challenges regarding discriminability, e.g. instances of *jump onto* and *jump over* may have rather similar trajectories in the beginning of the sequence. Instances of the *jumps-over* action look similar to the first half of typical *circles-around* actions.


The green circle jumps onto the blue rectangle

The green circle jumps over the blue rectangle

The green circle revolves once around the blue rectangle

The green circle pushes the blue rectangle from left to right

Figure 3.2: This figure shows the *jumps onto*, *jumps over*, *circles around* and the *pushes* action along with prototypical trajectories that could be expected a-priori. Some of the trajectories in the dataset differ substantially from this expectation, e.g. when the objects where first arranged differently by the participants prior to performing the actual action.

To collect the data, we implemented a simple game in which participants could slide geometric objects on a screen (see Figure 3.1 for an example screen-shot). Participants where asked to play 100 game rounds, each corresponding to a unique combination of action and objects to perform the action with.

In each trial, an utterance expressing an action was displayed on the screen along with two objects named in the utterance, and subjects were asked to perform the action described by the displayed utterance accordingly by sliding the corresponding object(s). Each displayed utterance described one out of the four different actions. For each action a single syntactic pattern was used to generate utterances describing the action, with different combinations of the objects appearing in the syntactic slots of the pattern. For our experiments, the following four patterns were used:

- *trajector* jumps onto *landmark*

- *trajector* jumps over *landmark*
- *trajector* revolves once around *landmark*
- *trajector* pushes *landmark* from left to right

where *trajector* refers to the object that should be moved and *landmark* refers to the object that serves as a reference point for the movement or the object that is pushed by the object in the *trajector* role.

We considered a total of 9 objects for trajector and landmark. These objects comprise 3 geometric forms (rectangle, triangle circle) in three different colors (red, blue, green). For each of the four actions we generated 25 different utterances as instantiations of the patterns, for example "the red circle pushes the green triangle from left to right". In each trial the game recorded the positions of both objects at a fixed rate. We collected data from 12 subjects with a mean age of 29,4 years, 9 of which were male and 3 female, yielding 1200 input examples altogether.

The recorded trajectories are given as a sequence of tuples with the current timestamp and the positions of the two objects associated with the instruction sentence describing the action. The complete dataset is available for download[1].

The data is given as a collection of elements in XML format. The game was designed to output sequences formatted as predicate logic terms, as this is the native format for the original model that learns linguistic constructions (section 2.6). The first statement denotes the time the trial was recorded and the scenario configuration file which specifies the involved objects, their shapes, color and initial positions. The initial statement is followed by a sequence of *move* statements whose arguments denote: The temporal offset in milliseconds since the beginning of the recording, the concerned object and it's position given as x and y coordinates. The data format looks as follows:

```
<example id="1">
  <nl lang="de">
    Das rote Rechteck schubst den blauen Kreis von links nach rechts
  </nl>
  <mrl lang="gameout">
    trial(start="13:56:09", config="configs/scenario0.conf");
    move(13698, rectangle,[50:144]);
    ...
    move(18522, rectangle,[412:369]);
  </mrl>
 </example>
```

[1]https://pub.uni-bielefeld.de/download/2904362/2904363

3.2 Qualitative components of cross-modal language and concept learning

The multi-modal model for grounded language acquisition that is developed in the consecutive chapter is based on theories that describe incremental processes to the early acquisition of language without requiring innate categorical structure to be given from the beginning. The model comprises two components that are responsible for the modality specific induction of categorical representations of action and language. For the *item-based* acquisition of linguistic constructions, there is an existing model (see section 2.6) which is already compatible with the theoretical foundation and can be adapted to serve as the component for language learning (section section 4.2 gives details on how the model was adapted).

To the best of our knowledge, there is no existing model for action learning that complies with the requirements resulting from the theoretical foundation the multi-modal learning model is based on. In the following we categorize these requirements, identify challenges from a technical perspective and discuss solutions based on the frameworks of time-series modelling that was introduced starting with section 2.1.

Simulation

As actions are inherently dynamic rather than static concepts we understand the meaning of verbs as evoking a grounded simulation. Thus, the underlying model must facilitate dynamic simulations of prototypical actions of a certain category. One of the key principles underlying the embodied construction grammar (J. Feldman et al. 2009) formalism is that language understanding is a process that heavily relies on simulation (Bryant 2008) in that the same motor programs that are used to execute actions are also used to understand language. Simulation plays an important role in learning because it allows us to mentally test hypothesis without having to actually realize the necessary actions. This is a key advantage as with mental simulation we can repeatedly try and optimize the way we perform actions without actually changing the state of the environment in a way we cannot undo reliably. Having an accurate simulation model to train our abilities gives us a good starting point to experiment and optimize our behavior in the real environment, or put in the words of the theoretical physicist Richard Feynman: “What i cannot create, i do not understand”.

One class of supervised machine learning algorithms that particularly fits the simulation requirement are generative models. Generative models try to approximate an assumed underlying process that generated the observed data in contrast to a discriminative model that does not adhere to how the data was generated but only assigns the observations to their respective categories. There are two broad variants of generative models, those that take a probabilistic perspective and those that take a functional approach.

Probabilistic generative models like a Hidden Markov model are generative models in the sense, that they can be used to generate typical data according to the model (Danks 2014, p44) and learn a joint distribution $p(x, y)$ of the input data x and target values y. This is in contrast to a discriminative model, which attempts to model the distribution of target values conditional on the input observation $p(y|x)$.

Generative models that take a functional perspective, e.g. the Recurrent Neural Network often encode the progression of a sequence in a functional form, sometimes with a superimposed probabilistic interpretation (e.g. softmax normalization). Functional models have a different mathematical formulation $h_{n+1} = f(h_n, x_n, \theta)$ where h is a hidden latent state and f is usually a nonlinear function that is often referred to as the *non-linearity* (Joong et al. 2017).

There are also models that take both perspectives in a single model. Generative adversarial networks for example consist of two competing neural networks, the first of these networks generates examples according to its current generative model that the second model classifies according to its discriminative model (Goodfellow et al. 2014). The task of the generative network is to generate adversarial samples that the second network would falsely classify as typical samples that came from the real data distribution. The task for the second model is therefore to correctly distinguish real examples (from the training data) from adversarial samples that were generated by the generative network.

On-line incremental learning

As humans we often do not have exhaustive databases of training data to learn from, rather we have to iteratively gather experience by interacting with our environment. By interacting with the environment, we induce change that is observable and gives us hints as to how the underlying processes in the world are structured. From a cognitive learning perspective, on-line learning additionally means that the learner does not memorize the learning data explicitly but incorporates the essence of what can be learned from the new data into the model directly. The component for action learning should imitate this incremental learning ability by starting with very specific representations in the early learning phase that gradually develop into more complex categorical representations as learning continues with the growing number of items seen.

From a technical perspective, on-line learning refers to the ability of the learning system to adapt its behavior (e.g. predictions) immediately in response to new training data. Each new data item is directly used to update the model for future predictions as opposed to batch learning techniques which obtain their predictive model by training on the entire data set at once. A stricter variant of *on-line learning* is *incremental learning* (Gepperth et al. 2016) which adds the additional constraint that new data points are no longer available for consideration after the model has updated it's internal state. This constraint is for example not met by approaches like *k-nearest neighbors* which essentially work in batch mode for inference by explicitly storing all examples seen so far in memory[2].

Most of the state of the art algorithms for machine learning are designed to work on batches of training examples to estimate model parameters that best fit their learning target. This mode of training is computationally a large simplification that makes the learning problem easier and computationally more tractable.

Incremental learning on the other hand comes with a variety of algorithmical challenges. Learning instantaneously with every new data item requires the model to extract the essence of what can be learned from the training data given the training target and find a compact representation to represent this essence immediately in its model parameters. Further, in incremental learning scenarios the training data originates in a process that produces observation time-series that are not independent and identically distributed as required by many batch learning algorithms. As statistical processes can change their behavior over time, an effect termed *concept drift* can further complicate the learning process as later examples can stem from a process that behaves differently compared to the process that generated earlier observations. In batch learning it is possible to shuffle training examples such that examples recorded earlier alternate with later examples in contrast to an on-line learning algorithm which has to take measures to mitigate the effect of correlated subsequent samples.

Throughout this thesis we use the term *incremental learning* in the strict sense that also includes *on-line learning*, meaning that all new input examples cause direct updates of the predictive model and are not explicitly stored afterwards.

One-shot early learning

Goal directed actions are often learned in interaction with tutors. To support this mode of learning, the system has to quickly adapt to the tutoring in order to receive further training input. If the system was unable to learn from single tutoring examples it would be stuck with the same behavior, eliciting the ever same tutoring signals until the tutors eventually cease their efforts. To support this mode of learning, the system is thus required to adapt its behavior with each new training example instantaneously. In a technical context this is often referred to as *one-shot learning*. In cognitive psychology a similar concept is known as *fast mapping* (Susan Carey et al. 1978), a hypothesized mental process that allows a learner to acquire a novel category

[2]There are also extensions of the *k-nearest neighbors* algorithm that find the nearest neighbors with (model-based) approximate search, which do not need to store every single training example explicitly

after the first exposure to an informative sample. *Fast-mapping* is of particular importance to many theories of early language acquisition in children.

Supporting *one-shot* learning in technical systems is often approached in the learning algorithm itself or in the way data is represented, both of which are briefly discussed in the following.

In algorithmic learning, every single sample is important when coverage of a new domain is low (Stolcke et al. 1993). Slight adaptations, such as in many traditional algorithms (e.g. back propagation) make no good use of the first available examples and thus need many examples to converge. One of the reasons for this effect is that many of these traditional algorithms start with a model that is wrong initially and gradually improve with more data.

Furthermore these techniques usually have no representation of the strength of evidence supporting a certain conclusion. In early learning it can be better to first remember the evidence in detail rather than use the evidence to slightly modify model parameters. Furthermore, early in the learning process the learner has no experience about the importance and information content (uniqueness) of the individual samples unless the learner has good prior knowledge of the domain. Without prior knowledge, early generalization can only be driven based on the similarity of the samples.

Shepard (Shepard 1987) shows that this kind of similarity based generalization decays exponentially in animals as they accumulate experience. When the animal has enough data to validate models from more complex classes it does not need to remember individual experiences explicitly but rather can discover general rules as generalities in the data. This mode of learning allows a learner to develop more complex models while maintaining a measure of confidence in its predictions.

In technical systems, similar behavior is often implemented in terms of a *bias* that is either encoded in the fundamental assumptions of the model or given explicitly as a prior favoring certain model configurations over others. An example of the first can be seen in Markov random fields and Bayesian networks which encode sparse interaction patterns by exploiting conditional independence relations (a certain state is influenced only by limited set of predecessors). For example convolutional Neural Networks (CNN) that are mainly used in image recognition assume that interaction between pixels is limited to nearby pixels, because pixel values are affected by only a small number of localized objects. More examples can be found in physics, where many theories assume that objects are only affected by other objects that are close by.

In order to support *early-learning*, the action model should smoothly transition from the memory-based early learning behavior into a learning mode with more complex parameterized models that generalize also to yet unseen examples by exploiting commonalities in the data.

Technically, this behavior is implemented in the component for action learning in the so called *model merging* approach (see section 2.3), where complex models are constructed from combining and integrating simpler ones. Simpler models in this case are models that mainly memorize parts of the input data explicitly while more complex models usually abstract from concrete data.

Additionally, we improve training convergence by choosing a data representation scheme that is closer to the innate semantics of the learning problem compared to the raw input data. For example when the learning task is to classify if a given image depicts apples or bananas, a color histogram could be a better representation of input data (given the particular learning task) than raw pixels, because it focuses on color and abstracts away information about where the colored pixels occur in the image. Simplifying the learning task in such a way utilizes background knowledge (that can not necessarily be learned from the same data) of an expert to speed up the learning process. In the model for action learning we represent the relation between objects symbolically in the QTC (qualitative trajectory calculus) encoding scheme (see section 2.5) based on the assumption that basic perceptual capabilities are learned prior to learning complex goal directed actions.

Early sequence anticipation

Early sequence anticipation refers to the ability of a system to make predictions over time-series data early in the progression of the sequence. This is of great importance especially when the system is deployed in an interactive learning environment where it acts and gets new training stimuli in response. This ability is also related to the *predictive coding* (Clark 2013) theorem, a hypothesized framework to explain human cognition where the brain constantly follows and updates multiple hypothesis simultaneously at varying levels of abstraction.

Most classic time-series prediction algorithms extract features from the complete sequence (e.g. bag representations, such as bag of words) to make their prediction and are thus not directly applicable to classify incomplete or partial sequences. However, by explicitly training on incomplete sequences many algorithms can be also applied in this setting. This adaptation may require to alter the optimization target for those models to not only include the classification accuracy but also the earliness at which the sequence can be classified with satisfactory accuracy.

In the multi-modal model for action and language learning, *early sequence anticipation* is required to verbalize actions while they are currently being performed. The component for action should thus give correct and stable predictions about the category an ongoing action belongs to very early in the progression of that action. The foundation for this ability is the early formation of categories when only few training data is available as discussed in the previous section. Hidden Markov models are a suitable foundation for early sequence anticipation, as they can estimate the probability to generate any given sequence regardless of the completeness of the sequence.

Cross modal learning

Human cognition does not develop in isolation but rather in environmental and cultural interaction, both of which are inherently *cross-modal* modes of learning as they span multiple senses and modalities.

As the action learning component is part of a larger experiment to investigate cross modal co-emergence of categorical structure in both action and language, the action model is required to support *cross-modal* interactive learning, where action and linguistic structure co-develops by shaping and influencing each other.

From a technical perspective a model that facilitates cross-modal learning and simulation should either represent concepts from different modalities in the same representational structure or expose training signals that can be used by companion models that are responsible for the other modalities to drive their learning process. The first would be more favorable from a cognitive perspective as we as humans also represent concepts from different modalities in the same neural substrate. The second could be favorable from a computational perspective as it simplifies the learning problem for computational models that are less powerful than the human brain, because it omits the necessity to learn a transformation from modality specific data into a shared representational framework (e.g. neural activations and interconnection strengths).

3.3 Implementing the component for action learning

For each of the above defined requirements there are several well established models to achieve the desired properties. However, developing a model that exhibits all of the required properties is challenging.

The requirement to simulate actions within a given category suggests a generative model, which supports the simulation of prototypical actions and also the completion of sequences when only the beginning of the sequence is given. The family of generative models offer a variety of models that are suitable for action modeling in principle, however many of these models fall short in supporting all of the above requirements. Recurrent neural networks and generative adversarial networks for example usually require a large body of examples to converge, as they start with a model that is random initially and gradually improved with more training data. Many generative models can be incrementally trained or adapted accordingly, however, learning actionable representations (that allow to classify with adequate accuracy) from single training instances is very challenging for most models, especially in conjunction with *incremental learning*, which constrains the learning process to consider each training example at most once.

Another aspect related to *early sequence anticipation* is that a model that performs well in later learning phases, when enough training data is available, does not necessarily also perform well in the early phases with sparse data to learn from. In computational models this is reflected in the choice of the model itself and the hyperparameters that modify its properties. Hyperparameters of a model are those parameters that are set before training begins, in contrast to the model parameters that are optimized as part of the training procedure. The choice of a certain model can also be considered a hyperparameter of the overall learning problem. Choosing a model and selecting suitable hyperparameters can be automatized by performing random or exhaustive search over all possible hyperparameters to find the optimal model-parameter configuration. There are several approaches to perform this search, one is the so called *gridsearch* which samples different combinations of hyperparameters in pre-defined intervals. The drawback of this approach is that the model has to be re-trained with each new set of hyperparameters, which can quickly become computationally prohibitive. For some models there are more specialized approaches, e.g. LASSO (Tibshirani 1996) to pick regularization parameters for regression models, but for most models this boils down to performing uninformed (brute force) search, which requires the data to be explicitly memorized for *batch-mode* training and thus fails to comply with the requirement to learn incrementally.

A model that fulfills most of the requirements is the Hidden Markov model (see section 2.2). As generative models, HMMs support simulation in terms of generating prototypical realizations of learned action concepts. The graphical structure of the model allows *one-shot* learning by introducing paths specific for novel training examples when coverage of a given action category is low. As probabilistic sequential models they readily provide confidence values which can be used to orchestrate learning across modalities and can estimate the probability of generating sequences even when they are incomplete as required by *early sequence anticipation*. HMMs also provide means to adapt their structural properties to support *memory-based* early learning as well as learning generalized representations of action when more data is available. For example Hidden Markov models regress to the simpler Markov chains when emission densities are constrained to emit a single symbol, which can be exploited in early learning to limit the complexity of learned representations.

Traditionally, HMMs are trained using the Baum-Welsh (section 2.2.2) re-estimation method, which is an instance of expectation maximization training. Expectation maximization iteratively optimizes the parameters of the model in batches over the training data. However, on-line incremental learning in the strict definition requires that no examples other than the current one should be explicitly stored for the training process. Additionally, the early learning and classification requirements presuppose a different training scheme to yield good initial models even when only a few training examples are available.

The proposed solution to the challenge of incremental learning is a model training procedure that differs from traditional batch learning techniques in that it simulates the emergence of categorical representations as suggested by the theoretical foundation. A technique that fits this mode of learning is the model merging approach (see section 2.3), a family of algorithms that begin the modeling process with simple models that are specific to only a few initial training examples. More complex models that abstract from the underlying data are achieved by combining multiple simple models and collapsing similar substructure within the resulting model.

This approach solves the model structure selection and the optimization of model parameters in a joint optimization process as opposed to most traditional training schemes that optimize parameters given a fixed model structure in batch mode. From a cognitive perspective, this approach is more plausible as the amount of brain capacity assigned to individual learning problems is likewise not fixed but is rather subject to a global optimization process (e.g. when one sense is impaired the brain assigns more capacity to processing signals from the unimpaired ones).

In the following we develop the incrementally learned graphical model for action learning that is the foundation of the action learning component in the multi-modal model for grounded language acquisition. First we define the model in terms of its architecture and the incremental procedure used to develop generalized models from specific ones. Followed by a brief introduction to the implementation of the model merging approach that realizes the emergence of categorical representations starting with very specific models, representing individual examples that are gradually merged into more general models. This approach is contrasted with a baseline implementation that follows the traditional expectation maximization learning scheme and briefly evaluated in terms of the required properties.

3.3.1 Data model and preprocessing

The model learns from sequential data of object positions sampled at a fixed rate along with a label that represents the class of action the participant was asked to perform (see section 3.1). Object positions are given in terms of the x and y coordinate of the objects center of mass, yielding a sequence of positions alternating between the two objects (x^1 and x^2) involved.

$$p_i = \dots \begin{bmatrix} x_t^1 \\ y_t^1 \end{bmatrix}, \begin{bmatrix} x_t^2 \\ y_t^2 \end{bmatrix}, \begin{bmatrix} x_{t+1}^1 \\ y_{t+1}^1 \end{bmatrix}, \begin{bmatrix} x_{t+1}^2 \\ y_{t+1}^2 \end{bmatrix} \dots, \; p_i \in \mathcal{D} \tag{3.1}$$

In a preprocessing step the sequences of coordinates are transformed into a sequence of discrete qualitative relations between the objects following the QTC encoding scheme (see section 2.5). In order to calculate these relations we first have to reconstruct the velocity vectors as the difference of two subsequent positions. The velocity vectors are used in QTC in order to determine whether the objects approach or distance from each other and whether this happens to the left or to the right of the reference line between their centers. The velocity vector for object j is reconstructed as:

$$v_t^j = \begin{bmatrix} x_{t+1}^j \\ y_{t+1}^j \end{bmatrix} - \begin{bmatrix} x_t^j \\ y_t^j \end{bmatrix} \tag{3.2}$$

The corner case at $t = t_n$ with $t \in 1, \ldots, n$ where no consequent position at $t = t_{n+1}$ exists is treated by duplicating the velocity from $t = t_n$.

From the positions coupled with the reconstructed velocities of two objects, the corresponding QTC_{C1} relations can be calculated as follows:

The side constraints in the QTC framework rely on the indication to which side of the reference line RL an object will have moved to in the consecutive time-step (see Figure 3.3). The consecutive position at a_{t+1} of object a is calculated from the current position a_t and the velocity v_t^a. The reference line is spanned between the positions of two objects a and b.

To decide whether the subsequent position of the object is on the left or on the right of the reference line RL we use a property of the cross-product between two points. The cross-product is positive iff the angle of the points and the origin is counter-clockwise, or stated differently if the reference line has to be turned counter-clockwise in order to align all three points on the line. We term the two object positions at time t that form the reference line as a_t and b_t. The position of object a at the subsequent time-step is given as a_{t+1}. To apply the cross product we translate the coordinates such that a_t lies at the origin $(0, 0)$ and calculate the cross product.

$$d = (x_{t+1}^a - x_t^a) \times (y_t^b - y_t^a) - (y_{t+1}^a - y_t^a) \times (x_t^b - x_t^a) \tag{3.3}$$

The sign of d indicates whether RL has to be turned counter-clockwise ($d < 0$) or clockwise ($d > 0$) in order to align with a_{t+1}. We do not explicitly test for colinearity because float math is subject to rounding issues and having $+0$ or -0 as a result of real valued calculations is highly improbable.

QTC comes with two numeric interpretations, the state descriptor and a numeric identifier derived by interpreting the QTC descriptors as a number in a ternary number system. If not specifically mentioned otherwise, we use the interpretation as symbolic identifiers for experiments.

Compression of sub-sequences

As QTC_{C1} is a rather coarse discretization, the associated discretization error is considerable, leading to sub-sequences in the data where similar continuous values all map to the same QTC symbol. This results in sub-sequences where the qualitative relation between the two objects can hold for a longer portion of the trajectory

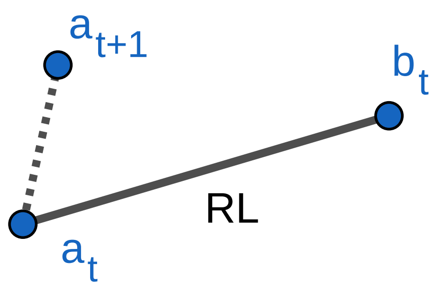


Figure 3.3: To decide whether a moves to the left of the reference line RL we calculate the subsequent position a_{t+1}, translate the coordinates such that a_t lies at the origin and calculate the cross-product between a_{t+1} and b (whose position vectors now base at the origin (0,0)) and use the property that the cross-product is positive iff a_{t+1} lies on the left of the reference line.

and is, due to the fixed rate sampling, repeated many times. In the case of goal directed actions, we assume that much of the goal semantics could be represented as pre- and post-conditions that mainly manifest in the beginning and the end of a sequence. Following this assumption, reducing the overall length of action sequences by reducing the number of recurring symbols could improve the performance of the model. This assumption is verified in the experiment in section 3.4.2.

Unlike many spatial reasoning systems where repeating states are simply omitted, we use a logarithmic compression of repetitive sub-sequences:

$$|\hat{s}| = \min(|s|, 10\ln(|s|+1)) \tag{3.4}$$

where $|s|$ is the original number of repeated symbols in the sequence and $|\hat{s}|$ is the resulting number of repeated symbols. The logarithmic compression of repetitive symbols in a sequence preserves some information about the relative speed of movements and is in line with findings from psychophysics known as the Weber-Fechner law (Bruss et al. 2010).

The learning problem for a model specific to action category C can now be formally stated as follows: Given a set of sequences S_C as examples of category specific QTC relations representing action trajectories from the dataset $\mathcal{D}$, the task is to estimate the model parameters $\lambda_C = (A, B, \pi)$ of the class specific HMMs.

3.3.2 Model definition

The component for action learning implements a usage-based approach to learning actions in the sense that the approach starts with simple models that are very specific for a limited set of actions and is incrementally generalized, driven by the objective to yield compact representations while trying to maintain most of the predictive accuracy. The preference for simpler models corresponds to Occam's razor principle and is implemented as a prior that prefers simpler models. Maintaining the predictive accuracy is implemented through a model merging procedure (see section 2.3) that merges specific models guided by the desire to yield generalizing models while at the same time maximizing the likelihood of generating the observed actions sequences under the generalized model in order to avoid over-generalization. The relations between objects involved in the modeled actions are represented qualitatively, corresponding to the QTC_{C_1} encoding scheme of the qualitative trajectory calculus (QTC), which was introduced in section 2.5.

Model merging for HMMs can be formulated by means of basic operators that work on the graphical structure of the model. As these operators introduce, modify or evaluate structure in the model they are interrelated with the choice of the HMM's structural parameters, which are reflected in the way emission and transitions are represented[3].

Model structure

The Hidden Markov models (see section 2.2) that form the foundation of this approach represent a stochastic process with an assumed underlying structure of discrete hidden states the process alternates between. The hidden states (nodes) are connected to each other forming an undirected graph of nodes. Each node in the model represents a sub-state of the statistical process the model approximates. These sub-processes are usually implemented as probability distributions.

The model is free to alternate between the nodes governed by the transition probability distributions that constitute the probability to transition to any given state s_j while currently being in state s_i. When being in a given state the model emits or accepts observations with a certain probability, governed by the emission probability distributions. During training, the learning procedure collapses similar structure within the model into more general and compact structure.

Collapsing structure in HMMs primarily means joining similar states into a new joint state that represents transition and emission characteristics from both ancestor states. This operation relies on the transition and emission representations to be mergeable as well. Furthermore, both densities should define a similarity measure that is used in discovering similar structure.

[3]When learning HMMs through model merging, the number of hidden states of the model is not a structural parameter as it is determined as a byproduct of the model induction process

Transition densities The transition probabilities are encoded in the transition matrix A (see subsection 2.2.1). In the traditional formulation, the transition matrix is unrestricted except for the requirement that it has to be row stochastic, such that each row constitutes a proper probability distribution. Imposing constraints on the transitions introduces a bias towards different model structures, e.g. assuming transitions from state $s_i \rightarrow s_j,\ i < j$ to be a-priori more probable than other transitions introduces a bias towards the so called left-to-right HMMs often applied in speech recognition. Although this seems favorable for action learning, as actions, similar to speech, strictly progress forward in time, we leave the transitions unconstrained as those constraints would have to be justified under the proposition that the model relates to *emergentist* theories of language acquisition.

Transitions are represented in terms of frequency histograms of observed transition. For each state $s_i \in S$ in the model there is a dedicated transition histogram representing transition from s_i to all other states $s_j \in S$ including self transitions. For similarity computation and for inference, the histogram is normalized to obtain a probabilistic interpretation. Merging of two transition histograms is implemented as simple addition of the underlying transition frequencies.

Emission densities The probabilities to emit or accept a certain observation (or symbol in the discrete case) are encoded in the observation probability matrix B (see subsection 2.2.1). The observations in the model are given as symbolic representations describing the relation between two objects in the qualitative trajectory calculus. In principle every discrete probability distribution for which a merge operation can be derived is applicable for a model merging based training procedure. However, QTC defines no immediate distance relationship between relation descriptors from which sufficient statistics like mean and variance could be derived. As QTC behaves more like categorical symbols, we represent the observations densities as non-parametric categorical distributions, which essentially correspond to histograms that are normalized to sum to one.

An important side effect of the categorical distributions is that they can easily assign all of their probability mass to a single symbol, and thus effectively turn hidden states of the model with those single symbol emissions into visible states, as there is effectively no hidden process involved that could generate other symbols. This is an important property for the early phases of learning because it allows the model to switch between simpler and more complex representations within the same model.

3.3.3 Item-based learning and generalization

The learning procedure for the action model differs from the traditional expectation maximization in that it operates in a strictly incremental fashion, updating the state of the model with each new data item received. The learning procedure

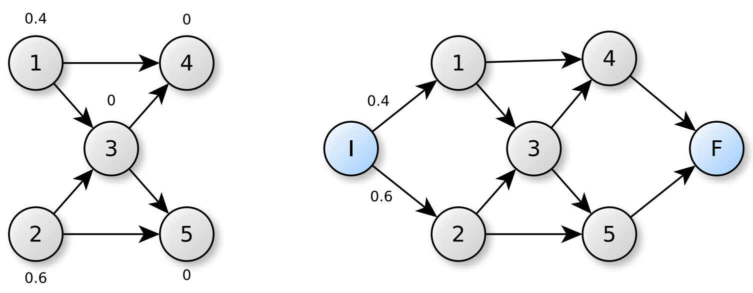


Figure 3.4: Introducing a virtual node for the initial and final state of the model simplifies expressing the model merging operations as pure graph operations. In this example the probabilities to begin in a certain state $\pi = \{0.4, 0.6, 0, 0, 0\}$ is indicated on the nodes in the figure on the left. With the introduction of virtual nodes (blue nodes) these probabilities become transitions from the initial node I.

is implemented adopting the model merging approach for HMMs with bayesian optimization (Stolcke et al. 1993). Model merging for HMMs can be interpreted as a search through the space of possible models to find the optimal model to represent the available data. This search is governed by a generalization strategy that decides which model variations to explore and an optimality criterion that estimates the goodness of any found solution.

To simplify the graphical interpretation of HMMs we take a slightly different perspective on the graphical structure of the Hidden Markov Models. In the original formulation, the initial state probabilities are given in terms of the $\pi = \{\pi_1, \ldots, \pi_n\}$ array (see subsection 2.2.1) which encodes the probability to begin a sequence in a certain state. Differing from this perspective we introduce virtual initial and final states and encode the probabilities to start the sequence in a certain state as transition probabilities when leaving the initial state (see Figure 3.4).

The model learns from labeled tuples (x, l) where x denotes the sequence of QTC relations that represent a single action trajectory and l denotes the category of action (target label) the sequence was observed for. The label information is used to train the HMM specific to the action category denoted by l.

Inducing generalized models from data proceeds in two phases:

Memorization phase Construct a maximally specific model for a newly received data item. In case of the action models, this is a Markov chain where each state in the chain represents (emits/accepts) a single symbol in the input sequence. The constructed chain is inserted into the existing (possible empty) model between the initial and final state. Memorization monotonically adds to the model size, introducing paths that are maximally specific for a single item and thus are unable to generalize. More specifically, a model that consists only of specific paths that generate all of the observed sequences, but no other than those that were observed.

Generalization phase To abstract from concrete observed sequences, similar structure within the model is collapsed into more compact representations that abstract from the concrete data. The generalization phase begins with computing a set of candidate merges C among all states of the model M_i in the current generalization iteration i. For each candidate merge $c \in C$, the merged model $c(M_i)$ and its posterior probability is calculated and the best merge c^* is selected. The generalization process enters the next iteration with the new model $M_{i+1} = c^*(M_i)$ until a stopping criterion is reached.

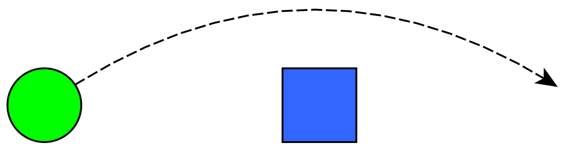
The process alternates between those two phases until all data items are processed. In the following we describe both phases in detail.

Memorization phase

Learning in the action model starts with integrating a new training sequence into the existing model. If no model exists for the category given by the labelled sample, a new HMM is created that is empty except for the virtual initial and final states.

Integration of new data starts by first obtaining a Markov Chain that is maximally specific for the new data sequence. Given a sample sequence $x = x_1, \ldots, x_l \in X$ of length l we can construct a sequence of states $q_1, \ldots, q_l$ with a dedicated state for each symbol in x. All new states are chained with probability 1, such that $P(q_i \rightarrow q_{i+1}) = 1; 1 \leq i < l$. To integrate the newly constructed Markov chain into the model we introduce a new transition from the initial state q_I to the first state in the sequence q_1 and a transition from the last state in the sequence q_l to the final state in the model q_T with probability 1. As all of the probabilities in the initial model are 1, the model constitutes a maximum likelihood model for the sequence x. If no model exists for the given category a new model is created that is empty, except for the virtual initial and final states. If a new sample is already represented by an existing path in the model we simply reuse the existing path and update the transition probability from the initial state accordingly.

For example consider the following sequence of a circle jumping over a rectangle:



The trajectory is given as a time-indexed sequence of coordinates for both objects.

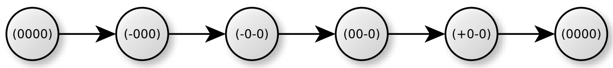
```
(1, green_circle, [10,5]); (1, blue_rectangle, [150,5]);
(2, green_circle, [15,5]); (2, blue_rectangle, [150,5]);
....
(1053, green_circle, [190,5]); (1, blue_rectangle, [150,5]);
```

The raw coordinates are first translated into a sequence of QTC symbols according to the preprocessing procedure described in subsection 3.3.1:

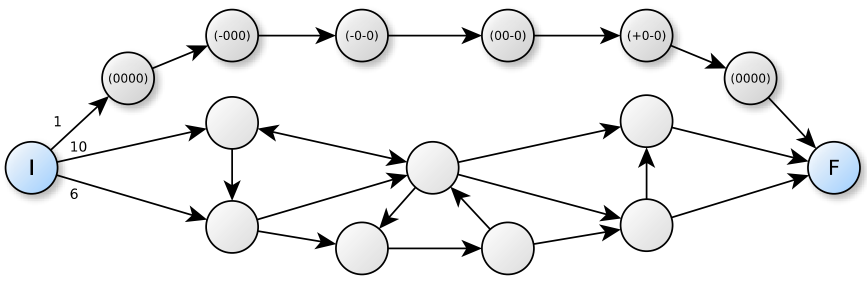
```
 (0000) (-000) (-0-0) (00-0) (+0-0) (0000)
```

The QTC symbols correspond to the following interpretation: First the circle moves towards the rectangle, then it moves to the left of the reference line between the two objects until it is on top of the rectangle. From there, it moves away from the rectangle on the left until it reaches its final position and stops. The QTC relations corresponding to the rectangle are 0 during the progression of the sequence, since the rectangle did not move.

The sequence of QTC symbols is translated into an equivalent Markov Chain. Markov chains differ from HMMs in that their states directly represent *visible* observations in contrast to the hidden states in the HMMs which correspond to a *hidden* statistical process, that is not directly observable but generates visible observations. The resulting Markov Chain



is then integrated into an existing (already partly generalized) model. The integration is done by introducing a new transition from the initial state to the first state in the new sequence. The last state of the new sequence is attached to the final state of the model accordingly.



For each transition we explicitly record how often this particular transition was observed in data as their statistical support and update the transition probabilities accordingly. For newly integrated paths the support is always 1, generalized paths that where obtained by collapsing structure accumulate their support values.

Generalization phase

The generalization phase proceeds in alternation with the memorization phase. While memorization adds new knowledge to the model as paths that are specific to a single sequence, in the generalization phase the systems abstracts from those specific sequences by collapsing similar structure into more general representations that abstract from specific sequences.

Generalization is implemented as an incremental process that identifies similar structure as pairs of states in the model that are similar with respect to their emission and transition characteristics. In each turn the generalization strategy identifies a set of candidate state pairs to be merged and ranks them according to their similarity. The strategy picks the most promising candidate pair and replaces them with a new, merged state that represents the emission and transition characteristics from both ancestor states. The merging process continues until a given stopping criterion is reached.

On the one hand, merging states distributes probability mass and thus establishes novel transitions between paths in the model that had zero probability before (see section 2.3), enabling the model to generate or accept sequences that were not explicitly observed during training. On the other hand the merging procedure also reduces the capacity of the model as with each merge operation the model has one less state to represent knowledge in. This approach is based on the framework of bayesian model merging that was introduced in section 2.3.

As in most generalization processes, it is essential to control the degree of generalization to yield an optimal model. Too general models tend to under-fit the data, e.g. always predicting the majority class without considering actual evidence. On the other extreme, over-fitting models often result from too little generalization, e.g. memorizing training instances without inducing abstract representations.

In the framework of bayesian model merging, the degree of generalization is mainly controlled by:

- The posterior, composed of a prior that assesses the current model *a-priori* (before actual data is seen as evidence) and the likelihood of the observed data under the model.
- The generalization strategy that governs the search for the optimal model and most importantly determines when the optimal degree of generalization is reached to break from the incremental generalization process.

For the likelihood calculation we follow (Omohundro 1993; Stolcke et al. 1993) and employ an efficient approximation that considers only the paths with the highest contributions and assumes that the so called viterbi paths are largely preserved during the state merging process.

Model prior

Generalization is closely tied to the notion of complexity, as learning can be interpreted as a process of incrementally reducing complexity by abstracting from data. The amount of generalization that occurs during merging can be measured as a drop in likelihood relative to the training data. The drop in likelihood is caused by the model distributing probability mass to similar sequences that are not explicitly represented in the training data. In turn, the merging process can be interpreted as trading off likelihood against a bias towards simpler, less complex models. Simplicity could for example be reflected in the resulting models having a lower number of states.

The key issue in controlling the amount of wanted generalization is to specify the optimal relation between the preference towards less complex models and the likelihood of the model representing the training data.

This trade-off can elegantly be expressed in terms of a bayesian posterior[4]:

$$P(M|X) \propto P(M)P(X|M) \tag{3.5}$$

composed of the prior $P(M)$ and the likelihood $P(X|M)$ terms. The likelihood can be efficiently calculated according to the approximation introduced in section section 2.3. The prior is not defined by the employed model merging approach and can be chosen according to the domain.

In bayesian statistics the prior is defined as one's belief about a certain quantity before actually having evidence to base an informed decision on. Priors to govern the generalization process in model merging can be for example based on measures that describe the structure of the model or on measures that additionally incorporate model parameters. The prior should be chosen such that it does not introduce any systematic bias towards a particular model as this would affect the learning process.

In line with the description length principle that defines simplicity as an attempt to yield a good compression for conceptual knowledge, the action model is based on a prior that reflects this concept in that it explicitly states its preference towards simpler models in general:

$$P(M) = e^{-|M|} \tag{3.6}$$

with $|M|$ being the number of states and thus the capacity of the current model M. In this perspective simplicity essentially relates to compressibility and therefore simple concepts likely result from high compression of the corresponding representations in the model.

While the model prior favors simpler models, its antagonist, the model likelihood, has its maximum at the initial unmerged model which consists only of maximally

[4]Since the data is fixed ($P(X) = 1$), the posterior can be defined as the product of the prior and the likelihood, the normalization is not needed

specific paths. As both measures are biased approximations, the pareto-optimal solution is not necessarily the solution that maximizes the model's accuracy.

To further improve the controllability of the generalization procedure we introduce the parameter λ to balance between the effects of prior and likelihood. With the balancing parameter added, the posterior becomes a model score that is defined as:

$$score(M) = \lambda P(M) + (1 - \lambda) P(X|M) \tag{3.7}$$

We evaluate the effect of the λ parameter on the model performance in subsection 3.4.3.

Generalization strategy

During generalization, each merging step attempts to maximize the model's posterior $P(M|X)$ (or the model score respectively), iterating until the optimal model is found, i.e. when no combination of remaining possible merges yields a model with higher posterior than the current one. The endeavor to find the best possible model is driven by a generalization strategy that attempts to perform the search efficiently.

The originally proposed *best-first* generalization strategy is a greedy strategy in that in each iteration it explores the merge with the highest estimated utility and continues to merge further states until the posterior of the resulting model with respect to the data no longer increases ($P(M_{i+1}|X) \leq P(M_i|X)$). A loss in posterior essentially corresponds to over-generalization in the model as with each merging step the available capacity of the model is reduced and the model is forced to discover more abstract representations in order to represent the contained knowledge with less capacity.

Greedy strategies can easily get stuck in local optima where the only potential to further improve the found solution is to accept less optimal solutions for a few iterations in order to find a better local or global optimum.

We extend the greedy heuristic with a look-ahead strategy that continues the merging process even when the posterior of the resulting models degrades at first. This effect is also present in numerical optimization of e.g. neural networks using plain gradient descent. The proposed lookahead strategy is similar to adding momentum in gradient descent based optimization.

We further extend the strategy to explore up to three merge paths in parallel, given by the top three merge candidates, until the first path results in a better model which is then accepted as the new model. The final strategy is formalized in algorithm 1.

```
Data: current model $M$
current best model $M_{best} = M$
current best model score $S_{best} = P(X|M)$
candidates: $C \subset (|M|x|M|) \setminus \{(s_1, s_2)|s_1 = s_2\}$
$C \leftarrow retieve\_three\_candidates(M_{best})$
for $c \in C$ do
    $\hat{M} \leftarrow$ new model with $(s_1, s_2)$ merged
    $S_{\hat{M}} = P(\hat{M}|X)$
    if $S_{\hat{M}} \leq S_{max}$ then
        $\hat{M}$ = lookahead($\hat{M}$); recurse for another merge
        $S_{\hat{M}} = P(X|\hat{M})$
    end
    if $S_{\hat{M}} > S_{max}$ then
        $M_{best} := \hat{M}$; current best model
        $S_{best} := P(X|\hat{M})$; current best score
    end
end
return $M_{best}$
```

Algorithm 1: State merging algorithm. Initialize the resulting model (M_{best}) with the current model and set the score for M_{best} to the score of the current model. Generate a list of candidate state tuples (s_1, s_2) to merge according to the similarity of their emission and transition densities. Construct a new model $\hat{M}$ from M with states (s_1, s_2) merged and check if the model scores higher than the current best model according to Equation 3.7. If the model scores higher it is remembered as the current best model, if not, the algorithm tries one further merge as *lookahead*.

Computing merge candidates

A model with $|M|$ states has $\frac{1}{2}|M|(|M| - 1)$ pairs of states that could potentially be merged. The almost quadratic cost in the number of states suggests a strategy that efficiently computes a list of candidate merges to try. In the component for action modeling the strategy is implemented as follows:

During early merging we only consider pairs of states with simple emission distributions that assign all of their probability mass to a single symbol. These states likely correspond to states that are part of newly introduced Markov chains. If no states with simple emission characteristics are left, the candidate retrieval begins to suggest pairs of states that are similar according to their emission distributions. Most of the computational complexity for computing candidate merges comes from the computation of the similarity between emission densities, which, in our case is

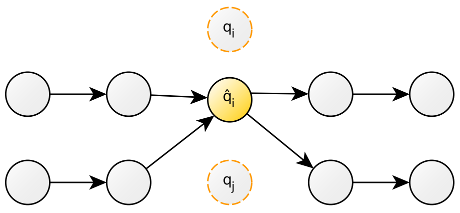


Figure 3.5: When two states q_i and q_j are merged, a new state $\hat{q}_i$ is recruited as a replacement that inherits the emission and transition characteristics from both of its ancestor states. Incoming transitions have to be re-routed to target $\hat{q}_i$ and outgoing transitions are adjusted such that they begin in $\hat{q}_i$.

calculated as the symmetrized[5] Kullback-Leibler divergence.

Preferring states with the same (single) output does not only speed up the algorithm as we do not need to compute the similarity between distributions, it also helps in preventing premature merges that would likely be assessed differently in the light of new data.

Merging states in Hidden Markov models

Merging two states q_i and q_j requires two independent operations. First, a new state has to be introduced as a replacement for the merged states and the associated transitions have to be consolidated. Second, the emission densities associated with those two states have to be merged as well. We detail the process in the following.

The merge operation starts with recruiting a new state $\hat{q}_i$ that replaces q_i and inherits its incoming and outgoing transitions. The incoming transitions to q_j are re-routed to target $\hat{q}_i$ and also the outgoing transitions from q_j have to be adjusted to begin in $\hat{q}_i$ (Figure 3.5 shows an example of this process). The result of the operation is a new state which inherits all incoming and outgoing transitions from the merged states and thus connects paths in the model that previously could have been independent.

To conclude the process, both emission densities associated with the merged states have to be consolidated into a single density for $\hat{q}_i$. The categorical distributions we employ in this component correspond to histogram of counts $c_1, \ldots, c_n$ which counts how often each of the possible QTC symbols have been observed in the constituting data. These counts are only normalized on demand, when samples are drawn from the distribution, which simplifies the merging of these two distributions to simply adding observation counts for each of the possible QTC symbols (see Figure 3.6).

[5]The Kullback-Leibler divergence $D(\cdot)$ is not symmetric, e.g. let P, Q be probability distributions $D(P||Q) \neq D(Q||P)$. The divergence can be trivially symmetrized by taking the sum $D(P||Q) + D(Q||P)$. While being computationally expensive we found that the symmetrized variant reduces the variance of resulting models especially in experiments with little training date.

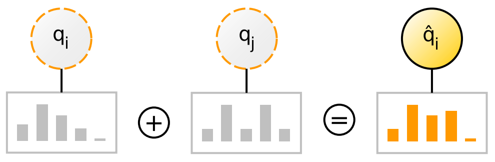


Figure 3.6: When two states q_i and q_j are merged, the probability distributions that govern the symbol emissions are merged too. Merging of categorical distributions corresponds to simply adding frequencies that are recorded for each observed symbol in the underlying data.

Implementation considerations

Multiplying many very small probabilities is numerically unstable on machines with limited numerical precision, as rounding errors accumulate with each operation. There is a variety of approaches to tackle problems associated with limited floating point precision, such as fixed point math or log probabilities. If not mentioned specifically we employ log-space math for all applicable calculations. In case of the forward-backward algorithm, log-space math is not applicable as the calculation relies on the addition of probabilities, which has no corresponding operation in log space. We employ a rescaling strategy as suggested in Rabiner et al. (1993) for forward-backward calculation. We calculate forward probabilities in the baseline model trained with EM and the proposed incremental model based on the assumption that in the physical domain there are no events that are impossible and thus have a probability of exactly zero. This is achieved by having virtual pseudo-counts in emission and transition densities which are effective when drawing samples from the distribution, but not for similarity computation or the compilation of merge candidates.

This approach relates to Cromwell's rule, named by statistician Dennis Lindley (Jackman 2009), which states that prior probabilities of zero or one should be avoided unless when applied to statements that can logically be assessed as being either true or false, e.g. in math.

3.4 Experimental evaluation

In the following we analyze the properties of the proposed model for action learning with respect to the requirements on the model as discussed in section 3.2. The evaluation begins with an experiment that compares the proposed incremental learning approach with a baseline model that conforms to the proposed model except that it is trained traditionally with the Baum-Welch algorithm as detailed in section 2.2.2. We show that although learning in an incremental fashion is conceptually more challenging, the results of the proposed model on a classification task are comparable to the traditionally trained baseline model.

The subsequent experiments investigate the impact of the compression scheme (section 3.3.1) for repetitive sub-sequences and the sensitivity of the proposed model to the λ parameter that balances the propensity to yield more compact models encoded in the prior that penalizes complexity against the likelihood that seeks to accurately represent the constituting training data.

We conclude the experimental evaluation by contrasting the proposed model with a state of the art representative from the recurrent neural networks family of models. The neural model is based on a Long-Short-Term Memory (LSTM) network. We evaluate the incrementally trained HMMs against the LSTM model on the classification task. In line with the requirements on the model that were discussed in section 3.2, we evaluate the performance of both models with regard to the number of training instances needed to accurately predict the target class and the anticipation of incomplete sequences. As the LSTM network is trained in batch mode, the requirement to learn online is not met by that model. However, there is no principled limitation to the incremental learning of LSTM based models.

3.4.1 Comparing incrementally trained HMMs with a non-incremental baseline

Inducing Hidden Markov models by model merging is a strategy that fundamentally differs from the traditional training regime for Hidden Markov models that is based on iterative Baum-Welch reestimation of the model parameters given a fixed model structure that is priorly defined. As the proposed model learns incrementally from each individual training example without ever having a global view on the complete data, the learning problem is more challenging than the traditional batch scheme that has access to the complete data during training.

In this experiment we seek to validate the incremental learning scheme by evaluating the performance of the incremental process against a baseline that is trained by Baum-Welch parameter estimation. In the following we refer to the two training regimes as **incremental** for the model merging approach and **batch** for the traditionally trained model. We consider a setting where the system is trained on action instances corresponding to 11 of the 12 individual performers in the dataset (see section 3.1). We evaluate the system by presenting it with samples from a 12th performer.

Following this evaluation scheme, the dataset is partitioned into 12 folds with 1100 examples to train and 100 examples to test in each fold. The partitioning is implemented such that examples from an individual performer in a given fold are either in the training set or in the test set but not distributed among both.

Model	Compressed				Not compressed			
	F_1	P	R	σ	F_1	P	R	σ
Batch	0.86	0.82	0.90	0.11	0.82	0.75	0.91	0.11
Incremental	0.86	0.82	0.90	0.12	0.82	0.75	0.91	0.10

Table 3.1: Results of the 12-fold cross-validation for the batch and the incremental approach with and without repetitive sub-sequence compression.

For both of the models we trained a dedicated HMM for each of the four action classes. Test sequences were classified according to the action specific HMM having the highest probability to have produced the respective sequence.

Both training approaches show very similar performance characteristics, having almost identical results (see table 3.1 left) with both having an averaged F_1 score (harmonic mean of precision and recall) of 0.86. The most notable difference is a slightly higher standard deviation between the individual results of the 12 folds for the incrementally trained models. Also when considering the class confusion matrices in figure 3.7 there are merely slight differences in class specific classification performance for both approaches.

The reason for the higher standard deviation most likely lies in premature merges that would have been assessed differently with more data available. When early on during training the system is presented with weak examples, the incremental approach had a tendency to introduce weakly generalized substructure specific to those weak examples and possibly a small amount of successive examples that, later, with more data, is abandoned in favor of properly generalized parallel structure. This effect leads to a under-utilization of training data and thus to higher sensitivity to

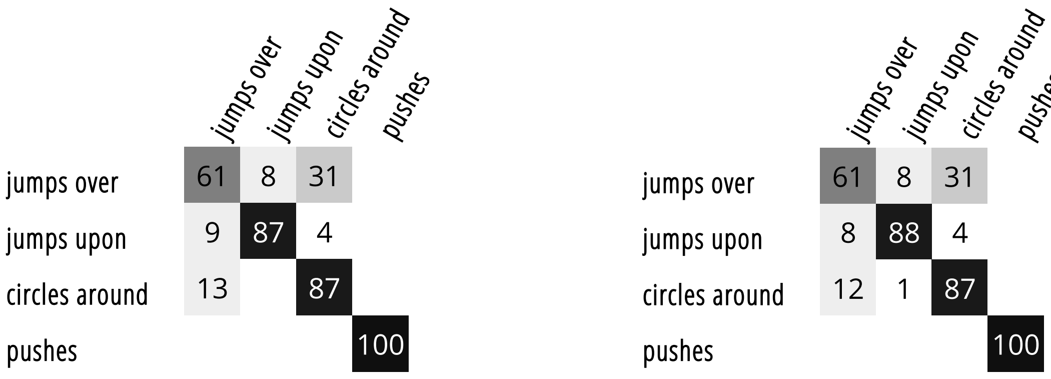


Figure 3.7: Averaged class confusion matrix for the batch (left) and the incremental (right) approach.

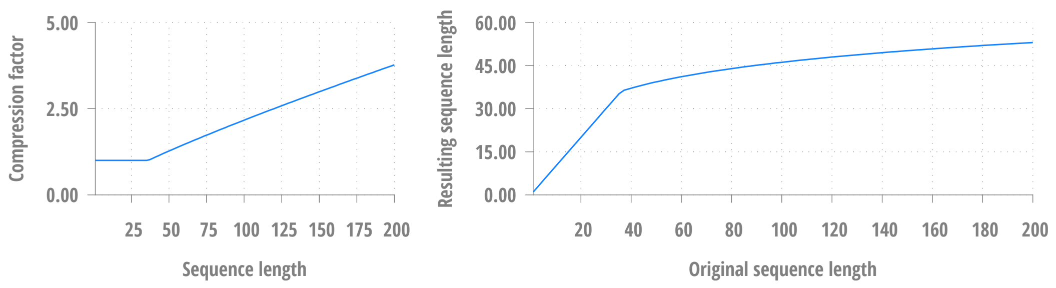


Figure 3.8: Effect of the logarithmic compression of repetitive sub-sequences. The logarithmic compression is effective starting at sequence length with more than $x \approx 36$ symbols

weak examples in the early phase of training.

In the following we investigate the effects of the compression of sub-sequences and the sensitivity of both training procedures to their most effective hyperparameters.

3.4.2 Compression of repetitive sub-sequences

This experiment investigates how the compression of repetitive sub-sequences (see section 3.3.1) affects the classification performance of both models. In total, the compression reduces the length of trajectories in the dataset to 150 QTC symbols on average which corresponds to a compression rate of 21.3%. The effect of the compression can be seen in Table 3.1, compression rates are visualized in Figure 3.8.

Both approaches perform considerably better with compressed sequences with an averaged F_1 score that is 4% higher in the compressed condition. Precision and recall are as in the previous condition unaffected by the training procedure, but the incremental approach had a slightly lower standard deviation between the results of the individual folds.

The compression scheme reduces the number of repeated successive symbols in the sequence and thus the length of the sequence in total. As the numbers of positions where successive symbols change remains the same, this compression puts more attentions to those changing positions in relation to the overall sequence length.

3.4.3 Parameter sensitivity

Most machine learning techniques introduce model parameters that are either optimized in the actual training procedure or that have to be specified beforehand. The first are often termed model parameters, the latter are often referred to as hyperparameters of the model that affect the training and thus the search for an optimal

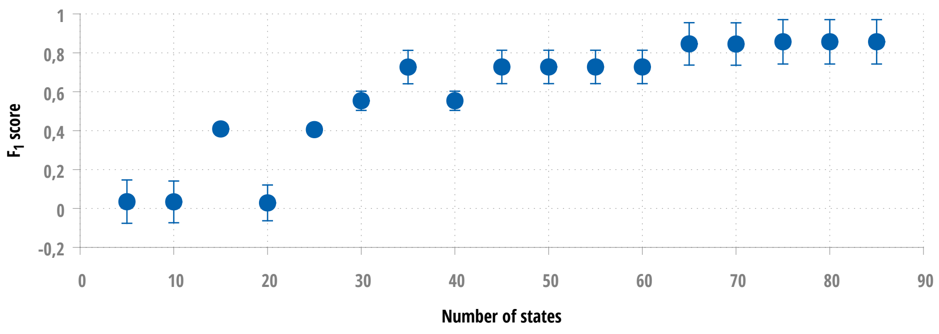


Figure 3.9: Development of F_1 scores as the number of hidden states is increased for the batch training approach. The errorbars indicate the standard deviation of the F_1 scores across the 12 folds.

configuration of model parameters. The model merging approach, as implemented in the incremental model, has a single tunable hyperparameter λ that weights the prior against the models likelihood. For the models trained in batch mode the most relevant structural parameter is the number of states, as the family and shape of transition and emission distributions is given by the learning problem.

In this experiment we investigate the impact of the λ parameter and the capacity of the traditionally trained model on the resulting classification performance. We also investigate how sensitive both models are with respect to those two parameters.

As can be seen in Figure 3.9 the classical Baum-Welch approach to HMM parameter estimation is highly sensitive to limited model capacity as a result to a limited number of states available to the model. If chosen too low, the optimization results in suboptimal models. The F_1 scores range from 0.03 to 0.86 as the number of hidden states in the model is increased.

The incremental approach on the other hand displays an almost linear response over the whole range of values for the λ parameter (Equation 3.7). While the F_1 score is stable with respect to λ, the number of hidden states in the model decreases

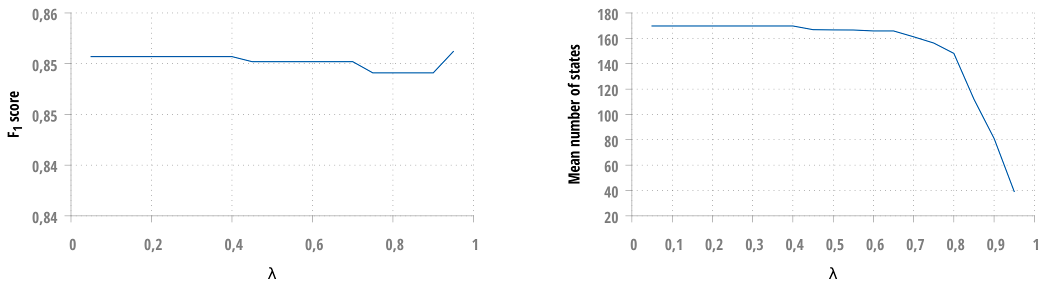


Figure 3.10: Sensitivity to the λ parameter of the incremental approach.

drastically (see Figure 3.10) from 170 to 39 (at $\lambda = 0.95$). Setting $\lambda = 1$ and thus evaluating the model only with respect to the prior favoring simpler models leads to poorly performing models with just a single hidden state. A slight peak score was achieved at $\lambda = 0.4$.

As a result we can conclude that the incremental training procedure performs comparably to the traditionally trained baseline, with the exception of a slightly higher standard deviation between training data folds. Also the compression of repetitive sub-sequences improves the classification performance considerably. In consequence the following experiments are conducted with incrementally trained models at $\lambda = 0.4$ and compression of repetitive sub-sequences enabled.

3.4.4 Comparing incrementally trained HMMs with a LSTM Baseline

In the previous section we evaluated the proposed component for action learning against a baseline that was trained using a traditional expectation maximization approach. In this section we evaluate the proposed model against a state of the art representative of the sequential neural networks family of models.

LSTM based action model

The LSTM based model is a discriminatively trained recurrent neural network that consists of a 128 node LSTM layer followed by a fully connected linear layer with softmax normalization and one output unit per target class. The softmax normalization allows the network output to be interpreted as class probabilities by constraining individual components of the output vector representing class scores to range between zero and one. The sum of all components is also constrained to sum to one.

In contrast to the HMM based approach which trains models specific to each of the individual classes of action, the LSTM based model learns a joint representation of the sequences of all 4 classes.

The model is trained by minimizing the cross-entropy

$$H(p, q) = -\sum_{x} p(x) \log(q(x)) \tag{3.8}$$

between the predicted and target class-distributions (p and q) using RMSprop (Tieleman et al. 2012) with learning rate $lr = 0.001$ and decay factor $\rho = 0.9$ over batches of 50 sequences. Minimization of the cross-entropy is equivalent to minimizing the negative log-likelihood of the data, which, in turn is equivalent to maximizing the likelihood of the data under the model, since the logarithm is monotonic.

Model	Compressed				Not compressed			
	F_1	P	R	σ	F_1	P	R	σ
HMM	0.86	0.82	0.90	0.12	0.82	0.75	0.91	0.10
LSTM	0.88	0.88	0.88	0.18	0.88	0.88	0.88	0.18

Table 3.2: Results of the 12-fold cross-validation for the HMM and the LSTM approach with and without sequence compression (Equation 3.4). Results are given as F_1 score, precision, recall and standard deviation σ.

We randomly select 10 percent of the training data available per class as a validation set to asses the performance of the model in each individual training epoch. After training a fixed number of epochs, we select the model having the highest accuracy on the validation set as the model for the subsequent experiments. Other than model selection we do not apply regularization techniques as the models had no tendency to over-fit and preceding experiments did not reveal any improvements in applying any further regularization techniques.

The network layout was optimized in a separate experiment to give the best trade-off between classification performance and training time. Training with 1100 sequences over 150 epochs takes about 4 hours on a single GTX-770 GPU while the incremental HMM model required only a few minutes to train. The selected best LSTM parameter set usually comes from the third quarter of training episodes.

The input to the network consists of sequences of QTC relations but in contrast to the HMM model the input to the network is not purely symbolic (the id of the relation as a single integer) but decomposed into a 4 element vector corresponding to the basic QTC constraints. The QTC relation (-1, 0, -1, 1) is thus represented as the vector [-1, 0, -1, 1].

HMM vs. LSTM

In this experiment we evaluate the performance of the HMM based model against the LSTM based baseline. The incremental HMM approach was specifically tailored to meet the set of requirements discussed in section 3.2, while an LSTM network is a state of the art representative for recurrent neural networks. The goal of this experiment is to evaluate whether the incremental approach can compete with the LSTM approach when trained in a novel-performer condition where the system was (as in the previous experiment subsection 3.4.1) trained on data from 11 participants and evaluated on data from the 12th. Additionally we evaluated both models in the early learning (paucity of data) and the early sequence anticipation condition.

	jumps over	jumps upon	circles around	pushes
jumps over	61	8	31	
jumps upon	8	88	4	
circles around	12	1	87	
pushes				100

	jumps over	jumps upon	circles around	pushes
jumps over	74	13	12	1
jumps upon	6	90	3	1
circles around	7	3	89	1
pushes	2			98

Figure 3.11: Averaged class confusion matrix for the HMM (left) and the LSTM network (right).

As can be seen in Table 3.2 both approaches perform comparably well. The LSTM network had a considerably higher standard deviation between the results of the 12 folds. We assume that the LSTM network gives higher precedence to differences in relative pace of the action performance, which is the factor which varies the most across participants in the data.

Note that only the HMM approach had a reject option, when none of the class specific HMMs assigned a probability higher than zero to the respective sequence. As the LSTM approach learns a joint representation across all 4 classes there is no implicit reject option, because the model will always assign softmax normalized confidences to each class. The case that more than one class is assigned the exact same confidence is highly improbably due to rounding issues in computer implemented floating point math.

Figure 3.11 shows the class confusion matrices for both approaches. The HMM approach frequently misclassifies *jumps over* sequences as sequences corresponding to *circles around* actions, for which the first half of the movement is identical to *jumps over*. This could be caused by the limitation of temporal context HMMs are able to take into consideration due to the markov assumption.

Early learning behavior

In this experiment we evaluate the performance of the two models when they are presented with little training data per class. Both networks are not trained from scratch when more training sequences are added to the training set, instead the incremental model is trained as usual with the novel training instance. The LSTM based model is re-trained on the set of training sequences in the current training set extended with the novel example. The parameters of the model from the previous training iteration are retained instead of being initialized with random values.

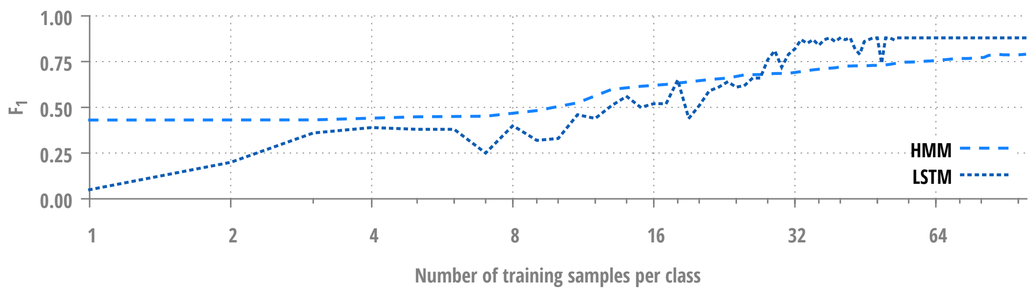


Figure 3.12: Early learning behavior, averaged over 10 random folds. HMMs outperform the LSTM network where very little training data is available, while the latter performs better with more than 28 training sequences. Note that the x-axis is scaled logarithmically.

Figure 3.12 shows that with an F_1 score of 0.44 the HMMs perform notably better than the LSTM network which starts at chance level after being trained on a single sequence per class. When trained with more than 28 sequences, the LSTM outperforms the HMMs clearly.

This condition is especially challenging for neural networks because they have many parameters to estimate from data and are known to require rather large amounts of data compared to models that were specifically tailored to operate under paucity of data conditions.

Early Sequence Anticipation

Applications in the field of human robot interaction often require the robot to produce action hypothesis when the respective action sequence is not yet completed. This experiment simulates this condition by letting the models classify incomplete sequences. We evaluate the classification performance with sequences truncated after 1 to 100 percent of their original length. The LSTM network has been trained for only 100 epochs to prevent it from striving too much towards relying on late parts of the sequences. Apart from that both approaches were not specifically trained for the classification of truncated sequences.

The HMM based model was quicker in capturing the concept of the *pushes* action, which was the only action where both objects moved. This is most likely the result of having a dedicated model per class in case of the HMM model. The LSTM based model performed better with *jumps upon* actions, supporting the suspicion that the LSTM assigns more weight on the pace of actions, which is also the case for the *circles around* action. Because *jumps over* is the first part of an *circles around* action, both models need about half of the trajectory to separate these actions as can be seen in Figure 3.13.

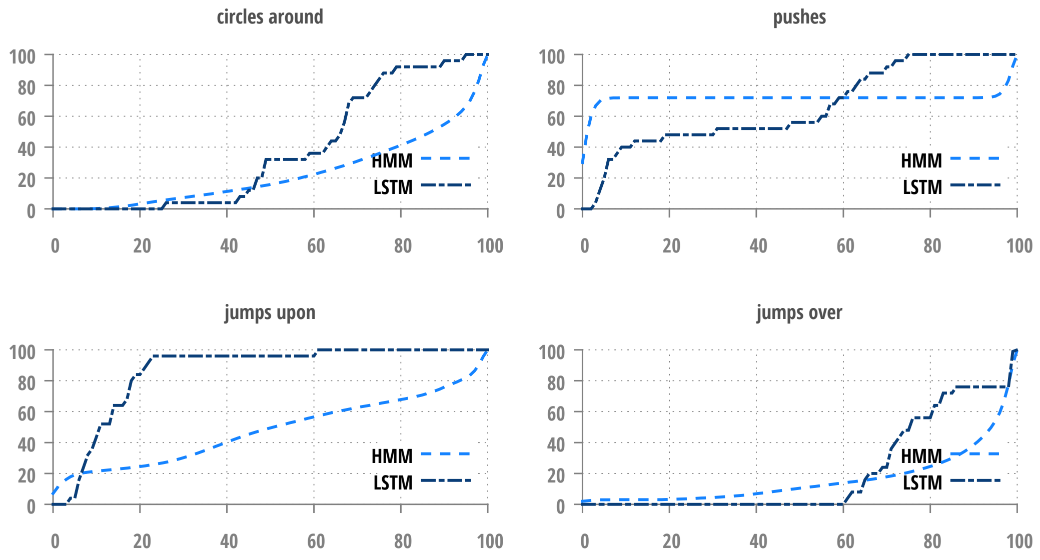


Figure 3.13: Early sequence anticipation behavior of the HMM and LSTM network for each of the four action categories. The x-axis represents the percentage of the sequence presented to the classifier and the y-axis the percentage of correctly classified sequences for which their classification result will not change if more of the sequence was presented (stable classification results).

3.5 Discussion

In this chapter we proposed a model for action learning that is designed to serve as a component in the multi-modal learning model that is developed in the next chapter. Induction of generalized categorical representations of action in the proposed model is implemented by incrementally merging specific HMMs into more general HMMs that have a higher entropy compared to the very specific initial HMMs. At the same time, the generalized HMMs should still assign substantial probability mass to the observed examples while minimizing model complexity. The incremental model merging approach is inspired by the observation that when faced with new situations, humans and animals alike drive their learning process by first storing individual examples (memory based learning) where few data points are available and gradually switch to a parametric learning scheme for better generalization as more and more data becomes available (Shepard 1987). Our approach mimics this behavior by starting with simple models that generate exactly one sequence which gradually evolve into more complex models as more data becomes available. With the incremental learning approach, the model continuously adapts its structural parameters parallel to the adaptation of the model parameters and thus strives to represent learned concepts at an optimal degree of complexity with respect to the model score given as training objective.

The models are optimized using a bayesian criterion that drives the induction and generalization process by two opposing forces, the prior and the likelihood. Initially, every new observation only adds to the model size, because the new samples are simply accommodated into the HMM. If the data is similar enough, however, the algorithm starts to collapse similar sequences into shared structure and thus decreases the model size. This way the model structure is determined in a purely data-driven fashion as a byproduct of the incremental generalization process.

The preceding experiments found that the proposed incremental action model is suitable to model actions of the types that occur in the dataset. The model performs well under conditions where only little training data is available to learn from or when sequences are to be classified early in their progression. The proposed model complies with all of the requirements (section 3.2) that result from a learning scheme that is compatible with *usage-based* and *emergentist* theories of early concept acquisition, which is a prerequisite for modality specific learning components in the multi-modal model (chapter 4).

The incremental model was evaluated against a traditionally trained baseline model and finally contrasted with a state of the art representative from the recurrent neural networks family of neural models. The proposed action model performed comparable to both baselines even though the learning problem in case of incremental learning is more challenging than it is for the two models that are trained in batch mode. Incrementally learned models receive training input one by one and update their internal state with each example. In contrast, models trained in batch mode receive their training input all at once and thus have the advantage of having a *global* view on the learning problem.

The proposed model is generative in that it supports to generate prototypical instances of the classes of action it represents based on a probabilistic framework. The generative nature of the model enables it to (mentally) simulate consequences of action, in that it can anticipate the typical progression of an action even when only the beginning of the sequence is given. When classifying incomplete sequences, both models have different strengths and weaknesses. The HMMs proved to be better in capturing strongly discriminative features, as in the *pushes* action where two objects were moving. The LSTM model was slightly better on average and proved to be more accurate in making predictions earlier in the sequence for all four action categories.

The model merging approach endows the system with the ability to learn on-line in the strict sense that each training example causes direct updates of the model structure and is not explicitly memorized for later consideration afterwards. The model has been shown to induce well performing representations of action even in conditions where very little training data is available. These early representations already reach about half of the peak performance starting with just a single training example in the classification task. The LSTM based model in contrast required about

23 examples to catch up with the classification performance of the class specific HMMs and took about 50 seconds of training time to converge while the proposed model took about three seconds. While the LSTM was implemented using a highly optimized code generation strategy based on the theano symbolic math compiler library, no particular effort was taken to optimize the runtime characteristics of the proposed model. It can be assumed that training times on our dataset could be reduced to fractions of a second when properly optimized.

Compression of repetitive sub-sequences

We proposed a logarithmic compression scheme for sub-sequences in which the same symbol is repeated many times. The compression reduces the overall length of sequences in the dataset by 21.3% on average (subsection 3.4.2), while preserving information about the acceleration along the trajectory, which increases the overall performance especially for very similar actions like *jumps over* and *jumps upon*, while still allowing to generalize over high variations in relative pace of the action performances.

Prior weighting

To offer more control over the learning process and specifically over the degree of generalization, we introduced a weighting parameter λ that weights the prior favoring simpler models against the model's likelihood. Especially in the early phase of the learning procedure when the models are based on little training data, this prevents the system from over-generalizing too early and inducing generalized sub-structure that would not have been induced in the light of more complete data. The experiment in subsection 3.4.3 evaluates how the λ parameter affects the model's complexity and performance. The experiment revealed that the parameter effectively controls the degree of complexity in the resulting models in terms of the number of states that are recruited to fit the learned representations in the model. The model has been shown to be relatively insensitive to different assignments of its value on the dataset. Due to the incremental nature of the model induction process and the approach that merges specific models into more general ones, the structural and model parameters are optimized in a joint process in which the model capacity (number of states) is continuously adapted to the complexity of the data. In contrast, the number of states in a traditionally trained model is fixed. If chosen too low, the resulting models tend to under-fit the training data. If chosen too high, the model often had a tendency to over-fit (Bishop 2006).

Applications

The model was specifically tailored to act as a companion model in an experiment that investigates the co-development of action and language. However it has properties that could be of use also in other fields of application, such as robotics and intelligent systems.

Learning from human tutors

Tutoring scenarios in robotics or intelligent systems are characterized by the circumstance that the system is mostly deployed in situations where it has to learn new concepts or capabilities in interaction with human tutors. To perform well in this mode of learning, the system has to quickly adapt to the tutoring in order to receive further training input. If the system was unable to learn from single tutoring examples it would be stuck with the same behavior, eliciting the ever same tutoring signals until the tutors eventually cease their efforts. We could show (see section 3.4.4) that the proposed model can immediately learn new actions with the first exposure to a corresponding example. Conceptually this is achieved by introducing specific structure for every new example that is then incrementally generalized through state merging. In contrast, e.g. neural models have to adapt their model parameters starting with an assignment that is mostly random (and thus probably wrong) initially. The adaptation process takes several examples, also because most gradient descent based learning algorithms try to adapt the parameters slowly to prevent the learning from simply skipping a good solution.

Realtime learning

Realtime learning refers to a mode of operation where the learning system is required to be fast enough to process new observations and experiences at least at the rate they arise in the environment the system is deployed in. The component of the action model that is computationally most demanding is the state merging procedure that occurs in the generalization phase. However, the two phases (memorization and generalization) of the induction process do not need to alternate strictly. In realtime scenarios, the model could first memorize data and let a background process generalize the model whenever there is time to do so.

Life-long learning in robotics

Life-long learning is not only a requirement humans are faced with. Also many intelligent systems are required to adapt to changing environments and novel concepts they are exposed to. For example, in the domain of elderly people care, it seems intractable to train a system for all possible situations that could occur. Instead, it would be preferable if the system could continuously self-adapt to changes in the environment, e.g. a different layout of furniture that is required to compensate for mobility challenges of residents. Life-long learning is also relevant in many other domains of applications ranging from cognitive robotics (Cangelosi et al. 2015), self driving cars or surveillance systems. Due to the incremental nature of the proposed model, the model is applicable in life-long learning scenarios even in conditions such as *concept drift* that are often associated with life-long learning.

Learning under concept drift

Concept drift is a phenomenon that is characterized by an unpredictable shift in the characteristics of some target variables the system predicts. Predictive modelling is faced with the challenge to learn a model of a concept from historical data that is then applied to the prediction of new data for which the correct answer is not known in advance. If the data distribution is non-stationary, novel examples can behave differently from the historical data the system was trained with.

LSTMs for example are sensitive to *concept-drift* because with each training-run they update their entire state in contrast to the incrementally trained HMMs, which only update paths directly related to the input example. Updating the entire model harbours the risk of overwriting existing knowledge, continuously shifting the model towards recent training data. Many machine learning algorithms are based on the assumption that the training data is i.i.d. (independent and identically distributed) meaning that successive examples are statistically independent but originate from a similar statistical process that produces identically distributed examples.

Concept drift can be approached from many different angles. One family of approaches often applied in cognitive robotics is the *two-layered* memory architecture that consists of *short-term* and *long-term* memory layers. The short term (STM) layer quickly adapts to changing distributions in input data and can sacrifice existing knowledge in the process. The long term memory (LTM) is parameterized to be more resilient against short term drift in input data and thus represents general knowledge acquired during longer time. STM and LTM are linked such that knowledge from STM slowly manifests in general representations in LTM over time.

The incremental model approaches concept drift by retaining paths that were previously established. Even if the input data distribution changes the knowledge about the previous characteristics of the statistical process is still represented in those paths. The model merely shifts probability mass towards the newer paths according to the evidence (counts) the model acquired. The incremental model thus unifies short-term and long-term representations in shared structure.

Limitations

The proposed action model proved to be suitable to serve as a component in the multi-modal model to learn grounded representations of action and language that is developed in the next chapter. However, the model has some limitations that potentially need to be addressed in order to make the model generally applicable.

Premature merges

The model merging approach as it is implemented in the present model corresponds to a monotonic search from models very specific to a single example to more general models. There is currently no mechanism that can undo merges that appeared advantageous at first but would have been assessed differently in the light of new evidence. If parts of the model were merged prematurely, the only compensation mechanism the current model provides is to introduce new specific paths that are again incrementally generalized, leading to parallel substructure in the model that represents conceptually similar action trajectories. Having parallel structure to represent similar data could potentially lead to under-utilization of training data and thus suboptimal performance of the model, as the probability mass is distributed among separate substructures.

As a rough measure, on our data we found that approximately 7% of the structure in the models were abandoned, meaning that the respective paths were only used to explain a single example during prediction. In our experiments, premature merges did not have a major effect on the performance of the models as the dataset contains only a few hours of recorded action trajectories. However, in different applications the effect could be considerable. For example in life-long learning scenarios where the system adapts continuously to new and changing actions, abandoned structure can accumulate over time. As the simplicity prior (favoring simpler models that have less states in general) does not consider if those states are actually used in explaining current data, this effect would drive the model into inducing more and more compact representations.

An *unmerge* operation could potentially be implemented driven by, e.g. confidences that are tracked during classification. When, for a given sequence, the most probable sequence of states in the model has been estimated, the model could, for each state along the path, record the contribution to the overall confidence. When a state is identified that reduces the confidence significantly more than others, this could potentially signify a premature merge.

In the following experiment we investigate how language and action can co-emerge in an emergentist framework. The incremental approach for action learning serves as the component to induce categorical representations of actions.

4 Implemented theory of the coupled co-emergence of linguistic constructions and action concepts

In this chapter we implement the computational model to investigate how linguistic and conceptual structure can co-emerge by shaping and influencing each other. The model exemplifies how grounded representations of action and language can develop inspired by *emergentist* and *usage-based* theories of grounded concept acquisition.

It has been well established that conceptual and linguistic development go hand in hand (Bowerman et al. 2001). For one, language structures thought and shapes the concepts we acquire. This is the main claim behind the theory of *linguistic relativism*, more widely known as the Sapir-Whorf hypothesis (Philip Wolff et al. 2011). For another, linguistic development does not happen in isolation but rather conceptual development is also a prerequisite for language learning as linguistic constructions need to be "mapped" to concepts that represent the meaning of the constructions. This so called mapping paradigm underlies most of the work on associational language learning involving cross-situational analysis (see McMurray et al. 2012; Siskind 1996; L. B. Smith 2000; L. Smith et al. 2008). However, the detailed mechanisms that are involved in the co-emergence of linguistic and conceptual structures have not received prominent attention so far. Computational models can contribute to enhance our understanding of such processes by providing an implementable and thus explicit theory that accounts for the phenomenon of co-emergence of representations across different modalities.

The proposed model accounts for the co-emergence of linguistic and conceptual structure for the case of action verbs and the grounded action concepts they denote. In chapter 3 we defined actions as repeatable processes that are so important, that we assign a name to them. We focus in particular on the case of action verbs and actions that can compactly be described by corresponding utterances and develop a system composed of two aligned components to learn the grounded meaning of a verbal construction without requiring the prior existence of corresponding action concepts. The model can distill the essence of the meaning of a verbal construction as a process of incremental generalization starting from a meaning that is very specific to a certain context in which the verb has been encountered. The meaning of verbs in this model are understood as evoking a simulation of the grounded process rather than being a static concept.

The approach is inspired by *usage-based* and *emergentist* (Behrens 2017) theories of language acquisition that are based on the assumption that language learning proceeds from specific to general where the specific constructions are incrementally generalized and entrenched by means of social joint-attention and continuous generalization. This approach thus mitigates the necessity for children to have innate categorical structure to map novel linguistic constructions upon. Increasing generalization also assists in resolving possible underspecification in linguistic input as the formation of general constructions can be delayed until enough experience is gathered.

The theoretical framework of *emergentism* assumes the existence of a powerful generalization process that drives the incremental emergence of categorical knowledge from observed examples. With increasing linguistics experience, more abstract patterns evolve from the learning process, but these abstract patterns are assumed to be still grounded in usage, e.g. in a prototypical generative model of the corresponding action. A central phenomenon in *usage-based* linguistics is *entrenchment*, the observation that repeated encounter of specific stimuli stabilize corresponding cognitive representations. However, repetition alone can not account for increasing abstraction of general categorical information. In order to form generalized categories, a learner has to recognize and exploit similarities as well as dissimilarities, by filtering out aspects that do not recur, while registering commonalities between new stimuli and already learned specific or general cognitive representations. Behrens (2017) describes *emergentism* as the term that emphasizes the idea that qualitatively new and more complex structures can emerge from simpler, basic facts, thus positioning emergence as a central component in human language learning. The emergence of complex structure is the result of the powerful generalization processes that is required by *usage-based* theories as a replacement to the necessity for innate categorical knowledge.

The categorical representations the proposed model induces can be used to recognize and generate sentences describing actions and their grounded meaning in terms of qualitative action trajectories. The model learns incrementally from coupled examples of action sequences together with a sentence that describes the corresponding action. Starting with very specific examples observed from the environment, the model gradually develops new and more complex structure by incremental generalization. Figure 4.1 shows an example where the model learns from two coupled examples of language and action.

We evaluate the model with respect to its generalization abilities both at the linguistic and conceptual level. The experiments are carried out on a dataset consisting of performed actions corresponding to four types of actions (jump on, jump over, circle around and push) performed by subjects when prompted with sentences verbalizing the action in question. The evaluation is conducted on three tasks i) a matching task consisting of deciding whether a given utterance describes a given action instance, ii) a selection task consisting in selecting one out of three action instances that is

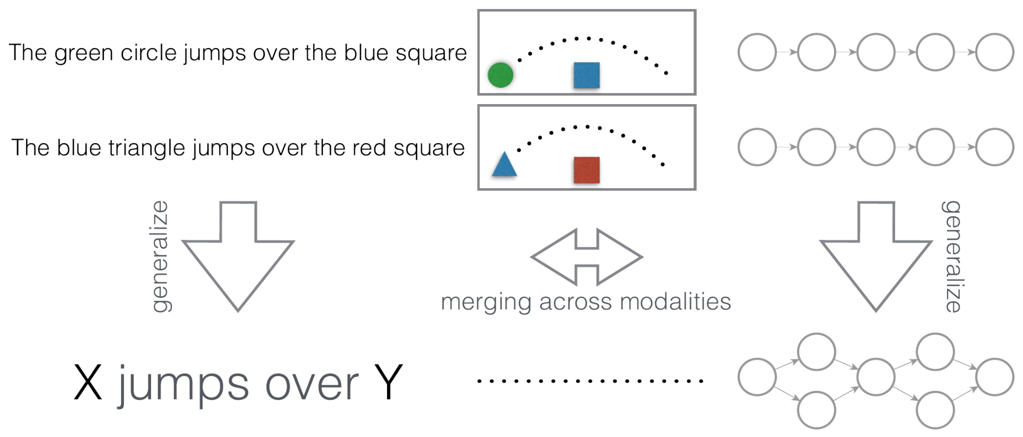

The green circle jumps over the blue square
The blue triangle jumps over the red square
generalize
merging across modalities
generalize
X jumps over Y

Figure 4.1: The model for multi-modal learning learns from coupled examples of language and action given as a trajectory of objects involved in the action together with a sentence that describes the action verbally. The model consists of three components, a component for action learning, a component for language learning and a component that coordinates the learning process and drives the cross-modal induction of general categorical representations through merging similar structure. The figure shows how concrete examples are generalized into abstract linguistic constructions such as “X jumps over Y” together with a generalized graphical model that probabilistically represents the two constituting action trajectories.

described by a given utterance and iii) a generation task consisting in generating a sentence describing a given action instance.

Model

The model consists of three components. The **component for language learning** and the **component for action learning** induce categorical representations of knowledge corresponding to their respective modality. The third component, the **generalization controller**, coordinates the incremental learning process such that linguistic and action concepts can co-emerge by shaping and influencing each other. Figure 4.2 gives an overview over the components the model is composed of.

The **component for language learning** is based on an existing model for the “item based induction of linguistic constructions” (Gaspers and Cimiano 2014) (see section 2.6) that is extended such that it can accommodate the multi-modal learning process. The **component for action learning** transfers ideas from *emergentist* and *usage-based* theories of language acquisition to the domain of action in that it also learns in an incremental and item-based fashion. The component is identical to the action model developed in chapter 3.

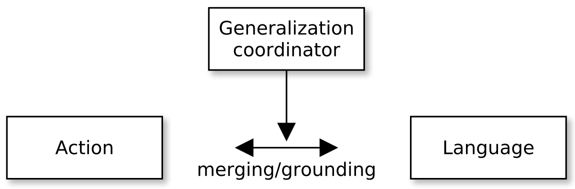


Figure 4.2: The model consists of three components. The component for action and language learning induce abstract categorical representations of concepts in their respective domain. The generalization controller drives the learning process, such that grounded abstract conceptual representations can co-develop in cross-modal learning.

Learning in the proposed model is organized such that language and action concept acquisition go hand in hand in the sense that generalization steps are not applied only separately, but generalization at the linguistic level should trigger the learner to also look for potential generalizations of actions observed in the context of the same generalized utterance. Equivalent linguistic constructions are thus expected to denote equivalent or unifiable grounded action concepts. Generalization at the conceptual level forces a learner to induce near-synonym relations, that is to postulate relations between linguistic constructions that look different at the surface level, but clearly have commonalities in their meanings. For example, actions denoted by the words *jump* and *hop* might look similar in the corresponding action but are very different in their linguistic surface form. This procedure allows to acquire equivalence classes of linguistic constructions for which the evoked action concepts can be unified in one model, for instance because the underlying action concepts are sufficiently similar.

Learning in the model is completely unsupervised utilizing only unification cues in the sense that if two concepts (or single examples) can be unified into a more general abstract representation in one modality, unifying the corresponding grounded representations in the other modality would also yield meaningful abstraction. Unification is only carried out on both ends of a grounding relation, e.g. when two concepts in action can be unified, the corresponding two grounded concepts in language are also unified. This way the original coupling of action and language that is present in the training examples persists during the training process.

An important aspect when modeling human language acquisition is the ability to learn online, where each example causes direct updates of the model's internal representations instead of storing the training examples for later batch processing. Learning online allows to dynamically adapt to changes in the environment and it enables learning under the premise that the learner is constrained with respect to memory. Pearl et al. (2011) assume that learning in humans is inherently memory constrained and in case of language only one utterance coupled to a representation of it's meaning can be processed at a time.

This mode of learning is reflected in the model by the incremental learning process that proceeds in two phases. In the first phase, the model memorizes observed examples consisting of action trajectories together with sentences describing the corresponding action that, in the second phase, are subsequently unified and generalized by successively abstracting from concrete memorized examples through structure merging and recombination. This two-phased approach further endows the model with the capacity to learn new concepts based on a single exposure to a given informative example. This is often referred to as fast-mapping or one-shot learning.

Both modality specific components for action and language comply with theories of *usage-based* and *emergentist* acquisition of categorical structure in that they are item-based and incremental in nature by updating their internal representations directly with each new observation. The model does not assume innate linguistic and categorical knowledge to pre-exists. However, it relies on the assumption that the learner has already formed basic perceptual capabilities to detect objects and their spatial relations in the scene as well as being able to represent speech signals as sequences of discrete symbols, e.g. characters. The generalization process required by *emergentism* is present in the model in the form of the incremental generalization procedure that incrementally unifies similar structure within one modality while preserving valid grounding relations across modalities that have been observed during training.

4.1 Learning scenario and input data

We consider a learning scenario in which the system learns from utterances describing different actions coupled with trajectories of object positions corresponding to these actions (see Figure 4.3). The scenario corresponds to the dataset as introduced in section 3.1 with the exception that for this experiment we modified the collected data in that we used two different verbs for each of the four actions. More specifically, for each action instance in the dataset we randomly choose one out of two possible utterances as the descriptions. The following patterns were used to generate the descriptions:

- *trajector* [pushes|shoves] *landmark* from left to right
- *trajector* [jumps|hops] onto *landmark*
- *trajector* [jumps|hops] over *landmark*
- *trajector* [revolves|circles] once around *landmark*

Notice that the patterns were chosen such that the synonym relation could not be revealed using language alone. For instance, taking solely linguistic variation into account could also yield an incorrect merging of patterns "*trajector* jumps onto

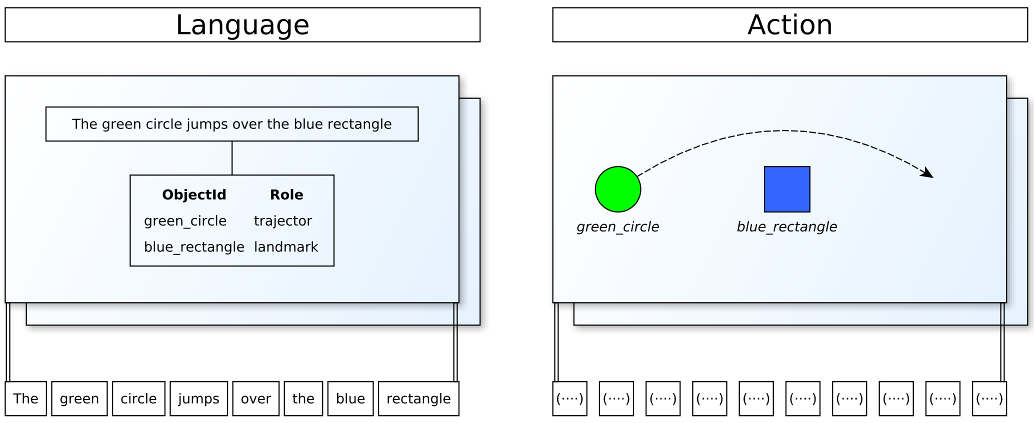


Figure 4.3: The input to the system consists of coupled examples of sentences describing an action given as a sequence of words and the corresponding action trajectory as it was performed by a participant after being given the sentence as an instruction. The input sentences are coupled with identifiers of involved objects and their role in the action. In this case trajector is the object that is actively moved during the action and landmark refers to the object that is not actively moved.

landmark" and "*trajector* jumps over *landmark*" into "*trajector* jumps [onto|over] *landmark*". Hence, the system has to take similarity at the meaning level into account in order to establish correct synonym relation.

4.2 Model Definition

In this section, we describe in detail our model for accounting for the co-emergence of linguistic constructions and the action concepts they denote. In essence, the model consists of two learning components corresponding to the two modalities action and language and a central component that coordinates the multi-modal learning using signals from both learning components (see Figure 4.4). Both modality-specific components incrementally develop representations that are grounded in each other and participate in a joint learning process where new knowledge extracted in one component causes direct updates also in both components. The superordinate generalization and grounding coordinator coordinates the cross modal generalization and grounding process using similarity signals from both modalities.

The component for action learning corresponds to the incremental action model developed in section 3.3. The component is responsible for inducing generalized action concepts from specific examples of performed actions following an incremental model merging approach based on generative probabilistic models, in our case Hidden

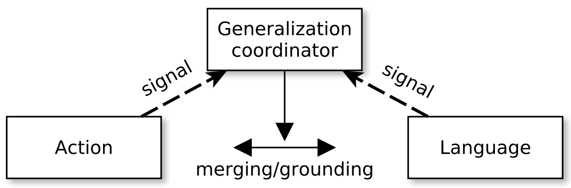


Figure 4.4: The multi modal generalization process is hierarchically organized, using signals from both modalities to orchestrate the unification of representations within the modalities by merging similar structure as well as the grounding of structure from one modality in structure from the other modality that is similar in meaning.

Markov Models, where the observation alphabet corresponds to a set of qualitative QTC (see section 2.5) relations that a learner is assumed to be able to recognize in a visual context.

For action, the learning problem can be defined as:

- Learn typical patterns of relative movements of *trajector* and *landmark* that are indicative for any given category of action.
- Induce generalized paths capturing movements that are typical for the given category of action.
- Generalize action concepts abstracting from specific action instances.

The component for language learning is based on a model that was published before and models the acquisition of syntactic constructions using symbolic meaning representations (see section 2.6). For this experiment we extended that model to use concepts from the action model to represent the conceptual meaning of sentences.

For language, the learning problem can be defined as follows:

1. Learn the names for objects through associational learning by tracking co-occurances of objects and mentions.
2. Induce generalized linguistic constructions by associational learning and the postulation of slots in general constructions.

The third component that coordinates the multi modal learning and grounding process uses signals from both components to control the degree of generalization within the learned representations. The component for multi-modal learning uses signals exposed by both learning components to accomplish the following generalization tasks.

- Establish grounding relations between modalities starting with early examples that are maintained during the generalization process.

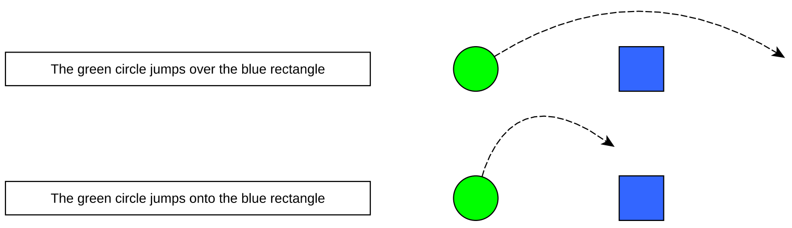

The green circle jumps over the blue rectangle
The green circle jumps onto the blue rectangle

Figure 4.5: Two sentences that appear to be (falsely) mergeable on the surface level into the more general pattern "the green circle jumps [over—on] the blue rectangle" but that are grounded in very different action trajectories. The generalization controller in this case would follow the grounding relations to the action domain and determine that the two associated action instances denote incompatible actions that can not be unified. The grounding coordinator would not carry out the proposed merge in this case and thus prevent the system from establishing an incorrect synonym relation between *over* and *on*.

- Coordinate the gradual generalization process by evaluating similarity signals from both learning components and eventually instruct the components to merge structures that have similar meaning across modalities. This process enables to resolve synonyms in language and to bootstrap the unsupervised learning in action.

The learning process is organized in two phases that are successively run for each individual training example. In the **memorization** phase, both components create a representation that is very specific to that particular example. The grounding coordinator establishes a grounding relation between the newly created specific representations. In the subsequent **generalization** phase, the components for language and action propose to collapse similar structure within their modality into more general representations.

The overarching coordinator component evaluates the proposed generalization operations and performs only those operations that also yield meaningful results when also generalizing the representations in the other modality that have a grounding relation to the proposed generalization candidates in the first modality.

For example "the green circle jumps over the blue rectangle" and "the green circle jumps on the blue rectangle" could, on the language level be proposed to be falsely merged into the more general pattern "the green circle jumps [over—on] the blue rectangle". In this case the coordinator would recognize that the action those sentences are grounded with, denote incompatible actions, specifically the "jumps over" and "jumps on" actions (see Figure 4.5).

The implementation of the incremental multi-modal learning procedure relies on signals from both learning components to drive the generalization process that are described in the following section. The definition of generalization signals is followed by the description of how the language model was extended to facilitate establishing grounding relations between action and language. In section 4.3 we describe how the incremental learning and generalization process is organized.

Generalization signals

Both learning components expose signals that are used to communicate the fact that similar structure developed within a modality specific model that has the potential to be further generalized. Similarity of structure is measured in both modalities by adopting the idea of relative entropy. The term *signal* refers to the learning process being fundamentally asynchronous, meaning that all three components proceed independent from each other and coordinate the multi-modal learning by exchanging signals. As can be seen in Figure 4.4 the generalization coordinator listens for the generalization signals to coordinate the learning process in the generalization phase (see section 4.3).

Action The action model defines a similarity relation between induced representations that is based on the notion of how probable a second model would accept sequences that the model generates. The similarity was derived from a distance metric between HMMs that is based on the Kullback-Leibler distance between distributions. This procedure is similar to the one proposed by Juang et al. (1985) and based on the intuition that the similarity between discrete Hidden Markov models can be estimated by letting the first model generate a number of sequences O_T (probabilistic walks from the initial to the final state) and then measure the difference between the log probabilities of both models to accept those sequences.

$$D(\lambda_1, \lambda_2) = \log P(O_t|\lambda_1) - \log P(O_t|\lambda_2) \tag{4.1}$$

To evaluate this measure we generate 10 sequences with a maximal length of 1000 symbols or less if the final state was reached and take the average of the resulting distances.

Language The generalization signal for the language component is based on a *mergeability* condition that states that two paths are mergeable if they differ in at most k positions. As introduced in section 2.6, the model distinguishes between two sets of elements that group similar entities in a path. Sets of slot filling elements represent elements that play a role in the grounded meaning, e.g. objects that either occur in the trajector or landmark role. Sets of linguistically optional elements are groups of elements that are not associated with any meaning in the observed context, more specifically these elements can differ across sentences that are observed together with similar meaning in terms of action trajectories. Generalization across those two types of sets is performed independently in the *slot driven* and *syntactic* generalization step of the language model.

Grounding syntactic constructions in qualitative action models

In order to facilitate cross modal learning, the component for inducing generalized linguistic constructions is extended to incorporate the action models captured by the HMMs.

Original model The model for language learning as originally proposed assumes the situational context in which the sentence was uttered to be given in terms of predicate logic formulas. The model assumes further that the meaning of the sentence in the conceptual domain is among the facts given as part of this context. For instance in the following example

NL utterance	The green circle jumps over the blue rectangle.
mr_1	*participate(SUBJECT:p1, STUDY:mml1)*
mr_2	*jumps_over(TRAJECTOR:green_circle, LANDMARK:blue_rectangle)*

the sentence “The green circle jumps over the blue rectangle” would be accompanied with the meaning “*jumps_over(TRAJECTOR:green_circle, LANDMARK:blue_rectangle)*” in addition to possibly other aspects of the context that are unrelated to the meaning of the sentence, such as the fact that the action was performed by participant *p1* while participating in the study *mml1*.

Extended model The model is extended to represent the grounded meaning of verbal constructions in terms of the Hidden Markov models that are induced by the component for action learning via grounding relations that are established from single examples and maintained during the process of incremental generalization.

In essence, the symbolic predicates ‘observed’ in context are replaced by (time-indexed) observations describing the movement of detected objects and the role they appear in.

NL utterance	blue_circle jumps over green_rectangle.
Objects	*trajector*: *blue_circle*
	landmark: green_rectangle
Moves/positions	*move(1234,blue_circle,[11:12]);*
	move(1277,blue_circle,[12:13]);...

When observing such an example, both a specific construction ‘The green circle jumps over the blue rectangle‘ and a specific HMM capturing that particular movement sequence are generated. We assume that a learning system is already able to recognize particular objects, e.g. *green_circle* and *blue_rectangle* as well as the role they play (e.g. trajector or landmark). Note that we do not assume that the system already knows the names for such objects, for instance there is no direct correspondence between the identifier of an object *green_circle* and it's surface form “green circle” in the sentence. Establishing these correspondences is part of the learning problem.

On the representational level, the language model captures learned utterances as paths in a graphical network. Each path constitutes a (generalized) pattern as a sequence of lexical items or words. Replacing the predicate based meaning representation with a meaning representation based on the previously developed action models associates each path in the slot and frame network with an action concept represented as a Hidden Markov model in the action domain.

As we focus on actions in which some trajector moves or is moved relative to some landmark or ground, we assume that the system is able to observe positions and identities of trajector and landmark. Our Hidden Markov Models in essence thus model the action specific probability of a given sequence of qualitative relations describing the relation between a trajector and a landmark over time as the fundamental meaning of encountered verbal expressions.

4.3 Multi modal learning of language and action

Cross-modal learning is organized differently compared to how learning was organized in the individual components. In the multi-modal model, learning is driven by an independent process that uses signals from both modalities to drive the generalization and grounding process. The actual learning is still performed by the individual components. The acquired knowledge is also local to the respective component. However, learned concepts in one modality always maintain a grounding relation to another concept in the other modality.

The induction and generalization process proceeds, as in the model for action learning, in two phases

Memorization phase Upon the first exposure of a certain example, both modalities create a representation that is very specific to the given example. In action this is a Markov Chain accepting/generating the respective sequence with a probability of 1. In language, each example first causes an update of the lexical layer, where connections between lexical units (words as surface forms) and semantic referents (unique identifiers) are established and reinforced if the network already contains the necessary information to decompose the sentence accordingly. When the lexical layer is updated, the sentence is memorized verbatim to update the construction layer in the generalization phase.

Generalization phase Generalization as abstraction from concrete examples is organized in two layers. At the top layer, the generalization coordinator gathers signals from the two learning components that indicate the existence of similar structure in the respective modality which can be merged into more general representations within that modality. As in the multi-modal model, learning is not modality specific but instead conceptual structure should co-emerge by shaping and influencing each other. Not every generalization step that would produce meaningful results in one modality is directly applied, instead, the generalization coordinator follows the grounding relations between modality-specific representations and evaluates if merging the grounded representations would also produce meaningful results. If that is the case, the coordinator instructs both components to execute the merge on the affected structure.

In the following we describe how the memorization and generalization phases proceed in detail:

Memorization phase

In the memorization phase, the multi-modal learning component would first ask both components to incorporate the given sample into their internal models and then establish a link between the newly introduced representations that grounds one modality in the other. Training instances consist of an action trajectory given as a sequence of QTC relations and the sentence that describes the action verbally together with detected object identities and their role in the action (trajector or landmark).

Action The memorization phase for the action model is similar to the procedure described in section 3.3.3 but differs in that new examples are not instantaneously included into existing models, as in this experiment there is no categorical information attached to the examples that could be used as supervision. Roughly summarized, memorization is implemented by first introducing a Markov Chain having a dedicated state for each QTC relation in the sequence. Each of those states emit their respective symbol of the sequence with a probability of 1 and is chained to transition to the next state, representing the successive symbol in the sequence. The resulting sequence constitutes a maximum likelihood model for the given example.

Language Similar to the memorization mechanism in the action model, the language model first creates a path specific to the given input sentence, e.g. for the sentence "The green circle jumps over the blue rectangle" the following construction would be generated and introduced into the network. In a subsequent step, the component

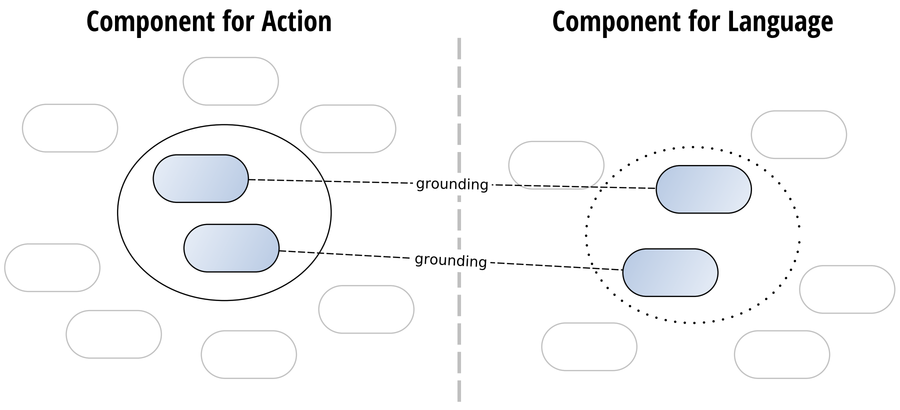


Figure 4.6: During the generalization phase, the generalization coordinator collects candidates that comprise similar structure within the learning models that can be merged to yield more general representations. For example, in the figure above, the two circled concepts on the left side are merge candidates in one modality and connected to respective concepts in the other modality via a grounding relation. The generalization coordinator would only merge structure if the grounded structure is also similar enough to be mergeable.

would track co-occurences of the lexical units appearing in the sentence and semantic referents (objects and the role they appear in). Co-occurences between lexical units and their referents are tracked in the word construction network as described in section 2.6. Co-occurrences are not tracked as raw frequencies but rather rescaled using an extended hebbian learning mechanism (Gaspers and Cimiano 2014) that strengthens associations between units and referents that were observed together but at the same time weakens other associations.

Generalization phase

In the generalization phase the model searches for commonalities between similar structure within each modality specific model and on the grounding level.

At first, both modalities search for similar structure within their modality and compile a list of sub-structures that are similar and constitute good candidates to be merged into a more general representation. More specifically, merge candidates are tuples of modality-specific representations (HMMs for action and patterns in the *slot and frame* network for language) that, if collapsed into a more abstract representation, would render the model to be a more general representation for the concepts it denotes. The fact that mergeable structure developed within a modality-specific model is communicated to the generalization controller asynchronously as signals.

The generalization coordinator takes the list of merge candidates from both modalities and for each candidate merge $c_m \in C_m$ from modality m follows the grounding relation to concepts in the other modality $\neg m$ to estimate whether those representations $c_{\neg m} \in C_{\neg m}$ could be merged too. This estimation is based on the same *mergeability* criterion that both learning models use to propose merge candidates (see section 4.2). If structure is mergeable at both ends of the grounding relation, the generalization controller would instruct both learning models to perform the merge accordingly.

The bidirectional interaction between the two learning components that results from the central coordination process can be summarized as follows:

- When the component for language learning encounters two specific constructions that can be merged into a generalized slot-and-frame pattern by abstracting from parts of the sentence through the introduction of slots, it performs this generalization only if the HMMs associated to the specific constructions are mergeable as well. Figure 4.6 depicts how representations of action and language are grounded in each other.

  The similarity measure is derived from a distance metric between HMMs that is similar to the Kullback-Leibler distance between distributions. We let both HMMs generate sequences and for each of these sequences accumulate the difference between the likelihood of that sequence given the generating model and the likelihood given the other model. This procedure is similar to the one proposed by Juang et al. (1985). If the models are mergeable, then both HMMs are merged into a new HMM that represents a more general action concept in the sense of accounting for more variability in action performance.

- In case the component for inducing generalized actions detects that two HMMs associated with different syntactic constructions are similar in the sense that they very likely represent the same action, then it is inferred that both constructions are synonyms of each other.

- In case two sentences are exactly the same, the two HMMs are directly merged.

Action The component for action learning represents individual memorized examples as maximally specific Hidden Markov models. As learning proceeds the component accumulates many individual HMMs at various degrees of abstraction that are incrementally generalized at the level of individual models and within each model. The first corresponds to merging sufficiently similar models. The second corresponds to the state merging procedure within models (see section 3.3.3). Generalization is always triggered by the generalization controller, however each component searches for similar structure that can be merged and sends a signal to the coordinator if such structure is found. In the component for action learning the search for generalizeable structure is triggered on two occasions:

- After a new trajectory is memorized
- After structure within the action model is collapsed and generalized through state merging

Mergeability between action models is defined as the similarity $0 \leq sim(M1, M2) \leq 1$ between the action models being above a given threshold of 0.86 which was optimized independently using randomized grid search. The similarity measure corresponds to the symmetrized Kullback-Leibler divergence as introduced in section 4.2.

Language Generalization in language is similar to the originally proposed procedure as described in section 2.6 with the exception that the language model does not directly merge mergeable paths but instead emits a signal to the generalization coordinator that mergeable paths exists. The coordinator then decides if the merge is to be performed based on whether it is also meaningful to merge the action models the linguistic constructions are grounded in.

A mergeability criterion mediates between specificity of constructions and possible over-generalization. Two paths are assumed to be potentially mergeable if they differ in at most k positions.

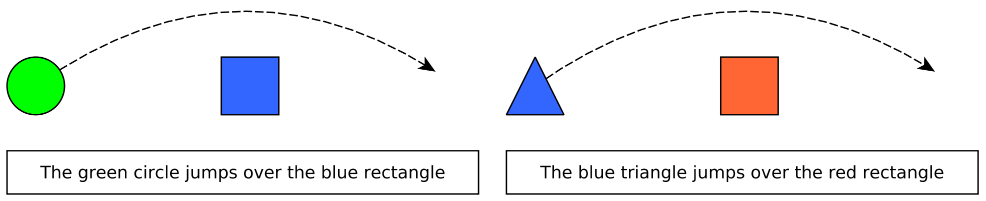
There are two types of sets of elements (SE): *slot filling elements* represent sets of lexical units (short sequences of words) that have a role in the associated meaning (e.g. act as trajector or landmark). Sets of linguistically optional elements are characterized by the fact that if they are replaced, they do not cause a change in the associated meaning. Both categories develop in independent generalization steps.

Sets of slot filling elements are identified by searching for differences in the patterns which lead to corresponding differences in corresponding meanings. For example two sentences “The green circle jumps ...” and “The red triangle jumps ...” would be associated with an action where the objects identified by *green_circle* and *red_triangle* would appear in the trajector role.

Cross-modal interaction in learning

In essence, both models take as input sequences of words and qualitative relations describing the relations between a trajector and some reference object, respectively. For example, in the following figure, the sentence “the green circle jumps over the blue rectangle” would be accompanied with a sequence of QTC relations that correspond to the semantic relations the two objects were in. Those relations were calculated using the position and velocity vectors sampled at regular intervals (see section 2.5).

Upon first occurrence of a certain utterance together with an action sequence, both models create a category most specific for that given sequences of words and action. In case of the action model the sequence is represented as maximally specific Markov chain and the sentence is represented as a path in the construction network. Later, these most specific categories are generalized as more and more similar examples are observed, leading to entrenchment and generalization. A schematic overview over the model was depicted in Figure 4.1. It depicts the following example of two input sentences from the dataset: 'The green circle jumps over the blue square.' and 'The blue triangle jumps over the red square' as well as two corresponding action sequences.


The green circle jumps over the blue rectangle
The blue triangle jumps over the red rectangle

Learning, as generalization from data, can be initiated by either one of the two modality-specific components:

Generalization driven by action The component for learning generalized action concepts from sequences of qualitative relations would, in the mentioned example, generate the hypothesis that both action sequences can be merged into a more generalized action category based on the assessment that both sequences are sufficiently compatible or similar. The system could infer that their corresponding utterances or linguistic instructions might also be regarded as equivalent. In this way, our model can also discover synonym relations, that could not be resolved using language alone.

Generalization driven by language At the same time, the interaction between both components can be reversed: Our component for learning generalized constructions from sequences of words could generate the hypothesis that both sentences can be merged into a more general construction *'X jumps over Y'*, abstracting from the specific slot fillers of the corresponding verbal construction. This mergeability of both utterances into a more general utterance would trigger the second action concept learning component to try to unify both action sequences into a more generalized action sequence in terms of a probabilistic model that still generates both sequences with high likelihood while not being overly complex.

Confidence

For a learning system it is crucial to have an estimate how confident the system is about it's acquired knowledge. This measure is very different from the confidence of the system about a certain prediction and does not necessarily have the same scale or (e.g. probabilistic) interpretation. The confidence about acquired knowledge is rather a measure that states how confident the system would be on average when faced with novel examples from already learned categories. The computation of this measure follows very different approaches in action and language.

Action The confidence measure of the action model is identical to the bayesian measure (see section 3.3.3) composed of a prior favoring simpler models in general and the likelihood estimate (see section 2.3) based on viterbi paths. This measure can simply be averaged across all models in the action component.

Language The component for language learning exposes a measure of confidence that estimates how accurate the mappings learned at different representational levels are. The confidence is based on a similar notion of relative entropy, not between models but relative to the maximum entropy. The intuition behind this measure is that, if a certain amount of information is acquired, this results in a reduction of entropy (or uncertainty) in the mapping between two sets x, y in the associative network. According to Gaspers and Cimiano (2014) the entropy of a certain mapping between form $nl \in x$ and it's associated meaning $j \in y$ is computed as

$$H(nl) = -\sum_{j=1}^{|y|} \omega'_{nl,j} \log \omega'_{nl,j} \tag{4.2}$$

where $\omega'_{nl,j}$ denotes the association strength between nl and j in the associative network. The network weights are normalized to sum to one. $H(nl)$ is then compared to the maximal entropy obtained by assigning equal probability to each meaning in y:

$$H_{max}(nl) = -\sum_{j=1}^{|y|} \frac{1}{|y|} \log \frac{1}{|y|} \tag{4.3}$$

A construction nl is considered to be learned if the relative entropy is below a threshold θ_E:

$$\frac{H(nl)}{H_{max}(nl} < \theta_E \tag{4.4}$$

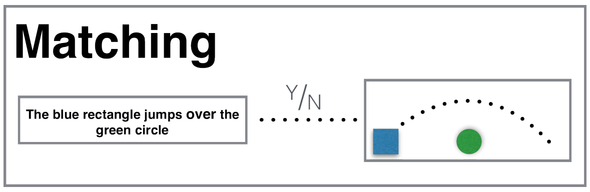
4.4 Experimental evaluation

Since we explore grounded language learning, we are interested in the system's generalization abilities both at the linguistic and conceptual (action) level. That is, the main goals of the system are to i) understand and generate novel utterances, and to ii) abstract over concrete trajectories of actions, in particular to also recognize actions performed by novel subjects. Thus, we consider two evaluation scenarios:

1. *novel-performer:* 12-fold cross-validation over all subjects, where the system is trained on data collected for 11 subjects and tested on the data of the 12-th subject.
2. *novel-utterances:* 25-fold cross-validation in which all utterances observed during testing are novel, i.e. none of them has been observed during training and thus cannot be understood or generated by performing rote-learning. Folds are generated by first collapsing data from all 12 subjects and then partitioning into 25 folds so that in each fold we have the same number of examples for each of the 4 action categories and 4 utterances which are not contained in any other fold. The sentence "the blue circle jumps on the green rectangle" could be exclusive for one fold while the sentence "the red triangle jumps on the green rectangle" could be included in multiple folds.

The developed system is evaluated in two different experimental settings, one concerning the understanding and one concerning the generation abilities. To measure the system's performance we compute precision, recall and F_1-measure (the harmonic mean of precision and recall). Recall is computed as the percentage of testing examples for which the system generates the correct result and precision as the percentage of correctly generated results of the number of testing examples for which the system actually generates a result (i.e. the system may choose that it cannot determine the result, for instance, because it has not been able to determine a suitable syntactic pattern and/or action model).
In order to estimate to what extent the system is able to detect synonyms for actions, we present a (mainly qualitative) analysis of the learned grammars.
In the following, we will first focus on language understanding abilities using a matching and a choosing test, and subsequently explore a language generation experiment. Afterwards, we discuss our results and put them into context.

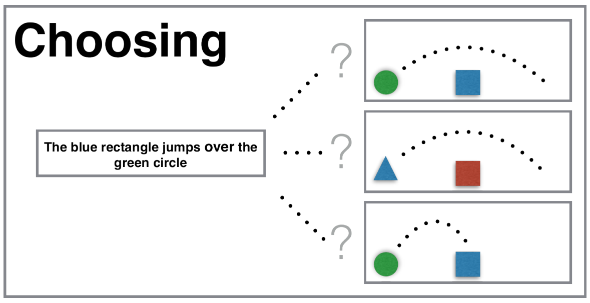
4.4.1 Matching test



In the first experiment, we evaluate the system's understanding abilities in a matching task. The test is depicted in the Figure above. The system receives as input pairs of utterances and action performances, presented as QTC sequences. The system has then to decide whether the utterance describes the action. These testing data are generated such that the action corresponds to the utterance in about 50% of the examples. More specifically, we keep the appropriate action for half of the testing examples and shuffle the action sequences for the other half such that the action does not correspond to the utterance. Any system has thus a 50% chance level of providing the correct response.

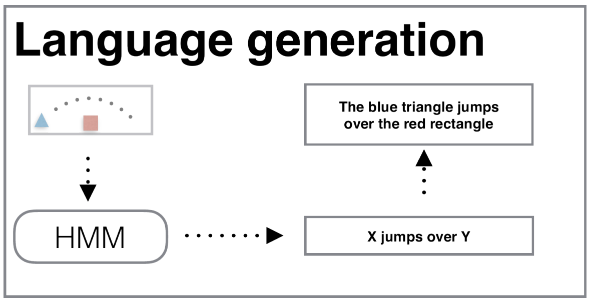
The matching test is implemented using our model as follows: given an input sentence, the system retrieves a generalized syntactic construction from the construction network that matches the input sentence. It then retrieves the associated HMM. If this HMM is the model that has the highest likelihood of generating the specific QTC sequence, then the system determines that the utterance matches the action.

4.4.2 Choosing test



This task is schematically depicted in the Figure above. When presented with an utterance paired with three action instances represented as QTC relations, the system has to decide which of the three actions the utterance refers to. Hereby, it is guaranteed that the utterance refers to exactly one action. The other two actions are confounder actions that either depict an action type that is unrelated to the utterance or an action of the same type as described by the utterance but with other objects. Given the sentence 'The blue circle jumps over the red rectangle', one example would indeed encode a blue circle jumping over a red rectangle, while the others would encode, e.g., a blue triangle jumping over a red rectangle as well as a green circle jumping on a blue rectangle. For this task we define the baseline at 33% which corresponds to the chance to pick the correct action randomly.

4.4.3 Language generation test



To evaluate the system's language generation abilities we first generate utterances for given testing actions. The utterances are generated by extracting all learned knowledge from the construction network – i.e. syntactic patterns, lexical units and groupings of elements – along with their associated HMMs and subsequently reversing the associations. For example, we might extract a pattern "X pushes Y" associated with a meaning comprising an HMM and the information what lexical units can occur in positions X and Y along with their corresponding role in the meaning (such as *trajector*). Given a testing action, the system first determines the HMM that has the highest likelihood of generating the sequence. Based on the grammar, it can then retrieve the corresponding syntactic pattern along with the information about lexical units and their roles. The generated utterance is considered correct only if it is identical with the example's actual utterance or the alternative utterance describing the same action.

We compare our results against a baseline that was established by choosing an utterance from the training data that has been observed with a similar meaning representation. Similarity is rated on both the involved objects (referents) and the action sequences. For the action sequences we implemented a simple matching score based on the Levenshtein distance between the compressed QTC_c sequences. For all pairs of trajectories t_1, t_2 we calculate a matching score as the Levenshtein distance normalized with respect to its theoretical upper bound $lev(t_1, t_2)/max(|t_1|, |t_2|)$. Because the distances are calculated over the compressed sequences it can also be considered similar to Dynamic Time Warping[1]. This baseline can however only yield matches in the *novel-performer* condition; in the *novel-utterances* condition none of the testing utterances has been observed during training and thus cannot be found by simply taking an utterance observed with a similar meaning.

[1]Dynamic Time Warping is an algorithm used to measure the similarity between temporally aligned sequences that may vary in speed.

Table 4.1: Results for the matching (language understanding), the choosing (language understanding) and the language generation test in the novel-performer and novel-utterance conditions with and without synonym detection.

Matching test

Setting	Synonym Detection	F_1	Precision	Recall
Baseline			50% chance	
novel-performer	No	75,42	75,42	75,42
novel-performer	Yes	99,08	99,08	99,08
novel-utterance	No	47,32	90,11	32,08
novel-utterance	Yes	86,50	88,69	84,42

Choosing test

Setting	Synonym Detection	F_1	Precision	Recall
Baseline			~33% chance	
novel-performer	No	67,70	100,00	51,17
novel-performer	Yes	99,33	100,00	98,76
novel-utterance	No	55,86	88,00	40,92
novel-utterance	Yes	80,62	92,00	71,75

Language generation test

Setting	Synonym Detection	F_1	Precision	Recall
Baseline		89% Levenshtein Distance		
novel-performer	No	68,67	68,67	68,67
novel-performer	Yes	74,00	74,00	74,00
novel-utterance	No	54,82	79,49	41,83
novel-utterance	Yes	64,09	67,63	61,08

4.4.4 Results

Results for all three tests along with their corresponding baseline values are presented in Table 4.1. The results reveal that the system achieves a large increase in performance over the random baseline, i.e. performing well above chance level, in both language understanding tests when synonym detection is active. Without synonym detection the system considers utterances as mergeable without considering if merging the grounded actions would yield consistent results as well.

For the **matching test** in the novel-performer condition, F_1, precision and recall are alike, since the system answers 'yes' if the HMM, retrieved for the sentence, has the highest likelihood of generating the observed action sequence, otherwise the system answers with 'no match', thus yielding an answer for each testing example. Since most utterances were parsed correctly (as indicated by high values for precision and recall), the system appears to have induced suitable grammar and action models in most cases, i.e. for most folds. The learned action models appear to generalize well to a novel performer for most human subjects. For the *novel-utterances* condition values are slightly lower, especially when the synonym detection is not used, which likely results from an insufficient determination of syntactic patterns, i.e. syntactic patterns may not have been learned before testing. When faced with utterances which are instances of unknown syntactic patterns, the system would not generate an answer, resulting in the low recall of 32% in the condition without synonym detection. But even in this condition, 90% of the actually generated answers were correct. With synonym detection the system is mostly able to respond even to unknown utterances, resulting in an F_1 score of 84%, which is clearly above the baseline of 50%. Taken together, the results are promising, showing a large increase in performance over the baseline. It is remarkable that in both, the novel-performer as well as the novel-utterance case, performance increases substantially when the synonym detection condition is active, by between 25% and almost 40% F_1-measure. The reason for the impact of the synonym detection is clearly due to the way the dataset has been constructed, replacing 50% of the utterances by synonym utterances that could not be reliably learned by language alone. Nevertheless, the results on the novel-utterance show that inducing synonyms by considering both sides of a grounding relation across modalities is indeed working very well.

On the **choosing test**, the system outperforms the baseline condition by large (67,70% and 55,86% vs. 33% on the novel-performer and novel-utterance conditions without synonym detection). Especially in the novel-performer condition the precision is 100%, indicating that if the system is given several potential meanings for an utterance and cannot determine the correct match it does not confuse the utterance with distractor meanings, even if these are also somewhat similar to the observed utterance, i.e. corresponding to the same action or involving the same objects. The impact of the synonym detection is also very large with increases in terms of F_1-measure by over 30% (from 67,70% to 99,33%) for the novel performer setting and close to 25% (from 55,86% to 80,62%) for the novel-utterance setting. Thus, taking the results for both tests together, the learned models appear to be suitable to yield generalized linguistic constructions and action models that generalize to unseen sequences as well as a reasonable discrimination ability between different actions.

In the **language generation test**, the system performs only slightly below the baseline in the *novel-performer* condition, showing that by merging observed action trajectories for several subjects into generalized action models the discriminative power is mostly retained. However, the learned grammar and models yield the additional benefit that the system is able to also generate utterances not observed during training. In particular, in the *novel-utterances* condition the system is still able to generate several utterances correctly, even though it has never observed them or their corresponding meanings before, which corroborates the generalization abilities of our model.

4.5 Discussion

We have presented a model that accounts for the co-emerge of linguistic constructions as well as corresponding grounded action concepts, mutually influencing each other to abstractly model the observed reality. At both levels, representation learning is performed in a bottom-up incremental fashion, unifying and merging specific instances into generalized representations that capture the essential characteristics of linguistic structures and action concepts that are ‘generative’ in the sense of being able to produce different surface forms. The model is based on similar representational concepts for both modalities in a sense that both modalities are represented in a graphical structure and learning is partly based on tracking co-occurence statistics between distinctive states in the course of an action performance or between words and their meaning. The model learns in an unsupervised fashion, such that categorical structure develops through similarity based unification of grounded structure across modalities.

The mutual influence of emerging representations across both modalities is bidirectional in our model. On the one hand, recognition of equivalent or mergeable linguistic structures drives the model towards merging/unifying action representations into equivalence classes, that is, into one generative model. This allows to learn the essence of action concepts denoted by action verbs. On the other hand, similarity in action models leads our system to the inference that two verbal constructions might indeed be synonymous, e.g. as in the case of *‘X jumps over Y’* and *‘X hops over Y’*.

To our knowledge, our model is the first model that depicts how linguistic verbal constructions and the action concepts they represent co-emerge, following similar principles relying on incremental generalization driven by the desire to yield more compact models that maintain predictive accuracy. Our model spells out these mechanisms in detail and thus provides a detailed implemented theory explaining how linguistic and conceptual development go hand in hand. Further, we provide a model that can both be used to ‘understand’ but also to ‘generate’ language. Our model allows a learner both to “talk” about observed actions, being able to categorize actions and verbalize them, but also to 'simulate' an action given an (input) utterance that describes the action. In this sense our model is one of the few (cognitive) models of language acquisition bringing also comprehension and generation together in the sense of Pickering et al. (2013). Further, it is the first model that depicts how synonyms emerge as a byproduct of grouping similar action models.

We presented a dataset that is on the one hand concise enough to probably facilitate discussion in emergentist linguistics but on the other hand is challenging enough to propose the model for further use in developmental robotics, e.g. in a scenario where the robot performs two dimensional manipulation operations on a table guide by language commands.

For action it is challenging to classify whether novel instances belong to an already formed category of action as, for instance, the *jumps over* action looks like the beginning of the *circles around* action and *jumps over* and *jumps upon* have a very similar overall trajectory. *Jumps over* and *jumps upon* mainly differ in their semantics in terms of the postcondition, i.e. in the first action the moved object lies on top of the landmark and in the other action the trajector is at the other side of the landmark. As the learning is completely unsupervised, the action model has to develop these categories autonomously by similarity based unification and by exchanging signals from the language learning component.

For language the challenges are similar in that the category formation is also organized in an unsupervised process based on linguistic similarity, exploiting invariances between similar linguistic constructions and compatibility to the grounded meaning (action models). Another challenging issue for language is synonym resolution to prevent from incorrectly building a category from sentences like “X jumps over Y” and “X hops on Y”.

In what follows we discuss possible connections of our work with respect to work on i) grounded cognition and in particular computational models of cognitive grammar, as well as, ii) linguistic relativity and the role of language in cognitive development, thinking and category/concept formation. We also discuss the relation of our work to current theories and models of language acquisition.

Grounded Cognition and Cognitive Grammar Our work is related to work that postulates that conceptual knowledge is grounded in modality-specific systems (Lawrence W. Barsalou 2008; Lawrence W. Barsalou et al. 2003)). As a special case of conceptual knowledge, language is also regarded as being grounded in perception and modality-specific systems (Pulvermüller 2013). In fact, our generative models are able to generate modality-specific simulations of perception and are thus inherently modal.

As learning proceeds, our system develops a grounded representation of the action denoted by the verb in form of a generative model, a Hidden Markov Model in our case. These Markov Models can be seen as intensional - vs. extensional - meaning representations that are grounded in perception and allow to perceptually ‘simulate’ the action denoted by the verb in question. Our HMMs can to some extent be seen as a specific implementation of the perceptual symbol systems proposed by Barsalou (Lawrence W Barsalou 2003).

Our work is also related to attempts to provide a simulation-based and embodied semantics for natural language. Feldman et al. have proposed X-Schemas as a way to capture the (embodied) meaning of a certain linguistic construction. The work by Feldman et al. on Embodied Construction Grammar (ECG) (Chang et al. 2004; J. A. Feldman 2006) is very related to our approach. However, the X-Schemas in which the meaning of linguistic constructions are represented are very symbolic compared to our qualitative models. Our qualitative models are still far away from a full grounding in the sensoric and actuator systems of an embodied system, but clearly go one step further than the X-Schemas used in Embodied Construction Grammar (ECG). The closest related work is the one of van Trijp et al. (Van Trijp et al. 2012), who have developed approaches by which robots can learn linguistic knowledge in the framework of Fluid Construction Grammar (FCG). However, as far as we know, they have not developed any approach that can actually induce these X-Schemas from observation. Further, the work of Feldman and colleagues has neither considered how synonym relations could be inferred by a system on the basis of detecting similarity between X-Schemas. This would indeed presuppose a notion of similarity between X-Schemas. Such a notion of similarity is inherent in our model, operationalized as 'unifiability' of two models. In general, there are to our knowledge no models that make detailed predictions how synonyms or near-synonyms are acquired.

Our work is related both to approaches to grounded acquisition of language in robots and cognitive systems, but also to approaches to the representation and acquisition of actions. With respect to approaches to grounded acquisition of language, there has been a lot of work on developing models which can acquire single words and their meanings (e.g. Fazly et al. 2010; Siskind 1996). In some approaches, this meaning is grounded in perception, but is typically limited to objects (Hsiao et al. 2008; Roy et al. 2002). Other approaches (e.g. Dominey et al. (2005) and Hinaut et al. (2014)) deal with the acquisition of syntactic constructions as we do, but typically do not ground these constructions in qualitative action models. With respect to the representation and acquisition of actions, different approaches based on prototypes (Kruse et al. 1997), Markov models (Sugiura et al. 2011) and neural networks (Droniou et al. 2014) have been proposed.

Many authors have emphasized the cognitive interaction between action and language (Anna M. Borghi 2014; A. Cangelosi et al. 2007; Glenberg et al. 2007; Willems et al. 2007). We have attempted to provide a detailed model that explains how action and language structures emerge in interaction, influencing each other.

Linguistic Relativity and Language as Enhancer of Cognitive Abilities Our proposal is further in line with the paradigm of linguistic relativity, corresponding to the claim that the language one speaks or hears influences one's own conceptualization and the categories one forms. While the strong claim that language determines thought has been largely abandoned (Philip Wolff et al. 2010), there is increasing empirical evidence showing that language influences thought. Wolff and Holmes (Philip Wolff et al. 2010) have described four ways in which language could influence thought, or more accurately, an emerging conceptualization:

- **Language as Meddler**: the effects of language on conceptualization or cognition occur by the spontaneous recruitment of linguistic codes and its meddling with observed non-linguistic categories to support decision making. Evidence for this comes from motion perception and color cognition domains.

- **Language as Augmenter**: in this case, language as symbolic system enhances our cognitive abilities by facilitating to conceptualize things that can not be purely grounded in perception or in a mechanistic/procedural routine. An example is conceptualizing larger quantities or understanding false statements, drawing spatial analogies.
- **Language as Spotlight**: having the effect of hinting at certain distinctions that are relevant in the given cultural context (such as grammatical gender, spatial frames of reference, spatial relations as well as the distinction between objects and substances).
- **Language as Inducer**, with the function of language as inducing people to conceptualize experience in a relatively schematic manner.

The understanding of language as a tool that enhances the computational and cognitive abilities of humans has lucidly been spelled out by Clark and Toribio (2012). In our account, language acts both as a spotlight as well as as an inducer of a schematic representation of experience. In fact, as suggested by Waxman and Markow (Waxman et al. 1995), language might serve as an invitation to form a new category. In our case, hearing an utterance for the first time leads to induce a category specific for that utterance. Generalization of several utterances leads to recognizing a schema and to induce a cognitive schema that represents the essence of the action category denoted by the generalized verbal construction. Language is thus playing the role of triggering the search for a schematic category, in our case a HMM that supports conceptualizing experience.

Category Formation There is indeed a lot of empirical evidence showing that language can facilitate category formation in the above sense. Xu (Xu 2002) found that the presence of distinct labels facilitated object individuation. Xu concludes that language may play an important role in the acquisition of sortal/object kind concepts in infancy and that words may play as *'essence placeholders'*. This is exactly what is happening in our model. On encountering an utterance the first time, our system creates a placeholder for the essence of this utterance. This early 'essence' is very specific for the given situation in which the utterance was heard and lacks any generalization. Latter, when hearing similar utterances, the corresponding essences are generalized by merging them into more general essences. Other researchers have shown that language can help to acquire the distinction between approachable and non-approachable creatures (Lupyan et al. 2007). Gentner and Boroditsky (Gentner et al. 2001) have suggested two processes that are active in learning the concepts that are denoted by words. They refer to *cognitive dominance* when concepts emerge from cognitive-perceptual processes and later the name for these categories is acquired. They refer to *linguistic dominance* when *'the world presents perceptual bits whose* *clumping is not pre-defined and language has a say how the bits get conflated into concepts'.* Clearly, as argued by Gentner and Boroditsky, this is not a dichotomy, but a continuum that spans a space in which the acquisition of a certain concept for a name can be located. According to Gentner and Borodtiksy, the acquisition of categories denoted by proper names or concrete nouns rather lies on the cognitive dominance side of the continuum. While kinship terms and verbs lie somewhere in the middle of the continuum, prepositions, conjunctions and determiners are rather positioned at the right side of the continuum. Our model explains the acquisition of categories towards the right end of continuum. The formation of such categories is triggered by the fact that learners are confronted with a new construction or name. It is certainly an open question where on the dominance continuum our specific actions such as pushing, jumping on, circling around are positioned at. This could be certainly determined experimentally. For the sake of providing a proof-of-concept for our model, we have assumed that the categories are not available previous to encountering the corresponding verbal constructions. While this is a mere assumption, our model is certainly not depending on that.

Emergence of meaning as a mapping Some researchers have criticized the 'mapping metaphor', that is the idea that language needs to be mapped to some priorly existing 'concept'. As mentioned in the introduction, Lila Gleitmann has emphasized that the assumption that language acquisition consists in the acquisition of (new) names for existing concepts is clearly an over-simplification. Our model overcomes this simplifying assumption by grounding concepts from one modality in corresponding concepts in another modality, starting with single multi-modal training items. Using similarity cues across modalities enables the model to incrementally generalize concepts even in conditions where categorical equivalence could not have been established within a single modality alone, i.e. when faced with synonyms.

Tomasello (Tomasello 2001) has criticized the 'mapping metaphor' on the grounds that it neglects that learning the meaning of words is actually a process of contextual inference in which the intentional structure of an action is considered to infer what the speaker is actually referring to. Rohlfing and colleagues (Rohlfing et al. 2016) have recently criticized the mapping metaphor on other grounds arguing that children would *not necessarily remember the connection between the word and the referent unless it is framed pragmatically*, that is, it is introduced in the context of a recurring interactional pattern with the purpose of achieving a joint goal between tutor and learner. While presented as an alternative to the mapping paradigm, they rather hypothesize that a communicative pragmatic frame facilitates to learn which concept a certain word evokes. The work of Rohlfing et al. and the proposal of pragmatic frames can be regarded as an elaboration of the general theory of Tomasello claiming that recognition of intention, shared attention and goals as well as the ability to simulate others as intentional agents are crucial ingredients by which children infer the meaning of a certain word or expression in context.

Part of the above mentioned criticisms on the mapping approach stem from the fact that the term 'mapping' is not clearly defined. As lucidly highlighted by McMurray et al. (2012), there are two notions of meaning: the referential meaning of an utterance or expression in a given situation and the intensional meaning of an expression. The referential meaning is inferred in a particular situation on the basis of an understanding of the situation. The intensional meaning corresponds to the situation-independent meaning of a linguistic expression, that is to its 'essence'.

Neither the theory of Tomasello nor the work of Rohlfing et al. make any predictions about how the intensional aspects of the meaning are learned over time and across situations as a byproduct of experiencing the word in different contexts. In a standard formal semantics paradigm, the intensional or situation-independent meaning aspects of a sentence can be captured using a truth-conditional approach, capturing the logical constraints that need to be fulfilled for the sentence to be true in a given world or situation. In this sense, our model tries to capture the essential meaning of a linguistic expression, albeit not using a standard truth-conditional semantics approach. In contrast, in our model intensional meaning is captured via probabilistic models that, instead of modelling logical conditions on the worlds in which the sentence is true, models the distribution or likelihood of observing a certain action sequence in a world or situation described by a verbal construction that is associated with the given HMM.

Most works dealing with the question how systems can learn the intensional meaning of a word have investigated how systems distill the meaning of a word by cross-situational learning, that is contrasting the different situations in which a word has been heard (McMurray et al. 2012; L. Smith et al. 2008). First, as argued above, referential and intensional meaning is often confounded. Second, according to our understanding of the notion of mapping, it does not imply that the corresponding concept pre-exists independently. In particular, our understanding of the term "mapping" does not necessarily imply that the category to which a word is mapped to exists already. In our approach, upon first encounter of a construction, it is 'mapped' to a maximally specific novel category that is created in the very moment in which the utterance and action are observed.

Nevertheless, the term mapping is not only problematic for the above reasons, but for the fact that it suggests that the meaning of words can be 'mapped' to a static symbol or handle. In contrast, we prefer to talk about *'evocation'* of a situation-independent meaning following the theory of Frame Semantics (Fillmore 1976) that postulates that words evoke more than just a static referent or handle, but more complex semantic frames, cognitive schemas or, in our case, grounded representations of the essence of the word's meaning that can be used to simulate. This is compliant with the view of Taylor and Zwaan (Taylor et al. 2009), who argue that: *'when a person hears or reads text involving action, there is activation of the motor systems in his or her brain, which corresponds to the referential semantic content of the description'.*

Regarding how a learner infers the meaning of a novel verb in context, Gleitman (see above) has proposed the syntactic bootstrapping theory according to which a syntactic construction (e.g. a transitive construction) can give cues about the potential meaning of a transitively used verb in context. However, for bootstrapping mechanisms to work, a learner needs to have learned a generic transitive construction and the general concept of *an agent that does something to another agent*. Gaspers, Foltz, et al. (2014) have provided an account for how such abstract constructions and categories might emerge. They presuppose that thematic roles such as *agent*, *patient*, etc. are already acquired.

Surely, syntactic knowledge as well as inference about the intentions of others plays an important role in inferring the meaning of unknown words in context. The recognition of intentions and of the teleology of actions emerges very early in childhood (Gergely et al. 2003). So far, our model is limited in that it neither models how pre-existing linguistic knowledge nor reasoning about the intentions of others or of the purposes of actions are factored in into the task of figuring out the meaning of a new verbal constructions. In fact, our model so far only considers the spatio-temporal structure of action concepts. The incorporation of these factors into our model is an obvious avenue for future work, albeit a very challenging one.

Analogy making Hofstädter and Sanders (Hofstadter et al. 2013) have recently argued that analogy making is a fundamental process in cognitive processing. According to their theory, category formation and language learning is driven by analogy-making, that is by fitting previously acquired categories to a given observation and then extending the category to the new observation, leading to generalization, which can be possibly perceived as an over-generalization for mature systems the language of which is heavily influenced by conventions. Take the example of the utterance *'I undressed the banana'*, which for mature speakers is an overt over-generalization, while from a cognitive perspective it makes a lot of sense. In some sense, this is what our model is making: it is constantly making analogies. It induces very specific action categories for a very specific utterance and then, when observing a similar utterance, it attempts to extend both the linguistic construction and the category to cover the new observation as well. In doing this, it is guided by the desire to minimize the complexity of the model, that is the number of bits needed to store the action model, while not loosing predictive power, that is not over-generalizing. This mechanism could be regarded as a direct implementation of the theory of Hofstädter and Sanders. We have provided the proof-of-concept in simulated experiments that this principle works for the case of simple verbal constructions denoting actions in which some object (an agent) moves with respect to some reference object. Whether our principles could be extended to other constructions and more complex cognitive models and simulations is an open question. However, there seems to be no principled reason why our approach would not extend to other categories. Arguably, HMMs would not be able to model all kinds of representations. But we stress that our proposal is not specific to HMMs. Our proposal requires that there is some (probabilistic) generative model that represents some (induced) category. For dynamic categories such as (action) verbs that involve a spatio-temporal signature (what linguistically is called often 'path') surely a sequential model will be needed, while for other static categories (e.g. objects denoted by nouns) a static prototype as simulation might be sufficient. Most likely, however, even object representations have a non-static meaning component what the representation of the potential to act on is and which objects are concerned. The latter has been often referred to as so called 'affordances' (Gibbson 1977). In terms of the theory of Hofstädter and Sanders, our generative models capture the *'essences'* of concepts of which we see a specific realization or *'surface'* in the real world.

5 Learning to act by curiosity driven self exploration

Many theories of early language acquisition rely on basic perceptual capabilities such as the ability to segment speech signals into separate tokens (e.g. phonemes or words) or the ability to interact with the environment through basic manual skills. While this is no principled limitation to the understanding of early language acquisition, however, it can be assumed that early concept and motor skill acquisition that accompanies the pre-verbal language acquisition has a relevant impact on how we communicate.

Extending computational models of early language acquisitions such that they encompass the early concept acquisition in the pre-verbal phase may reveal findings that could lead to better understanding of the impact that the pre-verbal phase has on the shape of linguistic representations and constructions.

In the previous experiment we investigated how linguistic constructions could be developed from parallel examples of action and language assuming that the learner already developed the perceptual capabilities to observe language and action. The system learned purely by observing actions involving two objects performed by human participants of a study in reaction to an instructive sentence that described the action to be performed.

This experiment is concered with exploratory self-action and the observation of consequences to learn a joint model that covers early perception and action cabability. Many attempts exist to cover very early phases of learning where only little pre-existing perceptual and conceptual knowledge can be assumed to be available. Also the capacity to interact with the environment is only rudimentarily developed at that stage. The early phase of concept and skill acquisition can be modeled by employing childlike robots that provide the learning system with similar experiences that human children of comparable age are exposed to when exploring their environment. Equipping the learning system with a fully embodied robot system goes beyond simple cross-modal learning, since modality overarching experience can not only be consumed but rather the experience can be actively shaped through the interaction with the environment. However, embodied systems are still very expensive and thus unavailable to a wide group of researchers.

Simulations can fill the gap in research that targets specific independent aspects of developmental processes or in situations where a physically embodied system would introduce unnecessary complexity or world noise[1].

In this chapter we exemplify in simulation how a very simple intrinsic motivation signal such as the desire to induce change can lead to the learning of complex behavior in an environment where a learning system explores the properties of a complex articulated object and its own capacity to interact with that object. We show that this very simple signal can drive the exploratory learning process into behavior that can be termed as "playful exploration" when visually examined.

There are many theories to how children acquire the knowledge how to induce change in the world in order to achieve a goal. We employ the behaviorist view, which postulates that there is only minimal innate structure infants start with and that they have a simple capacity to generalize in order to improve their abilities. Huttenlocher et al. (1983) argue that in this view infants action categories should involve only perceptually similar stimuli and categories get reinforced by common responses (e.g. physical effects, linguistic designation).

This experiment is related to cognitive and developmental robotics in that we are interested in developing models that allow a system to self-familiarize with a non-trivial articulated object by exploring a huge continuous space with discrete action possibilities guided by the desire to induce change in the object. The model is developmentally inspired in the sense that a system learns action categories by performing actions on an object and observing the consequences, relying on intrinsic motivation to induce change. In fact, some authors in the field of developmental psychology have argued that young infants have a propensity to systematically explore the connection between their own actions and the perceptual consequences in order to support inter-modal calibration of their bodies (Rochat 1998).

We realize the propensity to systematically explore the connection between own actions and their perceptually observed consequences through a bias that leads learning systems to attempt to induce change to maximize encounters with objects in which something 'happens', thus maximizing the situations in which the system can learn something about the effects of its own actions on the environment. We incorporate this bias for inducing change via a reward function that rewards a learning system for inducing movement in an object that the system is familiarizing with.

We implement the system as a deep reinforcement learning model (see section 2.7) relying on a neural network to compute the action-value function in terms of expected discounted future reward. As a reward is needed to guide the search for an interesting policy, we investigate the structure of rewards that are suited to guide the effective

[1]World noise is an often used term to refer to uncertainty and sensor noise introduced by real environments that are not as highly controlled as laboratory setups or physical simulations. Complex physical platforms usually suffer from world noise introduced by to some degree unpredictable response characteristics of actuator and sensor systems.

self-exploration of a system towards understanding the effects of their own actions on an object having the sole goal of inducing movement on the object.

5.1 Learning with intrinsic motivation

In this section we briefly introduce the concept of intrinsic motivation with regard to curiosity driven exploration. A comprehensive conceptual overview on how intrinsic motivation is applied in cognitive robotics as an inductive signal that drives the learning process can be found in Cangelosi et al. (2015).

Intrinsic motivation as a learning signal is often used in autonomous learning where the agent is free to choose what to learn based on its own agenda. In psychology, intrinsically motivated behaviors are often defined as actions that are chosen by the organism "freely" without having external incentives or consequences. In contrast, extrinsically motivated behaviors are produced in response to external signals or cues.

In autonomous learning, there are no external primary (goal driven) training targets given to the learner. Instead, the learner has to actively explore its environment in order to elicit interesting signals or observations to learn from. In order to effectively explore the environment, the agent needs a strategy that guides the exploration process (see section 2.7). In essence, the exploration strategy has to decide which actions to take based on experiences received from the environment and to weigh the available options in terms of their utility with regard to the goal or with regard to what can be learned when taking a novel path. The trade-off between reaching a defined goal and exploration of novel options is termed the "exploration exploitation dilemma" and is introduced in subsection 5.2.1. Intrinsic motivation provides a basis to formulate goals such that the system is not only focused to achieve a certain well defined goal but is rather focused on the goal of learning itself and provides the system with a form of *artificial curiosity*.

5.2 Learning problem

The experiment in this chapter is performed in simulation. However, the design of the experiment was inspired by a robot that tries to explore meaningful ways to manipulate objects lying on a table in front of it. This research was embedded in the FAMULA[2] project (*Deep Familiarization and Learning Grounded in Cooperative Manual Action and Language*), which is concerned with developing approaches by which a bimanual robotic platform can acquaint itself with the properties and affordances of objects by (guided) autonomous self exploration. The platform

[2]https://www.cit-ec.de/en/deep-familiarization-and-learning

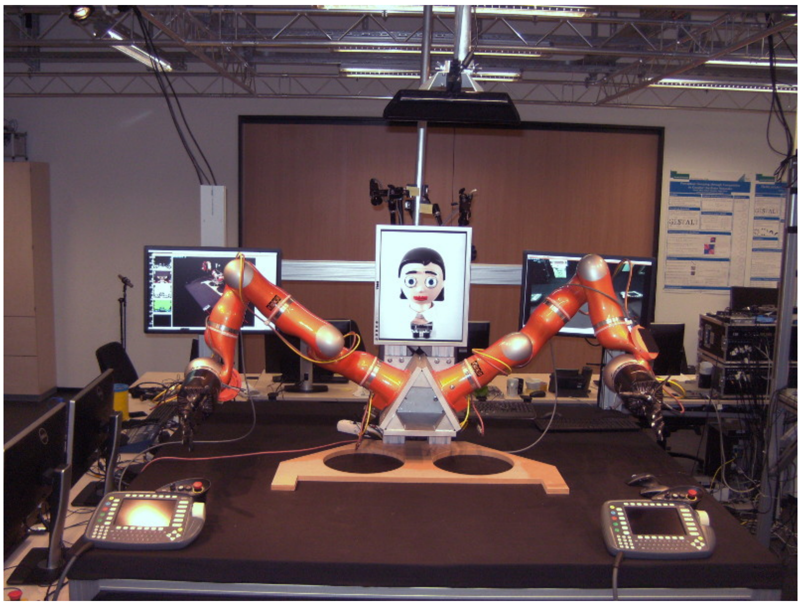


Figure 5.1: The FAMULA platform: A bimanual robotic platform supporting the investigation and development of approaches by which a cognitive system can familiarize itself with new objects and concepts through the interplay of manual action and language (`https://www.cit-ec.de/en/deep-familiarization-and-learning`).

combines rich bimanual actuation with anthropomorphic hands allowing for dexterous object manipulation (Figure 5.1).

Due to availability constraints and considering that having to monitor the robot personally during the long-running experiments would have seriously limited the amount of experiments that could have been carried out, the experiments were performed in simulation rather than on the physical platform. However, we assume that findings from experiments in simulation will prove qualitatively comparable to experiments on the platform that are scheduled for future work. In the simulated environment, the actuator is modeled after the tip of a finger of a robotic hand sliding over the table in front of the robot. To evaluate the exploration driven object familiarization, we placed a complex object with rich physical dynamics in the center of the simulated table (see Figure 5.2). The object was modeled after a toy clock that is available as a physical object and is robust enough to perform follow-up experiments on the real platform.

The robot can explore its simulated environment (Figure 5.2) by taking physical action and observing the results of its actions in terms of change in the environment and a sparse reward signal indicating how well the robot performed with respect to a measure of object-familiarization (see subsection 5.4.1 for the definition of reward signals). The simulated table provides rich and accurate physics with dynamics that

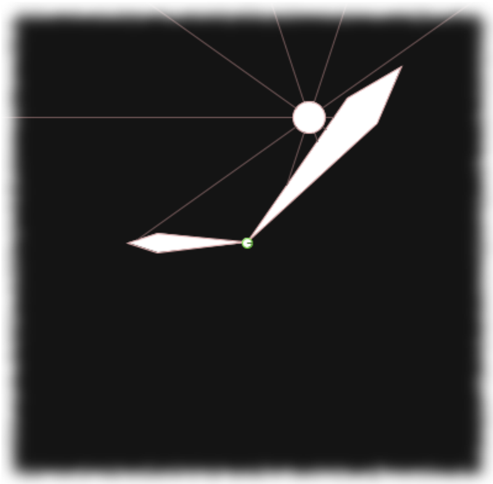


Figure 5.2: A simulated clock-like object consisting of two hands rotating around a centre. A simulated fingertip can move the object and observes its environment through 10 star-shaped distance scans to other objects or the boundary of the operation space.

are not only reactive in a sense that changes in the environment only happen in response to actions the robot takes, but reactions can also sustain for a longer period of time due to, e.g., the interplay between inertia and friction in moving objects. The robot can not observe the full state of the simulation at once, but instead has to infer the state of the simulation from partial observations in terms of a two dimensional circular distance scan to objects in its current vicinity.

The simulated clock has one large and one small hand which both are freely movable on a circular path around the clock center. A key challenge for the robot in learning the motion dynamics of the clock lies in the special non-euclidean way the clock hands move. In order to build a forward model which projects the current state to future states of the moving clock hands, the system has to learn to map the euclidean coordinate system it operates in into the polar coordinate system in which the motion of the clock hands can be described linearly. The simulated clock has some more interesting properties for the robot to discover. One side of a clock hand is fixed to the clock center and can not be moved, while the other side is freely movable. Applying a force to the outer side of a clock hand makes it easier to induce movement than applying a force somewhere in the middle of the clock hand due to static friction and the law of the lever.

Actions

To interact with its environment the robot can take one out of 6 actions including actions that accelerate the simulated fingertip in one of 5 equally partitioned directions

$(0, 72, 144, 216, 288)$ and one additional action which does nothing and can be used by the robot to fine-control the movement speed of the fingertip or the force applied to another object. Consequent to the actions being inert accellerations rather than direct movements and due to the high control frequency of 10 controller updates per second the fingertip can render smooth movements by blending the available discrete actions.

Observations

The robot observes its environment through 10 circular distance scans (Figure 5.2) scanning the vicinity of the fingertip. The distance scans have been implemented as raycasts emanating from the fingertip's center position measuring the distance to the first object they reach. This observation scheme is consistent with the state representation of the robotic platform which has an internal model of object positions and extents allowing to steer the fingertip even when the hand blocks the sight of the visual system. The fingertips are also equipped with force sensors that detect when the robot has contact with another object. Having contact with an object would in this setting correspond to at least one of the distance scans being zero.

Reinforcement learning with neural networks has been shown to work well also when the input to the system comprises only raw visual information (Arulkumaran et al. 2017). Using a distance scan as input is presupposed by followup experiments that are planned to be conducted on the physical platform. However, we assume that learning from visual input would primarily affect the training time and has a lesser effect on the task of developing basic motor skills to interact with complex objects.

5.2.1 The Exploration/Exploitation dilemma

Exploring the environment at random will eventually converge to the optimal policy, as valuable states get higher reward on average. However the state-action space can get arbitrarily large. Visiting every trajectory[3] possible in that space is often infeasible in finite time. This is often approached by exploration strategies that perform random walks and thus sub-sample the space of possible trajectories. Utilizing already acquired information about the value of state-action pairs can focus the exploration to the most promising trajectories. There is a variety of strategies that organize the exploration process based on different notions. One of the simplest and most often applied strategies is the ϵ-greedy exploration. It chooses a random action with probability ϵ or the best action, according to the current learned policy, with probability $(1 - \epsilon)$. Usually the exploration factor ϵ is annealed over time to privilege exploitation of already gained knowledge over random exploration.

[3]A trajectory is a list of successive states visited and actions chosen in those states coupled with the received reward

The *exploration/exploitation dilemma* is fundamental to reinforcement learning and represents a trade-off between slower convergence but higher chance of concluding with the optimal policy on the one hand and faster convergence to a probably sub-optimal policy on the other hand.

In deep Q-learning the approximation of the Q-function becomes increasingly accurate while the training proceeds and the exploration can be focused on higher rated paths. As the ϵ-greedy strategy does not consider acquired knowledge in the decision of whether or not to take a random action next (other than fixed annealing factors) we also investigate an exploration strategy that utilizes the uncertainty in the value assessments of available actions.

5.3 Model

Model free early motor skill acquisition is implemented as a reinforcement learning approach where the agent has minimal prior knowledge about the environment it operates in or about the extent and shape of its own physical manifestation.

Reinforcement learning is a machine learning framework where learning is driven by the desire to maximize some notion of cumulative reward R (Equation 5.1) by iteratively exploring the state-action space of the learning environment through performing actions and observing changes in the environment as a result of the robot's actions. Q-learning (Watkins et al. 1992), a form of temporal difference learning (Richard S. Sutton 1988), is one of the most popular reinforcement learning algorithms but it quickly becomes infeasible when the discretized state-space becomes large due to the curse of dimensionality (Lillicrap et al. 2015). Further, a naive discretization of state and action space can hide information that may be crucial to solving the task. In our model we build on Deep Q-Networks (Mnih, Kavukcuoglu, Silver, Graves, et al. 2013), which replace the discrete Q-function with a continuous deep neural network, thus acting as universal function approximator, with Double Q-Learning (Hasselt et al. 2015), which improves performance by limiting the overestimation of action-values conventional Q-learning is known for. Deep Q-learning problems consist of three components.

- A scalar **reward function** which gives the system positive or negative feedback $r \in \mathbb{R}$ on how well it performs. Future rewards are discounted by a factor γ per time-step, giving precedence to current over later rewards.

$$R = \sum_t \gamma^t r_t. \tag{5.1}$$

- A **policy** π which determines the optimal action given the current state-action value assessment.

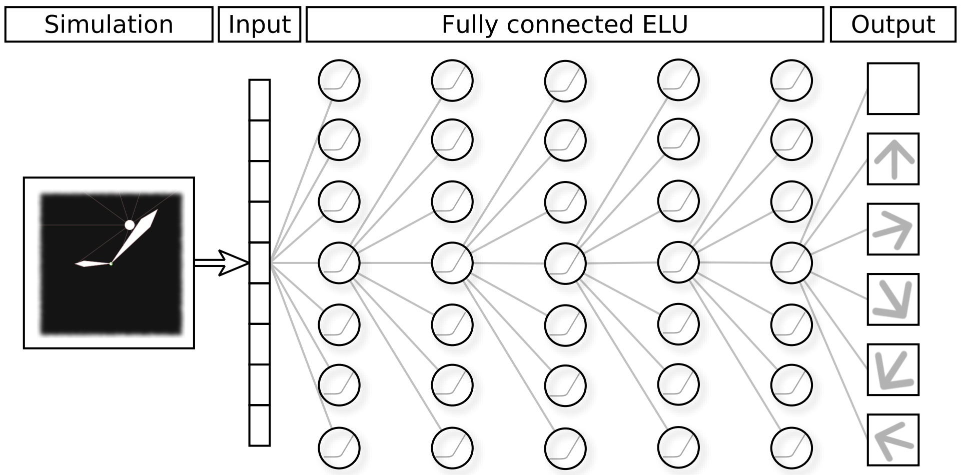


Figure 5.3: Layout of the neural network. The robot perceives its environment in terms of a 5-fold circular distance scan yielding a vector of 5 real valued distances to the closest object (including border of operational area) in the corresponding direction. Two successive observations are concatenated into one 10 dimensional vector which is fed as input into the 6-layer network with 96 nodes per layer. Each node in the network is fully connected to all other nodes in the preceding layer. The network uses exponential linear units (ELU) in the first 5 layers and linear units in the output that projects the network output into the space of expected long-term rewards.

- An **action-value function** which assesses the value of the current states' partial observation $s \in \mathbb{R}^d$ in terms of expected discounted future reward a robot can maximally achieve when taking action a in current state s and following the policy π afterwards

$$Q_\pi(s,a) = \mathbb{E}[r_t + \gamma r_{t+1} + ... + \gamma^n r_n | s_t = s, a_t = a]. \tag{5.2}$$

Q network

To estimate the Q-function we implemented a 6 layer neural network (Figure 5.3) similar to the deep q-learning network (DQN) in (Mnih, Kavukcuoglu, Silver, Graves, et al. 2013), which estimates the expected discounted future reward for each of the 6 actions. For deterministic policies we can reformulate $Q(s,a)$ (Equation 5.2) recursively by the so called Bellmann equation:

$$Q_\pi(s,a) = \mathbb{E}[r_t + \gamma Q(s,a) | s_t = s, a_t = a] \tag{5.3}$$

Instead of having a second parameter to the neural network along with the state vector, the Q network $Q(s,a;\theta)$ is implemented to directly output the value of all 6 actions at once $Q(s, \cdot; \theta)$, where θ are the current parameters of the network. This saves 5 additional forward passes per iteration and is equivalent to calculating the value of each state-action pair separately because in this setting all actions are possible regardless of the current state. The target for deep Q-learning (Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015) is

$$Y_t = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-). \tag{5.4}$$

This training target employs a separate target network θ^- which is structurally equivalent to the online network except that its parameters are a periodic copy of the parameters of the online network every τ time-steps. A separate target network is used to decouple the update from the target Q-values that are used to compute the training loss and thus reducing the probability for the training to fall into a feedback loop between the target and the estimated Q-values. Because of the max operation in the target, Q-learning and DQN tend to overestimate action values (Hasselt et al. 2015). We adapt the idea of Double Q-learning (Hasselt 2010) and decompose the max operation in the target into action selection and action evaluation. The update for Double DQN uses the online network (θ_t) for action selection and the target network (θ_t^-) for action evaluation under a greedy policy

$$Y_t = R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_t), \theta_t^-). \tag{5.5}$$

As activation functions for the inner layers we use exponential linear units (ELU) (Clevert et al. 2015)

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \tag{5.6}$$

instead of the originally proposed ReLU

$$f(x) = \max(0, x) \tag{5.7}$$

activations because we found that due to the sparse but strong updates in this setting, the activations of up to 25% of the network became constantly negative. ReLUs can not recover from constantly negative input because the gradient is zero for $x < 0$, which leads to vanishing updates.

To enable the network to estimate dynamic properties such as the speed of the fingertip or the clock hands we provide not only the current observation but also the observation of the last time-step as history. As input the last two observations are concatenated

$$s_t = o_{t-1} \oplus o_t \tag{5.8}$$

and normalized to zero mean and unit variance with per-batch normalization (Ioffe et al. 2015). We trained in batches of 32 examples sampled uniformly from buffered experiences which consists of transitions from the last 1500 time-steps. This so called 'experience replay' breaks correlation in strongly correlated data such as time-series

Method	$score_{max}$	$score_{avg}$	σ_{avg}
Goal derived reward	47	38	4.0
Goal derived + tutoring	69	53	6.8
Goal derived + tutoring + penalty	65	59	3.16
ϵ-greedy exploration	46	39	3.5
Boltzmann exploration	65	59	3.16
No regularization	60	53	7.4
L_1 Regularization	65	59	3.16
Dropout (ρ=.5)	10	5	1.8

Table 5.1: Evaluation of method combinations with their score (percentage of time-steps the object was moving) and standard deviations. Bold results represent the same configuration

in reinforcement learning by storing samples in a fixed-size buffer and training on random (mostly non-successive) subsamples. We use Adam (Kingma et al. 2015) as optimizer, which combines ideas from RMSProp (Tieleman et al. 2012) to deal with non-stationary objectives and AdaGrad (Duchi et al. 2011) to deal with sparse gradients as they particularly occur in reinforcement learning.

5.4 Experiments

To evaluate the ability of the system to develop basic motor skills and familiarize itself with a complex object like the toy clock, we trained the system in various conditions and evaluated the system's performance by measuring the percentage of time-steps within an episode in which the clock hands were actively or passively moving. Active movement in this case is caused by the agent applying a force to the clock hands, while passive movement is due to inertia and friction. In other words, the agent's task was to learn to induce as much change as possible in the environment.

Policies learned by reinforcement learning are shaped by the reward signal, with which the agent can structure its perception-action mappings in a goal-directed manner. In other domains like e.g. playing atari video games (Mnih, Kavukcuoglu, Silver, Graves, et al. 2013), the reward signal is determined by the game and was specifically designed to give frequent rewards to players to motivate them to keep on playing and improve their gaming performance. The simulated environment in this experiment has no intrinsic reward structure. The agent starts without any prior knowledge about the environment or about itself and its own capabilities, except that in each time-step it can choose one out of 6 different actions, for which the agent has no prior information about their possible effects.

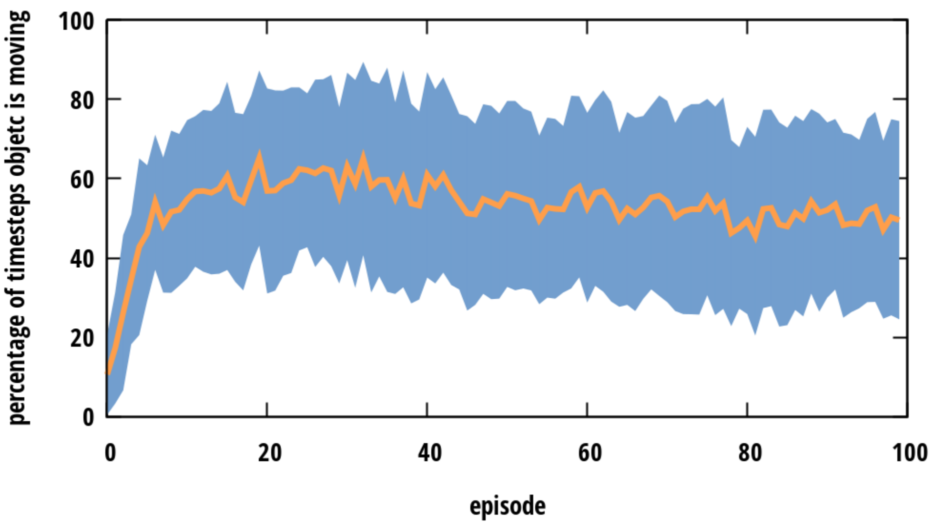


Figure 5.4: Evaluation of the systems score after every training episode with goal derived, tutoring and penalty rewards in combination with Boltzmann exploration strategy and L_1 regularization of the network. The shaded area corresponds to one standard deviation from the average across 25 evaluation runs

To discover structure in the observations, the agent has to explore the environment by taking actions, observing consequences and assess them with regard to the reward received. In the following experiments we evaluate how different reward structures shape the exploration process in conjunction with different exploration strategies.

In each of the experiments we trained the network for 100 episodes, where every episode is a complete run of the simulation for 1500 time-steps. To mitigate the impact of randomness from the random exploration strategies, we ran each experiment 25 times and calculated the mean scores (time-steps the object was moved) per experiment. The network topology and hyperparameters have been optimized separately by a combination of manual optimization and randomized grid search.

Table 5.1 lists results for different combinations of reward signals, exploration strategies and network regularizations. The scores are given in two conditions, $score_{max}$ denotes the peak performance when selecting the best model among the 25 runs of the experiment, while $score_{avg}$ denotes the performance averaged over all of the 25 experiment runs. The peak performance ($score_{max}$) corresponds to a setting where the agent can be trained multiple times in a separate process (off-task) and the best performing model is deployed afterwards. The average score ($score_{avg}$) denotes the expected score when the agent has to be trained on-task, which is probably the most common case, and σ_{avg} denotes the standard deviation over all 25 runs. Bold results in the table correspond to the same configuration displaying the best average performance. Figure 5.4 compares the performance of the best model configuration along with the standard deviation across different evaluation runs.

5.4.1 Reward Structure

The reward the agent receives from its environment is the primary driving force of the learning process. We investigate different strategies to structure the reward signal. As the simulation is compared to e.g. atari games rather unconstrained and offers many degrees of freedom, we additionally introduce small tutoring reward signals for the agent to learn promising exploration directions. Without tutoring signals the rich state space has proven to be too large for pure random exploration to converge in acceptable time with deep Q-learning.

Tutoring reward ($r = 0.15$) is given with 10% chance when the agent moves towards one of the clock hands. Goal derived reward with magnitude $r = 1$ is given every time-step the agent achieves to move a clock hand. This reward is directly derived from the performance measure and is maximal when the clock hands move at every time-step. Penalty rewards are given when the agent tries to leave its operating area on the table ($r = -1$) or when the agent stalls for more than 10 time-steps ($r = -0.15$). This also helps to pre-structure the exploration space towards the center of the operating area. Goal derived reward alone seems to be insufficient to learn a good policy. Goal derived reward in combination with small tutoring rewards improves the performance of the learned models significantly and achieves the highest peak score of all models. But as the peak score is prone to random effects from the exploration strategies, we put more weight on the average score, which is best when all three reward signals are combined.

5.4.2 Exploration Strategy

In the domain of atari games the ϵ-greedy exploration strategy proved valuable to find a good trade-off between exploration and exploitation of already learned moves. This policy simply chooses the best action available according to the current action-value estimates

$$\pi_t = \operatorname*{argmax}_a Q(s_t, a) \tag{5.9}$$

for the current state s with a probability of $1 - \epsilon_t$ or a random action with probability ϵ_t. The exploration factor ϵ is initially set to 1 and linearly decays to 0.01 within the first 10 episodes of training.

In contrast, Boltzmann exploration (Richard S Sutton et al. 2017) is an exploration scheme that takes relative action values into account. In contrast to ϵ-greedy it is adaptive in a sense that in states with clearly advantageous actions it behaves like a greedy strategy and follows optimal actions, while in states where the action-values are similar, the Boltzmann exploration introduces more randomness. Similar action values could arise in states that are universal in a sense that it does not matter much which action to take. Similar values can also arise in poorly explored states where the network is uncertain about which action is the best action to take. Boltzmann exploration is similar to drawing samples from a softmax distribution over the Q-values but takes also a temperature factor T into account which is annealed over the first 10 episodes of training to privilege exploitation over exploration:

$$\pi_t(a) = \frac{\frac{e^{Q(s_t,a)-Q(s_t,a_{max})}}{T_t}}{\sum_a \frac{e^{Q(s_t,a)-Q(s_t,a_{max})}}{T_t}} \tag{5.10}$$

where $a_{max} = \operatorname{argmax}_a Q(s_t, a)$. With temperatures close to zero, this exploration scheme behaves like Equation 5.9, while with higher temperature values it selects actions more randomly. As can be seen in Table 5.1, the Boltzmann exploration scheme clearly outperforms the ϵ-greedy exploration strategy in this setting. One observation from visually inspecting the simulation is that ϵ-greedy explores only a limited area of the operation space because the actions available to the agent are contradictory and selecting them randomly cancels their effect in average. Boltzmann exploration on the other hand takes already learned action values into account when randomly selecting actions, which, especially in combination with tutoring rewards, leads to early movements towards the object and better average evaluation score.

5.4.3 Regularization

In this experiment we investigate the impact of different regularization methods on the performance of the system. One observation while training different models was that even when the simulated fingertip was close to a clock hand the Q-values did not differentiate well. The expectation was that when the current state offers a clear opportunity to receive reward, the values of actions bringing the fingertip closer to a clock hand would lead to clear peaks in the action value assessment of the Q-network. But even in those opportunistic states the Q-values were close to each other and hence the best action was prone to fluctuate during training.

To measure the closeness of action values we adapted the idea of entropy over distributions by first performing a softmax normalization and calculating the entropy as the expectation over the information content

$$H(A) = -\sum_a p(a) \log p(a) dx \tag{5.11}$$

as a measure of surprise in the action values. We found that applying L_1 regularization improved differentiation of Q values in opportunistic states and also improved the behavior of the robot when visually inspecting the simulation. However, this effect is not well reflected in the performance measure applied in these experiments but could be useful for tasks with other goal definitions. The regularization strength ($\lambda = 0.1$) was optimized by grid search. L_2 regularization was also applied but performed minimally worse than L_1. Another often applied regularization technique in deep neural networks is dropout (Srivastava et al. 2014). Dropout randomly drops nodes from the network with a certain probability during training and thus simulates the training of many smaller subnetworks, which is similar to the idea of ensemble methods. This technique was proposed to mitigate overfitting by limiting the co-adaptation between nodes in the network. Dropout forces the network also to develop a distributed representation of inputs rather than having parts of the network specialize on specific stimuli, because when randomly dropping nodes, later units cannot rely on input from earlier units in the network. Interestingly, applying dropout with $\rho = 0.5$ in this setting lowered the performance of the robot drastically (see Table 5.1), which suggests that there are parts of the network which tend to specialize on specific (probably location specific) observations. The effect persisted even when granting more capacity to the network to compensate for the dropped units during training. The tendency to develop location specific representations could be investigated further in followup experiments.

5.5 Discussion

We exemplified how a very simple curiosity signal endows an agent to engage in guided self-exploration of basic perceptual and motor skills with the goal of understanding the consequences of their own actions on the given object. According to (Huttenlocher et al. 1983), young children learn categories that are either specific to self-action or observed action. This experiment corresponds to learning through exploratory self-action guided by the desire to induce change as an intrinsic motivation. We have proposed a reinforcement learning based model using deep Q-networks to compute the state-action values as cumulated future reward. We have applied the model to a simulated robot environment consisting of a robotic “fingertip” that can slide over a simulated table-like surface and manipulate the arms of an analog toy clock. We have shown that deep Q-learning is a suitable framework to support robot object familiarization and the development of basic perceptual and motor skills that could be used to further reduce the assertions needed as prerequisite for the development of models that investigate the early phases of language acquisition.

Our work is related to deep learning models that learn to play video games in that we use a deep Q-learning model similar to the one proposed by Mnih, Kavukcuoglu, Silver, Graves, et al. (2013) with additional double Q-learning (Hasselt 2010; Hasselt et al. 2015) and a Boltzmann exploration strategy instead of the originally proposed ϵ-greedy policy. However, it also crucially differs from this line of research in that we are concerned with learning in environments that have less inherent structure than video games that have been specifically designed to give frequent reward to keep human players motivated. In contrast to the aforementioned approach, we do not use convolutional layers to learn representations directly from pixels. It has been shown many times that convolutional layers perform fairly well in related tasks so that in this experiment we focus on exploration and reward structure and represent observations in a way that is more suited to robot object manipulation where the vision is mostly occluded by the robotic hand. Singh et al. (2004) investigate intrinsic reward signals in a discrete “playroom” environment where the robot can explore the interactions between different objects such as a light switch, a ball, a bell, movable blocks, etc. Most of the affordances in the environment only apply in interaction between the objects, e.g. the bell rings if the ball is kicked onto it. Intrinsic reward is only generated by unexpected salient events given as error in the prediction of action outcomes. Schmidhuber (1991) proposed to employ curiosity and boredom as confidence-based rewards signals, where curiosity is used as a driving force to explore areas of the action-state space where the predictive model has low confidence.

6 Summary

We have presented a computational model of language acquisition that models the joint emergence of linguistic constructions and the conceptual categories they evoke. The model is motivated by emergentist theories of language acquisition and transfers the underlying ideas also to the domain of action learning. The connection between the theoretical foundation and the technical implementation in terms of the proposed model is bidirectional in this regard. On the one hand, the technical properties of the model such as the fully incremental and data-driven approach to learn concepts within a modality that are grounded in concepts with similar semantics of another modality are relevant in many technical applications such as developmental robotics and human-computer interaction (see section 3.5). On the other hand technical implementations that are closely based on key concepts of theoretical frameworks can as well serve as computer implemented theories that allow other researchers to interactively test hypothesis that arise from the theoretical foundation. In order to facilitate the proposed model in follow up research we detailed possible connections to ongoing interdisciplinary research and related work in the discussion in section 4.5. In the following we briefly summarize how learning in the model proceeds before we answer specific research questions that structured the implementation of the model.

In line with 'slow mapping' approaches that claim that the acquisition of the meaning of a word is a long-term process that starts with the first encounter (S. Carey 1978; McMurray et al. 2012), we have provided an account of word learning in which, upon the first encounter of a linguistic construction or expression, a learner induces an ad-hoc category that is specific for the given utterance and context. From many encounters of similar utterances, learners distill the essence of the category evoked by the word through incremental generalization, guided by the desire to produce compact models while not reducing the accuracy of the predictions made by the model on sequences observed so far. Concepts in our approach are 'essences' evoked by the linguistic constructions and are generative models that can generate various 'surfaces'. The goal of a learner is to distill these essences from the many encounters with a given word or linguistic construction. In doing this, they incrementally stretch and extend the meaning of early categories to subsume other early categories by a process that could be understood as one of constant analogy making and extension of categories. Our proposal is thus very much in line with the understanding of cognition as analogy making.

We have looked in particular at the case of verbal constructions and corresponding actions in which a trajector moves with respect to some reference object along a characteristic spatio-temporal path. As actions are temporal sequences, the models evoked by verbs denoting actions have to be, at the very least, sequential models capturing the likelihood of sequences generated by the model. While there is no principled choice here, we have decided to capture the embodied meaning of verbs via Hidden Markov Models that can be used also to anticipate the completion of actions by following the graphical structure of the model.

Our approach models the first encounter of a word as the start of the acquisition of a category and thus is in line with the theory of linguistic relativism in the sense that language triggers the induction of a category. The model is composed of three components, a component for language learning, a component for action learning and a component that orchestrates the learning process across both modalities. Both modality-specific components perform representation learning in a bottom-up incremental fashion, unifying and merging specific instances into more general representations that capture the essential characteristics of linguistic structure and action concepts that are 'generative' in the sense that they are able to produce different surface forms.

The component for language learning is based on an existing model (section 2.6) that represents linguistic constructions as paths in a network of constructions. A suitable component for learning action representations did not exist to the best of our knowledge. In order to implement a suitable model we first discussed technical requirements that result from the principles of *usage-based* learning (see section section 3.2), before we detailed the technical implementation of the component based on the model merging framework.

The component for action (see chapter 3) addresses the task of incrementally inducing sequential models of action that are learned in an item-based fashion where each encounter of a new training instance leads to direct adaptation of the learned representations. The model is based on HMMs that are learned in a model merging procedure where general category-specific models emerge from gradually merging more specific models that have a lower entropy (see section 3.3.3). A critical issue in approaches based on model merging is controlling the complexity of the resulting models such that the complexity of the model is aligned with the complexity of the concepts they represent.

The action learning approach is based on a bayesian criterion composed of a prior that assigns higher probability to smaller models in general and a likelihood that measures the plausibility of the model to accurately represent the constituting data. Prior and likelihood represent two opposing forces. The prior has its maximum at the smallest possible model which represents actions as a single probability distribution of observations. The likelihood has its maximum at a model that simply duplicates the constituting data in its graphical structure. It is the task of the learning procedure to control the model complexity during the learning process such that it keeps an equilibrium between those two extremes.

The bias towards smaller models can be interpreted as a case of Occam's Razor, that prefers simpler explanations over more complex ones unless they are required to explain the data. The preference for compact representations is also incorporated in human cognition as our cognitive model is fundamentally limited in the volume of our brains. In order to fit more and more complex knowledge into our cognitive model we have to represent that knowledge in an increasingly compact and efficient manner. The way bayesian inference assigns probability inherently leads to simpler hypothesis by a tendency often referred to as Bayesian occam's factors (MacKay et al. 2003), which refers to the observation that the more parameters an expression has, the smaller the probability mass assigned to each subterm, since the joint probability is by definition constrained to sum to one. Both, the preference for less complex models and the preference for a good fit of model predictions and training data fundamentally represent the same trade-off as the under- and overfitting phenomena in machine learning. To increase the degree of control over the learning process we introduced a parameter λ that allows to fine tune the trade-off between model size and likelihood (see section 3.3.3).

The generalization controller gathers signals from the modality-specific learning components that indicate the existence of similar structure within the respective modality which can be merged into more general representations. The merging operation is only performed if merging grounded representations in the other modality would also yield meaningful results.

The design and implementation of the model was structured along the lines of the following research questions:

- **How can categorical knowledge of action be represented under the prerequisite that those representations should be emergent and learned in an item-based manner?**

  In chapter 3 we answered this question by first identifying the key requirements that result from a learning process that is emergent and item-based in nature. In compliance with those requirements we implemented a model that represents categorical knowledge of action as category specific Hidden Markov models that are incrementally induced following a model merging approach. Model merging is a well established approach that realizes the emergence of categorical representations starting with very specific models, representing individual examples that are gradually merged into increasingly general models. We transferred this approach to the domain of action using Hidden Markov models and simplicity of the resulting models as the inductive bias that drives the incremental generalization process. This way merging of specific HMMs results in more general models that have a higher entropy compared to the very specific initial models. At the same time, the merging process ensures that the generalized models still assign substantial probability mass to the observed examples while minimizing model complexity.

We evaluated the developed model against a well established procedure to train Hidden Markov models and against a state of the art representative of the neural networks family of models. We found that models induced by the incremental learning procedure perform comparable to HMMs learned by expectation maximization or the LSTM based model. Furthermore we evaluated the HMM based models in terms of different quality dimensions that resulted from the analysis of the requirements on the model that result from the specific incremental learning scheme.

- **Can general and simple concepts like simplicity and likelihood be applied to guide the incremental model-merging process and how do the resulting models compare to models that are learned in batch mode?**

In an incremental learning setting it is often more challenging to control the optimal degree of generalization as opposed to settings where the complete data is available to the training procedure at once. Simplicity and likelihood as optimization targets in learning action models represent opposing forces, that, considered separately, would drive the optimization process to result in different extremes of target models. A critical issue in model merging is controlling the complexity of the resulting models so that the model's complexity is justified by the data they are based on.

Simplicity as a separate optimization target would lead to overly simplifying models that under-fit concepts in the underlying data. Optimizing towards the point of maximal likelihood on the other hand would result in Markov models that simply duplicate their constituting data, given that the complexity (size and family of distributions) of the model is not otherwise restricted. This is mainly caused by the initial specific models being maximum likelihood models that exactly reproduce their respective sequence of observations. A larger model composed of those maximum likelihood models would still be a model that maximises the likelihood given the underlying data but it could not generalize to observations it has not been exposed to in training.

We found that contrasting the likelihood with the simplicity (see section 3.3.3) of the resulting models as a joint criterium that drives the optimization process leads to well performing models that automatically control their representational complexity as a byproduct of striving towards a pareto optimal solution between simplicity and likelihood.

To answer this question we benchmarked the models that were incrementally learned driven by the joint criterion against the well established expectation maximization batch learning scheme. We found that the incrementally trained models performed comparable to the batch learned models (see subsection 3.4.1)

- **How can cross-modal learning of grounded action concepts and linguistic constructions be orchestrated in an emergentist learning process?**

  We developed a model that consists of a component for action learning, a component for language learning and a third component that orchestrates the cross-modal generalization process. The model learns incrementally by first inducing simple representations on first encounter of a concept within a modality that - during generalization - are gradually abstracted into increasingly complex representations. This questions refers to the challenge how grounding relations can be established between concepts that develop in independent modality-specific components. The answer to this question is fundamentally represented in the way the third component of the model, the cross-modal generalization controller operates.

  In order to establish grounding relations we exploit the circumstance that the input to the model consists of examples of action that participants of a study (see section 3.1) provided when challenged with an instruction in natural language. Although these examples are noisy and not always correct they provide coupled instances of action and language that refer to the same action concept in most cases. Upon the first occurrence of a certain utterance together with an action sequence, both models create a category most specific for that individual sequence of words and action. The generalization controller establishes a grounding relation between those categories that is preserved during the incremental generalization process. Both learning components expose similarity based unification cues in the sense that two concepts within their modality could be unified into more abstract representations. The generalization controller receives those unification cues and validates that the corresponding concepts in the other modality (connected by grounding relations) can also be unified, such that unification is only carried out on both ends of a grounding relation. This way the generalization controller preserves the original coupling of action and language that is present in the training examples.

The presented model for multi-modal learning required a learner to already have acquired basic perceptual and motor skills as a prerequisite. To overcome this limitation we suggest a model (as a proof of concept) that opens a possible path to how these skills could be learned as a byproduct of guided exploration driven by the desire to induce change in the environment which can be seen as a simplified version of curiosity in humans. We have proposed a reinforcement learning based model using deep Q-networks as the fundamental abstraction to learn to perceive and manipulate a toy-clock in a simulated environment.

This experiment is related to deep learning models that learn to play video games in that we use a deep Q-learning model similar to the one proposed by Mnih, Kavukcuoglu, Silver, Graves, et al. (2013) with additional double Q-learning (see Hasselt 2010; Hasselt et al. 2015) and a Boltzmann exploration strategy instead of the originally proposed ϵ-greedy policy. However, it also crucially differs from this line of research in that we are concerned with learning in environments that have less inherent structure than video games that where specifically designed to give frequent reward to keep human players motivated.

The design and implementation of the model was structured along the lines of the following research questions:

- **How can a model be designed in the framework of reinforcement learning that exemplifies a possible path to how basic perceptual and movement primitives can develop without complex assumptions to begin with?**

  The model is based on deep Q-learning (see section 2.7, a variant of the classical Q-learning approach that approximates the Q-function by a (deep) neural network and thus transfers the ideas of Q-learning to potentially very large problems with continuous state-space representations.

  Reinforcement learning frames the learning problem as a (partially observable) Markov decision process where an agent autonomously learns a strategy to manipulate its environment through a defined set of actions in a goal directed manner. The strategy is not learned according to direct examples of state-action mappings as in supervised learning but rather the agent receives rewards as feedback signals at certain points in time. In our model the agent can perceive its environment in terms of distances to other objects (section 5.2). Compared to the prerequisite of the first model that required the system to be capable to perceive distinguishable objects and their positions, this is a very simple requirement, that does not involve any complex classification. To act in the environment, the agent was equipped with the capacity to accelerate in discrete directions.

  We found that the agent learned how to move in a goal-directed manner towards the hands of the simulated toy clock and how to keep them moving through most of the experiment based on the described simple perceptual and movement primitives the agent was equipped with.

- **Is simulated curiosity or more specifically the desire to induce change in the environment sufficient to guide a learning system that is primarily based on random parallel exploration and exploitation of knowledge that was gained in the process?**

  The design of the model was based on the idea that the system should develop basic perceptual and movement skills by *self-action* rather than by *observed action* as in the previous model (Huttenlocher et al. 1983) . Learning through *self-action* is represented in the way the agent explores its environment by taking action and observing effects in terms of change in the environment. In order to explore the environment in a goal-directed manner the agent needs to develop corresponding perceptual capabilities and an internal representation of the environment together with the state it is currently in.

  We found that simulated curiosity as an intrinsic motivation (section 5.1) that is implemented as the desire to induce change in the environment is sufficient to guide the exploration and learning process to a point where the agent is able to maximize the induced change in the environment.

- **What are possible co-factors that influence the exploration or learning process?**

  We found that conceptual parameters such as the reward structure or the exploration strategy that mediates between exploration and exploitation of already acquired knowledge (subsection 5.2.1) have larger effects on the outcome than technical parameters such as regularization or the model structure and capacity.

Outlook

Our model so far has concentrated only on modelling the spatio-temporal essence of actions in which some trajector is moved relatively to some reference object. This is a very restricted subset of actions. To account for other verbs, the teleological structure of actions would need to be modeled. A richer modelling of actions will also require representing thematic roles such as patient and agent in a developmentally appropriate and grounded fashion. Our model has assumed that a system can recognize such relations from the input and model them symbolically. In terms of the abilities of a developing language learner, our system assumes that the following abilities are already acquired or inborn: the ability to segment and track objects, the ability to segment and identify words, the ability to recognize and conceptualize basic scenes as well as the ability to jointly attend to a tutor that demonstrates actions. All these are in itself complex abilities the emergence of which needs to be explained. With our second experiment we illustrated a possible path how those abilities could develop sub-symbolically as a byproduct of an autonomous agent solving a simple perception-action task. An interesting follow-up experiment would be to attempt to develop higher level goal-directed action concepts, such as setting the clock to a specific time, based on the fundamental object manipulation capabilities acquired through this approach.

While our proposed computational system for modelling the co-emergence of linguistic and action representations has been shown to be effective in our experiments, it is still a very ‘mechanistic’ system, assuming that a system can represent construction networks and HMMs explicitly, perform statistical inference and merging operations on these representations, etc. It is indeed an open question if learning systems can encode and manipulate such representations on a neural substrate. Consecutive future work could seek to find architectures that show a similar behaviour but do not postulate the existence of data structures such as proposed in the model but instead rely on cross-modal deep learning and representational learning to explain the co-emergence of representations across modalities. An interesting intermediate step in that direction could be to learn a mapping between sub-symbolic representations that develop in a neural network such as in the second experiment and symbolic representations that are used as input for our cross-modal learning system. The work of Gärdenfors about conceptual spaces (Gärdenfors 2004) constitutes a compromise between sub-symbolic and symbolic representations in that they offer symbolic representations as connected regions in a space of quality dimensions (color, shape, weight, ...) but also postulate that there is a sub-symbolic mapping from points in qualitative space to sensorimotor experiences. Conceptual spaces also have a metric in quality space that allows to derive similarity measures between individual examples, which is a prerequisite for the multi-modal learning system. However, it is not clear how the quality dimensions can be derived automatically. Furthermore, mappings from words to categories are likely not static. For example color perception and naming are subjective and can not easily be captured by static mappings.

Bibliography

Anna M. Borghi, A. Cangelosi (2014). “Action and Language Integration: From Humans to Cognitive Robots”. In: *Trends in Cognitive Science* 6, pp. 344–358 (cit. on p. 113).

Arulkumaran, Kai et al. (2017). “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34.6, pp. 26–38 (cit. on p. 124).

Barsalou, Lawrence W (2003). “Abstraction in perceptual symbol systems”. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 358.1435, pp. 1177–1187 (cit. on p. 112).

– (2008). “Grounded Cognition”. In: *Annu. Rev. Psychol.* 59, pp. 617–645 (cit. on p. 112).

Barsalou, Lawrence W. et al. (2003). “Grounding conceptual knowledge in modality-specific systems”. In: *Trends in Cognitive Sciences* 7.2, pp. 84–91 (cit. on p. 112).

Baum, Leonard E et al. (1970). “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”. In: *The annals of mathematical statistics* 41.1, pp. 164–171 (cit. on pp. 14, 20).

Behrens, H. (2017). In: *Linguistics* 47.2, pp. 383–411 (cit. on pp. 4, 90).

Bengio, Samy et al. (1996). “An EM algorithm for asynchronous inputoutput hidden Markov models”. In: *International Conference On Neural Information Processing*. Vol. 78. Citeseer, pp. 328–334 (cit. on pp. 14, 20).

Bengio, Yoshua, Renato De Mori, et al. (1992). “Global optimization of a neural network-hidden Markov model hybrid”. In: *IEEE transactions on Neural Networks* 3.2, pp. 252–259 (cit. on p. 16).

Bengio, Yoshua, Patrice Simard, et al. (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2, pp. 157–166 (cit. on p. 32).

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag (cit. on p. 85).

Bowerman, M. et al. (2001). *Language Acquisition and Conceptual Development*. 3. Cambridge University Press, pp. 128–135 (cit. on pp. 49, 89).

Brunet, Greg et al. (2006). “A manifesto for model merging”. In: *Proceedings of the 2006 international workshop on Global integrated model management - GaMMa ’06*, pp. 5–12. DOI: 10.1145/1138304.1138307 (cit. on p. 23).

Bruss, Thomas et al. (2010). “On the perception of time”. In: *Gerontology* 56.4, pp. 361–370 (cit. on p. 63).

Bryant, John Edward (2008). *Best-fit constructional analysis.* University of California, Berkeley (cit. on p. 54).

Cangelosi, A. et al. (2007). “Integrating Language and Cognition: A Cognitive Robotics Approach”. In: *IEEE Computational Intelligence Magazine* 2.3, pp. 65–70 (cit. on p. 113).

Cangelosi et al. (2015). *Developmental robotics: From babies to robots.* MIT Press (cit. on pp. 87, 121).

Carey, S. (1978). “The child as word learner”. In: *Linguistic Theory and Psychological Reality.* MIT Press (cit. on p. 135).

Carey, Susan et al. (1978). “Acquiring a single new word.” In: (cit. on pp. 42, 56).

Chang, Nancy et al. (2004). “Structured Connectionist Models of Language, Cognition and Action”. In: *Ninth Neural Computation and Psychology Workshop* (cit. on p. 112).

Clark, Andy (2013). “Whatever next? Predictive brains, situated agents, and the future of cognitive science”. In: *Behavioral and brain sciences* 36.3, pp. 181–204 (cit. on p. 58).

Clark, Andy and Josefa Toribio (2012). “Magic Words: How Language Augments Human Computation”. In: *Language and Meaning in Cognitive Science.* Routledge, pp. 33–51 (cit. on p. 114).

Clevert, Djork-Arné et al. (2015). “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *ICLR2016* 1997, pp. 1–13. arXiv: `1511.07289` (cit. on p. 127).

Cormen, Thomas H et al. (1990). “Introduction to algorithms”. In: *Cambridge MA* (cit. on p. 17).

Danks, David (2014). *Unifying the mind: Cognitive representations as graphical models.* Mit Press (cit. on pp. 4, 8, 14, 55).

Dominey, Peter Ford et al. (2005). “Learning to talk about events from narrated video in a construction grammar framework”. In: *Artificial Intelligence* 167.1-2, pp. 31–61 (cit. on p. 113).

Droniou, Alain et al. (2014). “Learning a Repertoire of Actions with Deep Neural Networks”. In: *Joint International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, 6–p (cit. on p. 113).

Duchi, John et al. (2011). “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12, pp. 2121–2159. DOI: `10.1109/CDC.2012.6426698`. arXiv: `arXiv:1103.4296v1` (cit. on p. 128).

Fazly, Afsaneh et al. (2010). “A Probabilistic Computational Model of Cross-situational Word Learning”. In: *Cognitive Science* 34.6, pp. 1017–1063 (cit. on p. 113).

Feldman, Jerome A. (2006). *From Molecule to Metaphor: A Neural Theory of Language.* MIT Press (cit. on p. 112).

Feldman, Jerome et al. (2009). “Embodied construction grammar”. In: *The Oxford handbook of linguistic analysis* (cit. on p. 54).

Fillmore, Charles J. (1976). “Frame Semantics and the nature of language”. In: *Annals of the New York Academy of Sciences* 280.1, pp. 20–32 (cit. on p. 116).

Gärdenfors, Peter (2004). *Conceptual spaces: The geometry of thought.* MIT press (cit. on p. 142).

Gaspers, Judith and Philipp Cimiano (2014). “A computational model for the item-based induction of construction networks”. In: *Cognitive Science* 38.3, pp. 439–488 (cit. on pp. 2, 5, 38, 41, 91, 101, 105).

Gaspers, Judith, Anouschka Foltz, et al. (2014). “Towards the emergence of verb-general constructions and early representations for verb entries: Insights from a computational model”. In: *CogSci 2014, Quebec City, Canada, July 23-26, 2014* (cit. on p. 117).

Gentner, Dedre et al. (2001). “Individuation, Relativity, and early word learning”. In: *Language Acquisition and Conceptual Development.* Ed. by Melissa Bowerman et al. New York: ambridge University Press (cit. on p. 114).

Gepperth, Alexander et al. (2016). “Incremental learning algorithms and applications”. In: *European Symposium on Artificial Neural Networks ({ESANN})* April, pp. 357–368 (cit. on p. 56).

Gergely, G. et al. (2003). “Teleological reasoning in infancy: the naive theory of rational action”. In: *Trends in Cognitive Sciences* 7.7, pp. 287–292 (cit. on p. 117).

Gers, Felix A et al. (2000). “Learning to forget: Continual prediction with LSTM”. In: *Neural computation* 12.10, pp. 2451–2471 (cit. on p. 33).

Gibbson, James J. (1977). “The Theory of Affordances”. In: *Perceiving, Acting, and Knowing.* Ed. by Robert Shaw et al. (cit. on p. 118).

Gleitman, Lila (1990). “The Structural Sources of Verb Meanings”. In: *Language Acquisition* 1.1, pp. 3–55 (cit. on p. 49).

Glenberg, A. M. et al. (2007). “Grounding language in action”. In: *Psychonomic Bulletin and Review* 9.3, pp. 558–565 (cit. on p. 113).

Goodfellow, Ian J. et al. (2014). “Generative Adversarial Networks”. In: pp. 1–9. DOI: `10.1001/jamainternmed.2016.8245`. arXiv: `1406.2661` (cit. on p. 55).

Greff, Klaus et al. (2015). “LSTM: A Search Space Odyssey”. In: *CoRR* abs/1503.04069 (cit. on pp. 33, 34).

Hasselt, Hado van (2010). “Double Q-learning”. In: *Advances in Neural Information Processing Systems*, pp. 2613–2621 (cit. on pp. 47, 127, 132, 140).

Hasselt, Hado van et al. (2015). “Deep Reinforcement Learning with Double Q-learning”. In: *arXiv:1509.06461 [cs]* 2, pp. 1–5. DOI: `10.1016/j.artint.2015.09.002`. arXiv: `1509.06461` (cit. on pp. 125, 127, 132, 140).

Hebb, Donald O (1949). “The organization of behavior. A neuropsychological theory”. In: (cit. on p. 41).

Hinaut, Xavier et al. (2014). “Exploring the Acquisition and Production of Grammatical Constructions Through Human-Robot Interaction with Echo State Networks”. In: *Frontiers in Neurorobotics* 8.16, pp. 1–17. DOI: `10.3389/fnbot.2014.00016` (cit. on p. 113).

Hochreiter, Sepp (1991). “Untersuchungen zu dynamischen neuronalen Netzen”. In: *Diploma, Technische Universität München* 91.1 (cit. on p. 32).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 33).

Hofstadter, Douglas et al. (2013). *Surfaces and essences: Analogy as the fuel and fire of thinking.* Basic Books (cit. on p. 117).

Hsiao, Kai-yuh et al. (2008). “Object schemas for grounding language in a responsive robot”. In: *Connect. Sci.* 20.4, pp. 253–276 (cit. on p. 113).

Huttenlocher, Janellen et al. (1983). “Emergence of action categories in the child: Evidence from verb meanings”. In: *Psychological Review* 90.1, pp. 72–93. DOI: `10.1037/0033-295X.90.1.72` (cit. on pp. 3, 50, 120, 132, 141).

Ioffe, Sergey et al. (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Arxiv*, pp. 1–11. DOI: `10.1007/s13398-014-0173-7.2`. arXiv: `1502.03167`. URL: `http://arxiv.org/abs/1502.03167` (cit. on p. 127).

Jackman, Simon (2009). *Bayesian analysis for the social sciences.* Vol. 846. John Wiley & Sons (cit. on p. 74).

Joong, Yo et al. (2017). “Probabilistic Interpretations of Recurrent Neural Networks”. In: (cit. on p. 55).

Juang et al. (1985). “A probabilistic Distance Measure for Hidden Markov Models”. In: *AT&T technical journal* 64.2, pp. 391–408 (cit. on pp. 97, 102).

– (1990). “The segmental K-means algorithm for estimating parameters of hidden Markov models”. In: *IEEE Transactions on acoustics, speech, and signal Processing* 38.9, pp. 1639–1641 (cit. on p. 21).

Keogh, Eamonn J. et al. (2000). “Scaling up dynamic time warping for datamining applications”. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’00*, pp. 285–289. DOI: `10.1145/347090.347153` (cit. on p. 12).

Kingma, Diederik P. et al. (2015). “Adam: a Method for Stochastic Optimization”. In: *International Conference on Learning Representations 2015*, pp. 1–15. arXiv: `1412.6980` (cit. on p. 128).

Koller, Daphne et al. (2009). *Probabilistic graphical models: principles and techniques.* MIT press (cit. on p. 16).

Kruse, Eckhard et al. (1997). “Acquisition of statistical motion patterns in dynamic environments and their application to mobile robot motion planning”. In: *IROS’97.* Vol. 2. IEEE, pp. 712–717 (cit. on p. 113).

Lillicrap, Timothy P. et al. (2015). “Continuous control with deep reinforcement learning”. In: *CoRR* abs/1509.02971 (cit. on p. 125).

Lupyan, Gary et al. (2007). “Language is not just for talking: Redundant labels facilitate learning of novel categories”. In: *Psychological science* 18.12, pp. 1077–1083 (cit. on p. 114).

MacKay, David JC et al. (2003). *Information theory, inference and learning algorithms.* Cambridge university press (cit. on p. 137).

Manning, Christopher D et al. (1999). *Foundations of statistical natural language processing.* MIT press (cit. on pp. 16, 17, 21).

McMurray, Bob et al. (2012). “Word Learning Emerges From the Interaction of Online Referent Selection and Slow Associative Learning”. In: *Psychological Review* 119.4, pp. 831–877 (cit. on pp. 89, 116, 135).

Mitchell, T M (1980). “The Need for Biases in Learning Generalizations”. In: *Readings in Machine Learning* CBM-TR-117, pp. 184–191 (cit. on p. 26).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, et al. (2013). “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (cit. on pp. 125, 126, 128, 132, 140).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei a Rusu, et al. (2015). “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540, pp. 529–533. DOI: `10.1038/nature14236`. arXiv: `1312.5602`. URL: `http://dx.doi.org/10.1038/nature14236` (cit. on pp. 46, 47, 127).

Omohundro, Stephen M. (1993). “Best-first model merging for dynamic learning and recognition”. In: *Neural Information Processing Systems (NIPS)* 4, pp. 958–965 (cit. on pp. 23, 25, 26, 29, 30, 69).

Pearl, Lisa et al. (2011). “Online Learning Mechanisms for Bayesian Models of Word Segmentation”. In: *Research on Language and Computation* 2010, pp. 1–26. DOI: `10.1007/s11168-011-9074-5` (cit. on p. 92).

Pickering, M. et al. (2013). “An integrated theory of language production and comprehension”. In: *Behavioral and Brain Sciences* 36.4, pp. 329–247 (cit. on pp. 6, 111).

Pine, Julian M et al. (1997). “Slot and frame patterns and the development of the determiner category”. In: *Applied psycholinguistics* 18.2, pp. 123–138 (cit. on p. 4).

Pulvermüller, Friedemann (2013). “How neurons make meaning: brain mechanisms for embodied and abstract-symbolic semantics”. In: *Trends in cognitive sciences* 17.9, pp. 458–470 (cit. on p. 112).

Rabiner, Lawrence R et al. (1993). *Fundamentals of speech recognition.* Vol. 14. PTR Prentice Hall Englewood Cliffs (cit. on p. 74).

Rochat, Philippe (1998). “Self-perception and action in infancy”. In: *Experimental Brain Research* 123.1-2, pp. 102–109. DOI: `10.1007/s002210050550` (cit. on p. 120).

Rohlfing, Katharina J. et al. (2016). “An Alternative to Mapping a Word onto a Concept in Language Acquisition: Pragmatic Frames”. In: *Frontiers in Psychology* 7.470 (cit. on p. 115).

Rojas, Raul (2013). *Theorie der neuronalen Netze: eine systematische Einführung.* Springer-Verlag (cit. on p. 41).

Roy, Deb et al. (2002). “Learning words from sights and sounds: a computational model”. In: *Cognitive Science* 26.1, pp. 113–146 (cit. on p. 113).

Schatten, Rolf (2003). “Systemic architecture for audio signal processing”. In: *European Conference on Artificial Life.* Springer, pp. 491–498 (cit. on p. 41).

Schmidhuber, Jürgen (1991). “A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers”. In: *Meyer, J.A. and Wilson, S.W. (eds) : From Animals to animats* 1, pp. 222–227 (cit. on p. 133).

Shannon, Claude Elwood (1949). *The Mathematical Theory of Communication.* Bell System Tech. Journal (cit. on p. 28).

Shepard, R. (1987). "Toward a universal law of generalization for psychological science". In: *Science* 237.4820, pp. 1317–1323. DOI: `10.1126/science.3629243` (cit. on pp. 57, 83).

Singh, S. et al. (2004). "Intrinsically motivated reinforcement learning". In: *18th Annual Conference on Neural Information Processing Systems (NIPS)* 17.2, pp. 1281–1288. DOI: `10.1109/TAMD.2010.2051031` (cit. on p. 133).

Siskind, Jeffrey Mark (Oct. 1996). "A computational study of cross-situational techniques for learning word-to-meaning mappings". In: *Cognition* 61.1-2, pp. 1–38 (cit. on pp. 89, 113).

Smith, Linda B (2000). "How to learn words: An associative crane". In: *Breaking the word learning barrier* (cit. on p. 89).

Smith, Linda et al. (2008). "Infants rapidly learn word-referent mappings via cross-situational statistics". In: *Cognition* 106.3, pp. 1558–1568 (cit. on pp. 89, 116).

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. DOI: `10.1214/12-AOS1000`. arXiv: `1102.4807` (cit. on p. 132).

Stamp, Mark (2004). "A revealing introduction to hidden Markov models". In: *Department of Computer Science San Jose State ...* Pp. 1–20. DOI: `10.1.1.136.137` (cit. on pp. 16, 17).

Stolcke, Andreas et al. (1993). "Hidden Markov model induction by Bayesian model merging". In: *Advances in neural information processing systems*, p. 11 (cit. on pp. 14, 23, 25, 26, 57, 66, 69).

Sugiura, Komei et al. (2011). "Learning, Generation and Recognition of Motions by Reference-Point-Dependent Probabilistic Models." In: *Advanced Robotics* 25.6-7, pp. 825–848 (cit. on p. 113).

Sutton, Richard S. (1988). "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1, pp. 9–44. DOI: `10.1007/BF00115009`. URL: `http://dx.doi.org/10.1007/BF00115009` (cit. on pp. 46, 125).

Sutton, Richard S et al. (2017). *Reinforcement learning: An introduction.* Vol. 2. MIT press Cambridge (cit. on pp. 44, 130).

Taylor, Lawrence et al. (2009). "Action in cognition: the case of language". In: *Language and Cognition* 1.1, pp. 45–58 (cit. on p. 116).

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288 (cit. on p. 60).

Tieleman, Tijmen et al. (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural Networks for Machine Learning* 4, p. 2 (cit. on pp. 79, 128).

Tomasello, Michael (2000). "The item-based nature of children's early syntactic development". In: *Trends in Cognitive Sciences* 4.4 (cit. on p. 4).

– (2001). "Could We Please Lose the Mapping Metaphor, Please?" In: *Behavioral and Brain Sciences* 24.6, pp. 1119–1120 (cit. on p. 115).

Tomasello, Michael et al. (1993). "Cultural learning". In: *Behavioral and Brain Sciences* 16.3, pp. 495–552 (cit. on p. 2).

Van Trijp, Remi et al. (2012). "Fluid Construction Grammar: The New Kid on the Block". In: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics.* EACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 63–68 (cit. on p. 113).

Watkins, Christopher J. C. H. et al. (1992). "Q-learning". In: *Machine Learning* 8.3-4, pp. 279–292. DOI: 10.1007/BF00992698 (cit. on pp. 44, 46, 125).

Waxman, S.R. et al. (1995). "Words as invitations to form categories: evidence from 12-to 13-month-old infants". In: *Cognitive Psychology* 7.470 (cit. on p. 114).

Weghe, Nico et al. (2005). "A Qualitative Trajectory Calculus and the Composition of Its Relations". In: *GeoSpatial Semantics SE - 5* 3799.Dc, pp. 60–76. DOI: 10.1007/11586180_5 (cit. on p. 36).

Welch, Lloyd R (2003). "Hidden Markov models and the Baum-Welch algorithm". In: *IEEE Information Theory Society Newsletter* 53.4, pp. 10–13 (cit. on p. 20).

Willems, R. M. et al. (2007). "Neural evidence for the interplay between language, gesture and action: A review". In: *Brain and Language* 101, pp. 278–289 (cit. on p. 113).

Wolff, Philip et al. (2010). "Linguistic Relativity". In: *WIREs Cogn. Sci.* (Cit. on p. 113).

– (2011). "Linguistic relativity". In: *Wiley Interdisciplinary Reviews: Cognitive Science* 2.3, pp. 253–265. DOI: 10.1002/wcs.104 (cit. on p. 89).

Wolff, Phillip et al. (2011). "Linguistic relativity". In: *Wiley Interdisciplinary Reviews: Cognitive Science* 2.3, pp. 253–265 (cit. on p. 49).

Xing, Zhengzheng et al. (2010). "A brief survey on sequence classification". In: *ACM SIGKDD Explorations Newsletter* 12.1, p. 40. DOI: 10.1145/1882471.1882478 (cit. on p. 11).

Xu, Fei (2002). "The role of language in acquiring object kind concepts in infancy". In: *Cognition* 85, pp. 223–250 (cit. on p. 114).