State Discovery for Autonomous Learning

by

Yuri A. Ivanov

M.S., Computer Science and Electrical Engineering
State Academy of Air and Space Instrumentation, St. Petersburg, Russia
February 1992

M.S., Media Arts and Sciences
Massachusetts Institute of Technology,
February 1998

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2002

Signature of Author ________________________________

Yuri A. Ivanov
Program in Media Arts and Sciences
October 17, 2001

Certified by ________________________________

Bruce M. Blumberg
Asahi Broadcasting Corporation Career Development
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Certified by ________________________________

Alex P. Pentland
Toshiba Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by ________________________________

Andrew B. Lippman
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

State Discovery for Autonomous Learning

by
Yuri A. Ivanov

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on October 17, 2001
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Abstract

This thesis is devoted to the study of algorithms for early perceptual learning for an autonomous agent in the presence of feedback. In the framework of associative perceptual learning with indirect supervision, three learning techniques are examined in detail:

- short-term on-line memory-based model learning;
- long-term on-line distribution-based statistical estimation;
- mixed on- and off-line continuous learning of gesture models.

The three methods proceed within essentially the same framework, consisting of a perceptual sub-system and a sub-system that implements the associative mapping from perceptual categories to actions.

The thesis contributes in several areas – it formulates the framework for solving incremental associative learning tasks; introduces the idea of incremental classification with utility, margin and boundary compression rules; develops a technique of sequence classification with Support Vector Machines; introduces an idea of weak transduction and offers an EM-based algorithm for solving it; proposes a mixed on- and off-line algorithm for learning continuous gesture with reward-based decomposition of the state space.

The proposed framework facilitates the development of agents and human-computer interfaces that can be trained by a naïve user. The work presented in this dissertation focuses on making these incremental learning algorithms practical.

Thesis Supervisor: Bruce M. Blumberg
Title: Asahi Broadcasting Corporation Career Development Associate Professor of Media Arts and Sciences

Thesis Supervisor: Alex P. Pentland
Title: Toshiba Professor of Media Arts and Sciences

Doctoral Dissertation Committee

Thesis Supervisor: ______________________________

Bruce M. Blumberg
Asahi Broadcasting Corporation Career Development
Associate Professor of Media Arts and Sciences,
Massachusetts Institute of Technology

Thesis Supervisor: ______________________________

Alex P. Pentland
Toshiba Professor of Media Arts and Sciences,
Massachusetts Institute of Technology

Reader: ______________________________

Aaron F. Bobick
Associate Professor, College of Computing,
Georgia Institute of Technology

Reader: ______________________________

Leslie Pack Kaelbling
Professor of Computer Science and Engineering,
Department of Electrical Engineering and Computer Science,
Massachusetts Institute of Technology

To Sarah and Sasha

Acknowledgments

First and foremost, I would like to thank my committee who spent so much more time thinking about my thesis than they had to.

Bruce Blumberg has been a great advisor and collaborator. Bruce always had a difficult question in his pocket whenever I would become too certain about how things should work. I am really grateful to him for building such a great group and giving me the freedom to come to my own conclusions in my own ways.

Thanks to Alex Pentland, my co-advisor, for maintaining such a high degree of academic excellence in the Vision and Modeling group that kept me on my toes all these years.

My deepest gratitude goes to Aaron Bobick, my first advisor at the lab, for opening the doors for me into the world of academia. On both personal and professional level Aaron's contribution to my life can only be measured in megatons.

Many many thanks to Leslie Pack Kaelbling who was the only person on my committee who has not been my advisor at one point or another, but I really wished she could have been one too. Well, she really was, just not on paper, going far beyond the call of duty of a committee member.

I think the best thing that I got after 5 years at MIT are my true friends - Drew Wilson who never discarded any of my ideas, however silly they were, Chris "Papa" Wren with his generous support and fearless thinking, and Bill Butera with his limitless enthusiasm and constant reminders for me to be true to myself.

I learned so much from all the people at the Media Lab - I would like to thank them all. In no particular order:

Characters: Mike Johnson, Marc Downie, Rob Burke, Damian Isla, Bill Tomlinson, Aileen Kawabe, Scott Eaton, Ben Resner, , Matt Berlin, Lily Shirvanee, Matt Grimes, Chris Kline, Michal Hlavac, Ken Russel.

VisModlians: Tony Jebara, Ali Azayerbaejani, Baback Moghaddam, Irfan Essa, Kris Popat, Teresa Marrin, Lee Campbell, Martin Szummer, Dave Becker, Nuria Oliver, Ali Rahimi, Steve Schwartz, Deb Roy, Stephen Intille, Jim Davis, Rich DeVaul, Thad Starner, Jen Healey, Claudio Pinhanez, Tom Minka, Raul Fernandez, Carson Reynolds, Ashish Kapoor, Jocelyn Shierer, Karen Navarro, Sumit Basu, Tanzeem Choudhury, Brian Clarkson, Flavia Sparacino, Roz Picard.

I'd like to thank the people who I have been aspiring to one day call colleagues and who taught me so much, sometimes without even knowing it: Oliver Faugeras, Federico Girosi, Eric Grimson, Berthold Horn. Special thanks to Tomaso Poggio for his insightful encouragement and guidance.
Thank you everyone, it has been a blast!

On a more personal note, thanks to Marjorie for making me so welcome in her life, to Steve for showing me the grace and honor of a true Marine, to M'Liz for showing me the silly side of life, to Larry and Marie and the rest of the Jersey gang for being so cool. Thanks to Mom for her gift of poetry and music, her love and support and to Dad for showing me how gentle and caring a true father can be. Thanks to Sasha for giving me the right perspective on life and for being such a good kid and for coming to give me a hug in the middle of the defense.

And, finally, my love and infinite amount of gratitude go to Sarah, my wife and best friend. You know what? I love you more than ever!

Contents

1 Introduction **21**

1.1 Learning perception 21

1.2 Scope of the document 24

1.3 Document structure 25

1.4 Related Work 26

1.4.1 Models of learning in biological organisms 26

1.4.2 Pattern recognition 27

1.4.3 Speech and Gesture Recognition 28

1.4.4 Reinforcement learning 29

2 Framing the Problem **31**

2.1 Overview 31

2.2 Earlier Work 31

2.2.1 Applications 32

2.2.2 Lessons 34

2.3 Biological Inspiration 37

2.3.1 What Do We Call Perception? 37

2.3.2 Perceptual Learning 38

2.3.3 Semantic Grounding of Perception 40

2.4 Technological Motivations 42

2.5 Problem Setting 45

2.6 Notation 49

3 Learning Memory Based Models **51**
3.1 The Goal 51
3.2 Overview 51
3.2.1 Two Approaches to Solving the Problem 52
3.2.2 What Is Here 53
3.3 Sampled Classification Rules 53
3.3.1 Incremental Learning Algorithm 54
3.3.2 Classification with Compression Sets 59
3.4 Experimental Results 69
3.4.1 Trial by Eire 69
3.4.2 The Iris Set 71
3.4.3 The Wine Set 72
3.4.4 The Vowel Set 73
3.4.5 Discussion 74

4 Support Vector Classification of Sequences **77**
4.1 Dimensionality Normalization 79
4.1.1 Time alignment 79
4.1.2 Batch Time-Normalization 80
4.2 Support Sequence Machine 81

5 EM for Weak Transduction **85**
5.1 The Goal 85
5.2 Overview 86
5.3 Introduction 87
5.3.1 Long Term Perceptual Learning 87
5.3.2 Related Work 89
5.4 Estimation of the Associative Policy 90
5.4.1 Multi-State Bandit Problem 90
5.4.2 Solutions with Known State 91
5.4.3 Solutions with Hidden State 92

5.5 Clustering Under Reward . . . 93
5.5.1 Weak Transduction . . . 93
5.5.2 Reward-driven variational bound . . . 95
5.6 Reward-Driven Expectation Maximization . . . 100
5.7 Experiments . . . 101
5.7.1 EM for state estimation . . . 101
5.7.2 Multi-State Bandit with Hidden State . . . 104
5.8 Discussion . . . 110

6 Learning Markov Models 113
6.1 The Goal . . . 113
6.2 Overview . . . 114
6.3 Introduction . . . 116
6.3.1 Tricks of Animal Trainers . . . 116
6.3.2 On-line Gesture Learning . . . 117
6.3.3 Graphical Models . . . 117
6.3.4 Previous Work . . . 120
6.4 Multi-Model Gesture Representation . . . 121
6.4.1 Probabilistic Models of Gesture . . . 122
6.4.2 State Space Model . . . 124
6.4.3 Inference and Testing . . . 126
6.4.4 Multi-Model Viterbi Approximation . . . 128
6.4.5 Reward distribution . . . 132
6.4.6 State Sharing . . . 132
6.5 Model Extraction . . . 134
6.6 Algorithm Summary . . . 136
6.7 Experiments . . . 137
6.7.1 Mouse Gesture . . . 137
6.7.2 Vision System . . . 140
6.8 Discussion . . . 142

7 Conclusions **143**

7.1 Contributions 144

7.2 Further Work 148

A Nearest Neighbor Performance Bounds **153**

A.1 Asymptotic Bound 153

A.2 Finite Sample Bound 155

A.3 VC Bound on a Condensed Classifier 155

B Divergence for Gaussian Densities **159**

List of Figures

1-1 Duncan in his habitat. 23

2-1 Recognition of structured sequences 32

2-2 Recognition of conducting gesture 33

2-3 Semi-hidden representation in surveillance task 33

2-4 Parking lot monitoring system 35

2-5 Consistent application of rewards and punishments reinforces the association between a behavior and its context. 40

2-6 Duncan and his master 46

2-7 Models of the environment for associative search tasks 47

3-1 Problems of greedy action selection 54

3-2 Sampled labelling procedure 58

3-3 Margin and Boundary Compression Sets 62

3-4 The sheep 69

3-6 Training Duncan 70

3-5 Sheep|dog 70

3-7 Results of running the algorithms on the Iris dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far. 71

3-8 Results of running the algorithms on the UCI Wine dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far. 72

3-9 Results of running algorithms on the Japanese Vowel dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far. 73

4-1 Utterances for classification 78

5-1 N-armed bandit model 90
5-2 Multi-state bandit model 91
5-3 Strong and weak transduction 94
5-4 EM bound 96
5-5 Augmented reward bound 99
5-6 REM algorithm 101
5-7 Comparison of EM and REM 103
5-8 Effect of β on cluster assignment 103
5-9 Likelihood maximization with binary bandit 106
5-10 Likelihood maximization with full bandit 107
5-11 Source for the reward maximization task 108
5-12 Reward maximization with full bandit 109
5-13 REM and EM after a single run 109

6-1 Macro- and micro-states 114
6-2 Learning gesture models with reward 115
6-3 Canonical graphs 118
6-4 Independence structure of the full model 119
6-5 Reward-driven decomposition 120
6-6 Probabilistic Models of Gesture 122
6-7 Cut-set conditioning 124
6-8 Cut-set tree search 124
6-9 Problem of sequence length 127
6-10 Viterbi parse 128
6-11 State sharing model 133

6-12 Sampling scheme 135
6-13 Learning mouse gesture 138
6-14 State space factorization 139
6-15 Mouse gesture - extracted models 139
6-16 Learning hand gesture 141
6-17 Full and factorized transition matrices 141
6-18 Hand gesture - extracted models 142

List of Tables

3.1 Summary of compression rules and sampling strategy 67
3.2 Results on the Iris Set 71
3.3 Results on the Wine Set 72
3.4 Results on the Japanese Vowel Set 73

4.1 Performance of SVM with the DynA kernel on batch classification problem 83

Chapter 1

Introduction

1.1 Learning perception

This thesis is devoted to the study of algorithms for early perceptual learning for an autonomous agent in the presence of feedback.

Perceptual learning is an important aspect in building a complete autonomous system. Imagine an agent that needs to function well in an unfamiliar environment. The agent can observe the environment with a set of sensors and affect it with a set of actuators that it controls. Learning how to control the actuator for any given set of sensor readings can be a difficult problem and is the primary focus of many reinforcement learning algorithms. A problem that is not as frequently addressed is how to learn the perceptual organization of the sensory field such that it delivers the highest utility to the agent.

During my stay at MIT I have been fortunate to belong to two distinct groups: one primarily dealing with machine perception – sight and sound; and the other, with building artificial creatures capable of having a mind of their own. Naturally, my interests developed in such a way as to span both of these areas. As a consequence, in this work I am attempting to join the two interests in a mutually beneficial fashion – this document is devoted to a study of algorithms that use perceptual information to facilitate autonomy, while using the autonomy and the information available to the agent as a result of its direct manipulation of its environment to refine its perception.

The thesis is inspired by the existing knowledge about early human and animal learning. One key topic that it examines in detail is the role of experiences in forming perception. It is empirically evident that animals and humans are very good at adapting their perception in accordance with their experiences. As will be discussed later, certain aspects of speech acquisition, color perception and spatial object recognition provide good examples of such processes taking place. The work presented in this document takes a look at several algorithms for learning perceptual organization under indirect supervision, attempting to make them practical.

Within the framework of this thesis it is useful to think about types of agents – robots or animated creatures – that a teacher, or a trainer, can train to respond to certain stimuli. The presence of the teacher and the autonomy of the agent afford certain advantages and impose limitations on approaches one may consider taking while building such agents. The presence of the teacher makes it easier to speed up perceptual learning, directing the agent when necessary, while autonomy makes it impossible to use training examples directly, but only via some evaluative techniques. Indeed, in the task of the perceptual learning, the direct supervision would need to indicate to the agent's perceptual system the exact category to which the observation should be attributed. However, in the autonomous system such information is only manifested in the agent's behavior and may be unavailable, or sometimes even detrimental to the learning process (see section 2.4). In other words, the perceptual organization is subject to learning in the agent but remains hidden from the teacher.

Up until recently the emphasis in the agent's adaptation has been that the agent passively observes its environment, or, as it is often the case in reinforcement learning, actively samples its environment to find a better policy of its action selection. In my work I look at algorithms that allow the agent to be "taught". The agent possesses a set of primitive skills and a set of primitive sensors. The goal of the agent's adaptation is to acquire new skills, based on the demonstrated examples, to learn an organization of its perceptual system from the context in which the skills are typically applied, and develop the mapping that connects the learned perception with the acquired actions. I derive my inspiration from animal training, presenting algorithms and

Figure 1-1: Duncan in his habitat.

agent architectures, which allow for making the first step towards training agents in much the same way as animals are trained.

To give a concrete example, imagine that we would like to train a personal agent to respond to a set of voice commands. It is possible to build a system that can quickly learn a small set of such commands. In this thesis I show an example of an agent that does just that. Duncan (fig. 1-1[1]) is an animated synthetic dog, that can be taught to sit and roll over and perform a variety of other tasks in response to a spoken command in a matter of minutes[2], regardless of the language in which the commands are given. The algorithm used in Duncan's perceptual system provides an illustration of the general range of problems that I find especially interesting, as the influence of the agent's experiences on the formation of its perception is particularly explicit.

While the whole of Duncan's brain is a complex system, which allows it to act autonomously and includes a behavior system, a motor system, a navigation system etc., the approach to the design of its auditory perception discussed here allows us to view it in isolation. In this respect, the algorithm is simple. Viewed separately, Duncan's auditory perception is based on the following procedure: the input samples that result in a treat are stored, while the ones that do not are thrown away. After a

[1] Sketches and artwork courtesy of Scott Eaton.

[2] The auditory perceptual system of Duncan was used in several installations – "Isle of Man's best friend", "sheep|dog – Trial by Eire" and "Clicker by Eire". A scene from "sheep|dog" is shown in figure 1-1.

while, Duncan collects a large repertoire of distinct patterns that have caused delivery of a reward in context of different actions. This essentially results in memorizing all past observations, which is not extremely satisfying. A number of questions arise: How much does one need to memorize? How should one deal with large amounts of data after some amount of time passes? How does one learn efficiently? How does one learn only what is important? These are the questions that this thesis tries to answer while offering some practical solutions.

1.2 Scope of the document

The goal of this document is to present algorithms for on-line learning of *state space* for an autonomous learning agent. By state, I mean some sort of a cluster in the input space, or in the space of observations, of the agent, which might have a temporal component. The notion of a state here is perhaps a bit different from that frequently used in reinforcement learning literature. The internal representation of the state remains largely statistical, while it is used in a context of a simple reinforcement learning task.

The extent to which I look into problems of reinforcement learning is extremely limited and I am not claiming to cover any new territory in reinforcement learning per se. I will be using and modifying some fairly well explored algorithms, making them suitable for the framework of statistical learning. One distinctive feature of the approach that I take with respect to perceptual learning is that I try not to rely on a known but uncertain set of states to be used for action selection, as is frequently done. Instead, I focus on the advantage for the agent of having its perceptual space organized in a particular way. This entails re-estimation of the state space simultaneously with learning to react to new observations, which can be called a *state space discovery under reward*.

The thesis is devoted to the exploration of learning perceptual categories under reward. In this context, the thesis is exploring the algorithms that allow the agent to learn *quickly*, while finding perhaps not optimal but adequate representations, that can further be refined when more data become available.

The most important message of this document is that while solving practical problems of giving autonomous agents situated and grounded intelligence one needs to consider the learning problem in its entirety. The setting in which this document proceeds does not fit nicely into any of the established taxonomies of computer learning. In this thesis I introduce the new notion of *weak transduction* that allows us to formulate problems of impoverished supervision as described later in the text.

1.3 Document structure

The document is organized around three main technical chapters, chapters 3, 5 and 6. In each of them I focus on a specific subproblem of the perceptual learning under reward.

Chapter 3 is devoted to very early stages of learning, where the amount of data that has been observed so far makes it nearly impossible to compute the necessary statistics, that are typically required for distribution-based techniques. This condition calls for memory-based learning methods to be used in their stead. The chapter formulates an incremental learning algorithm for memory-based representations and explores several approaches to reducing its computational and memory requirements.

Chapter 4 serves as an addendum to the previous chapter, demonstrating one of the techniques of sequence classification used in chapter 3 in more detail. It deserves to be placed in a separate chapter because, while being used for sample compression, it presents a technique that is interesting for a number of other applications. This is shown in a batch classification test on one of the standard data sets with results that are superior to the best reported in the literature.

The next chapter, chapter 5, looks at consequences of reward for statistical estimation of perceptual state. Compared to the previous, this chapter presents a longer-term model that becomes effective as the amounts of data and interactions with the environment grow. This chapter introduces the notion of *weak transduction*, that properly frames the problem of perceptual learning in the context of machine learning paradigms. In this framework, the connection between observations and their labels is made via feedback to the action selection of the autonomous agent. The resulting perceptual organization is induced by interactions of the agent with its environment to derive a measure of utility of the perceptual representation.

Chapter 6 targets perception of sequential input with relatively low dimension, but, again, in the face of short supply of training data. It presents an algorithm that is useful for vision-based gesture training of the autonomous agent. It attempts to combine the advantages gained in the previous two chapters. While the bias of the chapter 3 is expressly towards learning auditory perception, chapter 6 concentrates its attention on visual learning.

The thesis will conclude with a discussion of all presented methods, noting their advantages and shortcomings and identifying directions of further research.

1.4 Related Work

The work of this thesis would not be possible without the efforts of many people contributing to the fields of animal learning, robotics and machine intelligence. In this section, the most relevant work is only briefly mentioned. Further in the exposition, each chapter will be put in the context of relevant work more thoroughly, discussing the work in more detail to maintain proper grounding and the context of the relations.

1.4.1 Models of learning in biological organisms

There exists a large body of research in animal learning, as well as in building models of these learning processes that take place in animals and humans. The two main areas of special interest to this thesis are theory of conditioning, and categorical perception.

The initial work on animal learning is due to Thorndike [77], who formulated the *law of effect*. Pavlov [57], studied the formation of reflexes, or learned responses to presented stimuli. Rescorla and Wagner [63] developed a model of conditioning showing that it accounts for a number of phenomena encountered in animal learning.

Despite its popularity, the model is limited. Attempts to develop a more general model resulted in temporal difference model by Sutton and Barto [73], mixture based representations by Dayan and Long [15], differential model by Mignault and Marley [52], timing model by Gallistel [30], neural models by Schmajuk [67] and many others. However, it still remains difficult to find a simple model that explains all known phenomena in a satisfactory way.

In perceptual learning, many models were proposed by researchers studying biological systems (eg. [51, 47, 20]). Perceptual learning happens on both large and small scales, eg. [26]. Dror [22] Howard *et al.* [35], present experimental studies of high level skill-related perception concluding that the quality of perception is highly correlated with the learned skill in which it is used. Spelke [68], Zollinger [91], Uchikawa and Bointon [78], Birnham *et al.* [8] and many others examine the issue of differentiation between innate and learned perceptual abilities. In addition, experiments in three-dimensional object recognition, ([75, 24]) conclusively show that representational learning in the perceptual system is taking place as well.

These two areas are of central interest to this thesis. This work attempts to examine algorithms that embody the idea that the two types of learning in autonomous agents should occur simultaneously.

1.4.2 Pattern recognition

The thesis builds on many techniques of pattern recognition. Fukunaga in [29], as well as Duda and Hart [23], include a review of nearest neighbor classification rules and their incremental realizations. The enduringly popular kNN classifiers, used in a chapter of this thesis, are based on work by Hart and Cover ,[13], analyzing the asymptotic performance of the nearest neighbor strategy. Hart [33], formulated the condensed neighborhood rule, which in this thesis is applied incrementally and reexamined in light of later developments in support vector classification.

One of the techniques used in chapter 5 is based on the latent variable models, eg. [3, 4], and the Expectation-Maximization (EM) algorithm first reported by Dempster and Rubin [16]. Neal and Hinton [54], give an iterpretation of EM that makes the development of the algorithm for weak transduction possible, as shown in subsequent chapters. This interpretation was pointed out in the tutorial notes by Minka [53]. Cherkassky and Mulier [12], present a concise and systematic overview of the field of machine learning, which includes the connections of the more traditional methods with the methods of statistical learning theory [80], now gaining popularity. Devroye *et al.* [18], and Lugosi [48], provide analysis of distribution-free methods for classification on a finite sample from the point of view of statistical learning theory, which led to the development of the compression set techniques in this thesis. Osuna *et al.* [56], as well as Cauwenberghs and Poggio [10], developed some practical methods for training Support Vector Machines, which were instrumental to the formulation of the ideas for the algorithms of the chapter 3. The bound compression set is based on the work by Ben Hur *et al.* [2].

And, of course, I am indebted to the organizers and maintainers of the UCI repository of Machine Learning Databases [5], for providing the data sets used in this thesis.

1.4.3 Speech and Gesture Recognition

Finding semantic grounding for perception in action selection is the main thread that unites algorithms presented in this thesis. Earlier Roy and Pentland [64], addressed extraction of semantic content of a spoken language in early language acquisition. Semantics form a much broader and general "Superlanguage" that is universally understood. The authors focused on cross-modal perceptual evidence, whereas this work addresses the search for semantics in outcomes of the interaction with the outside world.

Chapter 6 is devoted to a classification of sequences. It draws on research in such inherently sequential domains as speech and gesture recognition. Rabiner and Juang [61], as well as Jelinek [39] provide an exploration in the fundamentals of speech recognition introducing some of the methods used in the chapter. Viterbi [81], developed a dynamic programming algorithm which is the basis of the inference mechanism of the algorithm, shown in chapter 6 and includes the modifications first introduced by Darrell and Pentland in [14].

Wilson [86], developed a system which is very close in spirit to the algorithm for gesture learning presented here, albeit with static structure. Hong *et al.* learned a finite state machine - based representation of gesture from a batch of gesture data [34], while Pavlovič and Rehg [58], experimented with Dynamic Bayes Nets to solve the problem of learning the discrimination between gate types from visual data. The gesture learning algorithm of this thesis performs incremental structure learning within the setting of on-line training.

1.4.4 Reinforcement learning

To a large extent the topic of this thesis was influenced by the ideas borrowed from reinforcement learning literature, with an introductory text by Sutton and Barto [74]. Kaelbling *et al.* gives an excellent survey of ideas and methods of the field [43]. Perhaps the most prominent examples of algorithms from reinforcement learning are Q-learning [82], temporal difference learning, due to Sutton [73], and more modern and formal techniques based on partially observable Markov decision processes [42, 50]. Thathachar [76], studied learning automata and developed a number of algorithms, in particular reinforcement pursuit, appropriate for associative tasks, addressed in the thesis. The work, perhaps the most relevant to the topic of the chapter 5, is the reinforcement driven algorithm for learning Vector Quantization by Likas [45].

Further references situating the work of this thesis within the context of the related research will be given in the course of the exposition in each chapter separately to maintain proper context.

Chapter 2

Framing the Problem

2.1 Overview

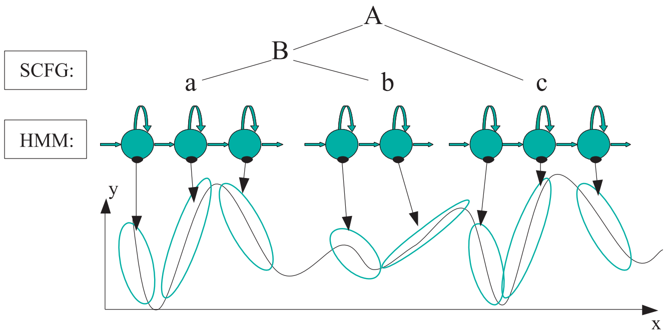
This chapter is dedicated to establishing the motivation for the work of this thesis. It describes some earlier work that motivated the development of learning algorithms presented here. It then presents a discussion of biological aspects of perceptual learning and deriving semantics from interactions with the environment. It then concludes with setting up the goals for the rest of the thesis and formulating the general problem, which this work addresses.

2.2 Earlier Work

Earlier in my work [6, 37, 36], I was interested in building large hierarchical recognition systems for detecting complex gestures and surveillance events. The essence of the system was realizing the expert knowledge in a set of grammatical descriptions of events of interest. The input signal to the grammatical analyzer was produced by a set of independent hidden Markov models (HMMs). Each HMM in the bank was independently trained on a set of training data corresponding to the "primitives" of interest. The primitives were selected by the expert on the basis of his or her subjective knowledge of the typical vocabulary of a larger scale structure, present in a gesture or a surveillance event. The grammatical structure was hand-coded in a

$$A \rightarrow Bc$$
$$B \rightarrow ab$$
$$a = \{A_a, \{\mu_i, \Sigma_i\}_{i=1..N}\}$$
$$b = \{A_b, \{\mu_i, \Sigma_i\}_{i=1..N}\}$$
$$c = \{A_c, \{\mu_i, \Sigma_i\}_{i=1..N}\}$$

a)



b)

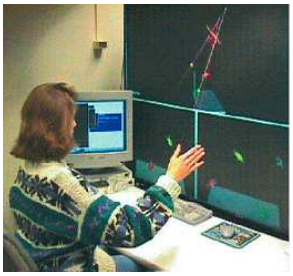
Figure 2-1: For the grammar in a) the underlying structured input sequence, shown by the curve on the bottom of b), is parsed with a set of hidden Markov models. HMMs are trained independently on some supervisor-selected subsequences forming the alphabet of the context-free grammar, and are combined into sentences parsed by the SCFG parser into sentences of the grammar.

stochastic context-free grammar (SCFG), while underlying primitives were difficult to decompose further into meaningful components. This resulted in a two-level representation of the time-series signal where the lower level encoded the statistics about the raw signal, and the higher level manipulated events.

The solution that was proposed in that work is illustrated in figure 2-1. In figure 2-1 b) the underlying input sequence is represented by the curve on the bottom, with a set of HMMs trained independently on some supervisor-selected subsequences. The HMMs form a vocabulary of primitive components, and are combined into sentences described by the grammar.

2.2.1 Applications

One application of this approach was embodied in the the system built for recognition of conducting gestures [6] (see fig. 2-2). The set of 5 Hidden Markov Models was trained on primitive hand gestures captured from a conductor with the *STIVE* vision system [88](fig. 2-2 a)). The input space consisted of hand velocities in X, Y, and Z coordinates. The primitive gestures were picked by an expert conductor from the knowledge about the vocabulary of conducting gestures and the structure of overall gesture was encoded by a Stochastic Context-Free Grammar, based on the set of conducting forms [65].

a) 

b)

```
PIECE ->    BAR PIECE        [0.7]
          | BAR              [0.3]
BAR   ->    TWO              [0.5]
          | THREE            [0.5]
THREE -> down3 right3 up3 [1.0]
TWO   -> down2 up2           [1.0]
```

Figure 2-2: The gestures captured by the *STIVE* vision system shown in a) are parsed by the SCFG parser [70], as given by the grammar in b). The non-terminals, representing semantic divisions of the input are shown in capitals, while lower case terminals correspond to the primitive HMMs.

a) 

b)

$A \rightarrow Bc$
$B \rightarrow ab$
$C \rightarrow cD$
$D \rightarrow de$
$a = \{E_a, \{x_i, y_i\}_{i=1..K}\}$
$b = \{E_b, \{x_i, y_i\}_{i=1..L}\}$
$c = \{E_c, \{x_i, y_i\}_{i=1..M}\}$
$d = \{E_d, \{x_i, y_i\}_{i=1..N}\}$
$e = \{E_e, \{x_i, y_i\}_{i=1..P}\}$

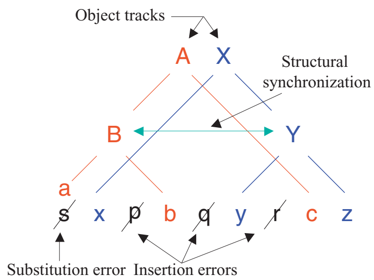
c) 

Figure 2-3: The task in the surveillance system is to detect activities in the parking lot (a). The task is complicated by the presence of multiple objects, interactions between objects and tracking mistakes. It is solved within the same hidden/overt formalism, where the vocabulary is formed from a set of tracker events. Each terminal in the grammar (b) carries along the event tag, E_x, and the complete object track, consisting of x and y positions at each frame. Object identity is recovered along with completing the parse of the chain of corresponding events. Interactions are interpreted via the presence of *structural synchronization* between independent parses (c). In this framework, the error recovery is of paramount importance, since terminals that belong to one parse tree have to be ignored in others.

Another embodiment of the representation shown above was a complete surveillance system [36], the task of which was parking lot monitoring (see fig. 2-3). The system connected a multi-object tracker [32], with the parsing engine, using the SCFG formalism. The tracker produced pieces of object tracks, which were probabilistically labelled with an object class – a person or a car; and tracker state events – `object-lost`, `object-found`, etc.. The higher level expert constraints, expressed via a grammar (structured as in fig. 2-3 b)), allowed the system to (a) collect pieces of tracks, coming from the tracker, into complete tracks, that would take an object through the scene; (b) detect interactions between objects via structural synchronization (fig. 2-3 c)); and (c) arrive at the most likely object labelling that would integrate spatial and temporal constraints in the maximally likely explanation. Each terminal in the grammar carried along the tracker event tag, E_x, and the complete object track, consisting of x and y positions at each frame, allowing the enforcement of track consistency constraints. Object identity was recovered along with the most likely interpretation of the chain of corresponding events. In this framework, the importance of error recovery was paramount. In fact it is mostly due to including the error into the formalism explicitly, that the system worked at all, since that allowed treating terminals of the event stream pertaining to one object to be separated from terminals of another.

Figure 2-4 shows one example of a detected interaction. The pane on the left shows the output of the parser assigning partial tracks to the complete parse and giving it an action label. The example in the picture is that of a person driving into the scene, parking the car and walking out of the parking lot. The bottom panel shows the temporal extent of each of the components of the "`Drive-In`" event.

2.2.2 Lessons

Many interesting questions arose in the process of building the systems shown in the previous section. The principal decision in the system architecture, was to sort out "hidden" and "overt" states. Clearly encoding the available knowledge and splitting the state set into hidden and overt allowed us to take small computational risks. In

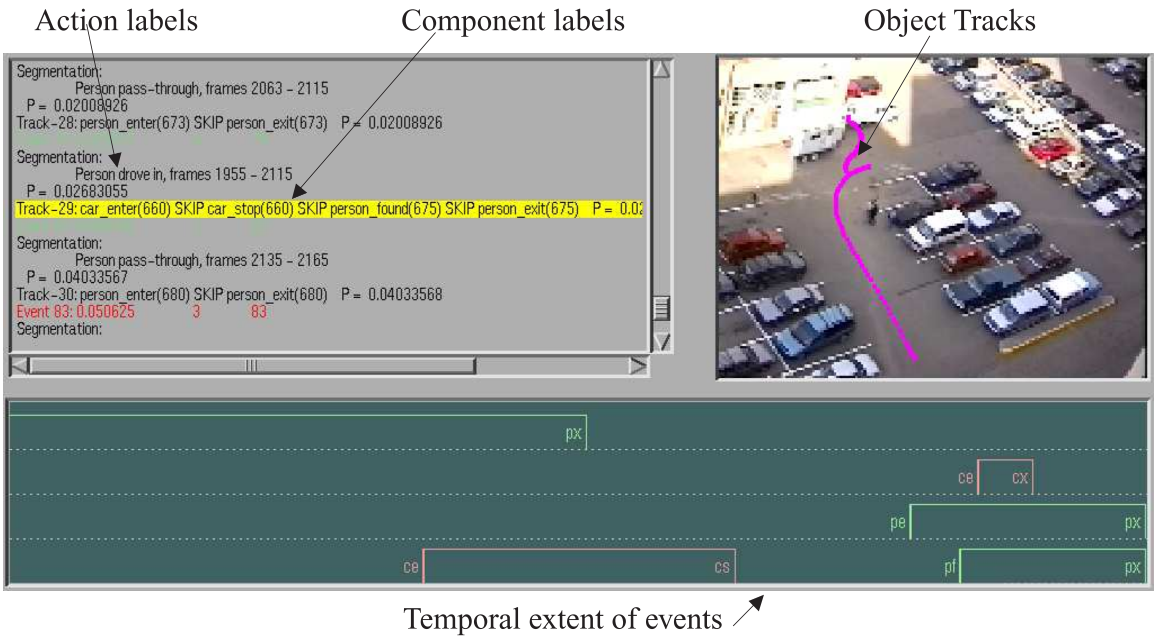


Figure 2-4: A screen shot of the surveillance system in the process of parsing the `drop-off` event. The top let panel shows high level event parses emitted at each step. The top right panel is displaying the static background with the object tracks contained in the current parse overlayed on it. The bottom pane shows temporal extent of the parses and the most likely object identities.

the context where states needed to be estimated, we used a set of simple tractable models. In the overt layer, where states are known, we could let ourselves use more computationally expensive model, such as SCFG, knowing that the worst possible case they would have to address would in general be very simple.

The most frequently asked question was about structure and parameter learning in this framework. Even though the point of building the full working system was to test inferential mechanisms in a large mixed structure, it always seemed like a good idea to include some sort of structural estimation into the inference. There exist algorithms for structural learning in complex hierarchical representations that grammatical descriptions provide (eg. [69]). However, the stumbling block for using it in our system was the fact that all these algorithms rely on either having a good set of models for the input alphabet, or collecting a large amount of transcribed training data that would allow the extraction of component models before the grammar learning takes place.

Fortunately, in some domains such extensive corpora of transcribed training data are available – natural language processing and speech recognition are the most prominent examples of that. The latest developments in these fields clearly demonstrate how far learning can be taken in the presence of large data sets. The area of computer vision has not been as fortunate. Difficulties in obtaining large sets of complex data seem to be primarily grounded in the fact that vision is not a communicative, but rather a declarative channel of information. Thus, the visual input is rarely as structured as the input in many problems related to language and speech. A visual channel does not readily possess a finite vocabulary, implying that collection of a large corpus of training data is not a simple task.

The experience of building the two complete systems above motivated us to explore possibilities of extracting the component primitives necessary for inferring the hierarchical structure. Consequently, that led to a larger question in a context of building an autonomous system that would not require elaborate crafting of its internal knowledge structure – is it possible to learn complex structures like the above from scratch, where only physical capabilities of its sensory input and mechanical constraints of the output actuators are known? And how can we do it quickly?

The answer to these questions lies to a large degree in the outcome of examining the search space that the algorithm needs to explore in order to find a desired solution, as well as identifying what solution is actually desired. In the absence of any constraints, all solutions and all parameter settings are equal – a phenomenon stated in a "no free lunch theorem" [87]. It can be further formally shown [18], that there is no classifier that performs universally better than any other one in such circumstances.

On the other hand, nature provides a proof by existence of the possibility of such algorithms. This seeming contradiction indicates the fact that additional constraints that are present in animal learning are derived from other sources, not related to the observational data. These sources give *meaning* to the component perceptual categories that the animal develops as a consequence experimentation with its environment via internal and external benefits.

Thus the goal becomes providing such constraints to the learning system in a consistent set of “benefits”, allowing to get “better” in some aspect of its decision making. This goal firmly places the agent in the context of its environment, as now it is clearly necessary for it to not only observe the environment in search of the regularity, but to actively experiment with it, in order to derive the grounding of the attained understanding in practical application of it.

2.3 Biological Inspiration

Edelman used the term “perceptual organization” as a collective term for a diverse set of processes that contribute to the emergence of order in the visual input [25]. This definition can, of course be extended to include all perceptual input. It is important, however, to make a distinction between perceptions and sensations.

2.3.1 What Do We Call Perception?

Humans and animals acquire knowledge about the outside world through sensations and perception. Sensations tell animals that there are “things” in the world outside themselves; perception tells them what and where they are located and what they are doing. Together, sensations and perceptions link their brains to the world and allow them to form mental representations of reality.

It is known [28], that more than 50 percent of the cerebral cortex in the human brain is devoted to visual processing, while much of the remainder is allocated to audition and speech perception. The visual sense is especially impressive as it allows humans to create and maintain detailed representations of their environment from a very noisy and sparsely sampled array of retinal sensors lit by rather blurry patches of light. From these simple inputs, complex models of the world are created that allow one to leap to instantaneous conclusions about what is “out there”.

The information, coming from the sensory input, is combined with and filtered through a massive array of internal representations, beliefs and expectations to form the basis for making an informed guess about the real state of the world and the ways to change it to one’s advantage. The fact that perception relies so much on the knowledge of the world implies that it should not be studied and understood in isolation – it must be linked with other cognitive processes, such as learning, memory, judgment, and problem solving [28].

In animals and humans, perception is a process that seems effortless – it is immediate and correct. Interestingly, current research in psychological aspects of perception suggests that continual learning is necessary in order to accurately perceive the world [1]. The process of keeping perception correct and current should be regarded as "learning", though in many ways it is unique since unlike in many learning processes in perceptual learning conscious awareness of it is not required.

2.3.2 Perceptual Learning

Perceptual learning, or the adjustment of perception to the stimuli of the environment, is sometimes distinguished from cognitive learning, the latter term being reserved for the modification of problem-solving behavior. Many phenomena suggest that perceptual learning happens in human infants and continues through adulthood.

The area of psychology that studies formation of categorical perception provides large body of examples of developmental formation of perception in humans. For example Dror [22], presents an experimental study of spatial perception in a group of US Air Force pilots with a control group consisting of non-pilot subjects. The authors concluded that pilots had better judgement of metric (but not categorical) spatial relations than non-pilots. Howard *et al.* [35], demonstrated a similar phenomenon in categorical perception of minor/major triads in a group of people with varying degrees of musical skill. Although the overall variation in the differentiation was not large, it was correlated with the level of the musical skill of the subject.

The question of what exactly is learned and what is innate in these situations, as well as in cognition in general, still remains unanswered. This topic has been the subject of a number of philosophic arguments since Plato's time [21]. A large body of work in the areas of psychophysics and neurobiology shows that the basic perceptual abilities of the human perceptual system (such as the ability to perceive luminance contrast in the visual system) are largely innate, while others (such as some varieties of object constancy) are learned and depend on experience [68]. Most significantly, the mechanisms of perceptual organization used by infants in learning how to perceive the world around them seem to be active throughout adulthood [25].

A finer degree of analysis of psycholinguistic aspects of color perception across cultures was conducted by Zollinger [91], Uchikawa and Bointon [78], and others. The studies focused on perception of color categories among culturally homogeneous control groups from Japan, Germany and the United States[1]. The results of the experiments concluded that the 11 basic primary colors are represented in most languages in the world, indicating their universal differentiability, while perception of derivative colors had clear cultural bias. In addition, Lynch *et al.* [49], clearly demonstrate that the difficulty in color naming in infants is not due to limitations in perceptual capabilities, suggesting that categories are not innate but are subject to tuning. These studies allow one to conclude that in fact perception is not independent of acquired abilities, yet lies very close to basic neurobiology.

Yet more evidence of developmental character of perception comes from research in language acquisition. Birnham *et al.* [8], test categorical perception of native and non-native speech in multiple age groups. Their findings indicate that not only does the categorical perception of native utterances get refined with age, but also non-native categorical perception gets "tuned-out", resulting in lower degree of distinctions made in non-native speech.

In addition, experiments in three-dimensional object recognition, ([75, 24]) hint that representational learning in the perceptual system is taking place as well. Performance of experimental subjects in the repeated task of recognition of a rotated object showed increasingly uniform response times to presented stimulus, regardless of the degree of rotation, while normally, the subjects' response time in recognition depends monotonically on the mis-orientation of the stimulus with respect to some canonical attitude. This indicates that the sparser mental representation, requiring a longer decoding time, was progressively replaced with a more memory-intensive, but

[1]The former study included two Japanese groups from areas with different degrees of Western influence and one German, while the latter – one Japanese and one American

"computationally" efficient representation.

All of the above lets one conclude that perceptual learning does in fact take place in animals and humans. For the purposes of this dissertation, an additional argument needs to be made. Very often perceptual learning is viewed in the framework that does not allow any amount of feedback to be delivered to the mechanism in control of perceptual learning. In certain instances this is a valid framework, for example, the three-dimensional object recognition experiments described earlier embody the self-evident mechanism known as unsupervised learning. In contrast, skill-related perception, such as a pilot's spatial perception, some instances of color perception, and examples from language acquisition are critically grounded in receiving a feedback from the environment in which they are applied. The latter type of perceptual learning is the focus of the thesis.

2.3.3 Semantic Grounding of Perception

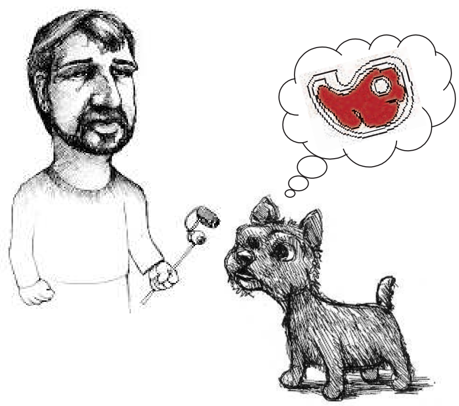


Figure 2-5: Consistent application of rewards and punishments reinforces the association between a behavior and its context.

The clearest example of refining the skill-related perceptual organization can be seen in animal training. For instance, dog obedience training pursues the goal of teaching a dog to reliably associate selecting a particular action with reward in the presence of some stimulus. In building up its perceptual organization an animal forms a set of perceptual categories where each category is somehow meaningfully different from any other in the set. The formation of the skill-related "meaning" of the category is guided by the outcome of the experimentation of the animal with the environment where this category is perceived existent. How is this meaning derived?

Imagine that a dog is learning to sit in response to an utterance "sit", pronounced by the trainer. Without questioning the dog's linguistic abilities, it can be safely argued that outside the context of getting a treat while sitting upon hearing the utterance, this acoustic pattern has no other meaning to the dog. The dog would not treat this utterance in any special way unless the treat was disposed a number of times in this exact co-occurrence.

Similarly, the semantic grounding of perception can be derived by an autonomous agent from the utility of it for achieving its goal. Categorization of perceptual input is not useful unless reacting to it in some way produces a benefit. Further, Bobick [7], argued that the perceptual categories form as "overgeneralization" of these benefits.

While learning the consequences of actions and relating these experiences to perception is in general hard, acquisition of perception- and behavior shaping experiences under training is a much simpler and direct task. There is much to learn in this respect from animal trainers. Living with domesticated animals for many thousands of years, people have found many ways of optimizing the animal's learning process. All these tricks can be summarized in just a few words – the point of training is making it easy for the animal to find the utility of a particular behavior.

The tricks of animal handlers deserve a careful study from designers of learning algorithms. In training a dog, trainers reward desired behaviors in desired circumstances. It is important that what is desired is clear to the animal. Then the consistent application of rewards and punishments will reinforce the association between the context of a behavior and the behavior that is being evoked. The training sequence in which the desired behavior is achieved usually with only small variations follows the following routine:

- Create, cause or wait for the behavior to occur;

  Eg.: **The trainer lures the dog to sit**. Every time the dog sits the trainers delivers a treat. As a result the dog learns that the trainer wants it to sit. *This identifies the action of sitting down as a target for further training.*

- Define the behavior;

  Eg.: **The trainer trains the dog to sit without luring**. Trainer directs the dog to make a distinction between which is a beneficial behavior and which is not. *This increases the probability that the dog will select the desired behavior* *if it wants a treat.*

- Cue and reinforce the behavior;

  Eg.: **The trainer trains the dog to sit on command**. The trainer presents a stimulus – an utterance and/or a gesture and rewards the dog *only* if it sits down when the stimulus is present. The dog is made to specialize – search for a perceptual context in which the behavior is beneficial. *This forms the perceptual category of the stimulus, giving it the semantic content of "sitting down".*

- Maintain the behavior.

  Eg.: **The trainer trains the dog to sit down on command in diffent places, with different distractors and rewards**. The dog is made to generalize – it refines its idea of the perceptual context, that the time of day and the color of the wallpaper do not matter. *This stage further refines the category.*

Switching momentarily to the domain of learning algorithms, one can easily identify parts of the autonomous system that are being affected in each stage of this routine. The first two steps are concerned with establishing the training context and identifying the action model under refinement; the remaining two refine the categorization capabilities of the perceptual system. Indeed, the first step picks a family of grossly relevant behaviors, while the second refines it by rejecting the ones that do not belong to the desired category. The third and fourth steps do the same for perception, while maintaining the association.

2.4 Technological Motivations

As Dorigo and Colombetti [19], point out, an argument in favor of the necessity of learning perceptual organization can be made from an observation that the goal of adaptation in an agent is learning how to best perform in its environment given its capabilities. Ultimately, the designer of an agent would like to encode some of the knowledge in the agent's algorithm in such a way that is convenient for the designer. Ideally it would let the designer formulate the agent's goals using semantically loaded statements like “Approach the corner and turn towards the exit”. The agent’s requirements are to formulate the relations between sensory and immediate control signals. A definite gap between agent’s and designer’s concepts of reality is usually filled with such abstract concepts as a “world model”, which assumes knowledge of every possible relation between the two representations, allowing for converting the designer’s statements into patterns of voltages. Models engineered in place of that “absolute knowledge” a) tend to reflect human’s point of view in forming sensory groupings; and b) inherit a structure of human linguistic constraints [19].

Put concisely, human-designed perception inherits human-imposed flaws. In contrast, in the learning framework, adaptation of agent’s perception results in realization of the world constraints in a form natural to the agent.

In addition, an argument can be put forward in favor of “developmental learning” – continuous learning for autonomous systems that develop new representations, skills and perception as necessary (e.g. [84]) – where it is inconvenient to frame the learning problem in terms of training and testing phases. Such separation is often done in traditional statistical learning for the purpose of accountability and simplicity of proofs. Reinforcement learning research in this case extends its grasp to a much more realistic setting and a satisfying goal: life-long on-line learning being one of its objectives. In designing autonomous agents there is no reason to build in a mechanism that at some point stops adapting to the new data. Learning and testing in this setting are the two parts of an ongoing process. It remains a big challenge to develop algorithms and architectures that possess the desired properties and are provably correct, while being capable of autonomous operation.

Of course, as was mentioned in earlier sections, autonomous independent learning cannot proceed when there are no clear self-evident constraints in the input data. For instance, understanding the meaning of pixels in the image received from the camera is all but an impossible task without properly grounding it in the context of the task of the agent. It is so for the simple reason that there is an abundance of patterns in the visual field, the vast majority of which are inconsequential to the task. Thus, the problem of learning a good perceptual organization becomes that of deriving utility of any given configuration.

The problems of utility and reward-driven optimization, have been the focus of a large body of research in reinforcement learning [43, 74]. Reinforcement learning assumes that the interactions of the agent with the environment result in some amount of benefit or loss to the agent and seeks a solution to the problem of finding the best behavior for the agent to maximize the benefit and minimize the loss on the long run.

The work of this dissertation borrows some very simple ideas from reinforcement learning, attempting to extend the statistical estimation techniques to use the additional information provided to the agent by external subjective reward. The setting of the thesis is strictly incremental, allowing to update parameters after observing every new data sample. The contribution of this thesis is in exploring some techniques for on-line learning equivalence classes in the perceptual space with the aid of reinforcement. There are three issues that I address in the remainder of document:

1. How to learn perceptual equivalence classes with higher utility?

   It is convenient to use techniques of unsupervised learning to estimate the perceptual field of the agent, which will probably contain the categories of interest. How can we extend these algorithms to take into account the utility of any given configuration to derive the set of categories that prove the most useful?

2. How to learn on-line efficiently?

   Learning on-line is difficult because only one data sample is examined at a time. All the previously seen data should have been already processed and stored away or discarded. This presents a problem that some old data, which was not immediately useful might turn out to be useful in the future. How should such data be utilized to provide a better support for future discovery?

3. How to learn the few useful categories and NOT learn the plethora of not so useful ones?

   Many trends and patterns in the agent's sensory field do not possess any useful correlations with the consequences of the agent's action selection mechanism. Therefore I will assume a *constructivist* view of learning of these patterns. The primitives are only learned once they have resulted in increase in the overall reward intake by the agent.

Presence of a trainer in this document is assumed only implicitly, as all algorithms presented in the technical chapters are built to operate on-line without any clear stopping conditions in sight. All algorithms are accepting one observation at a time, be it a single data point, or a sequence. In chapter 6 this distinction is not even made – the observation in that case is everything that has been seen so far. This setting naturally responds to including an explicit trainer into the model, and, of course, can be guided more or less successfully by a trainer via a particular order of presenting the data, or a particular structure of rewards, given to the algorithm. With some simple extensions, these algorithms could be trained via the training sequence that is used in animal training.

2.5 Problem Setting

Having given the historical, biological and technological motivations, the problem that this thesis addresses is that of building practical, usable systems that allow for the perceptual system of the agent to be refined over time with some degree of *naïve* supervision. The word naïve here is related to the fact that the type of the feedback that is of interest for this work does not assume any knowledge of the trainer about the details of the learning algorithm, while expressing the trainer's subjective degree of satisfaction with the learner's performance.

The memory-based algorithm of the next section was used as the basis for the auditory perceptual system of an animated dog, Duncan, the main actor of an interactive installation "Clicker by Eire", developed by the Synthetic Characters group (see fig. 2-6). Typically, in training real dogs, the first phase of the training procedure identifies a group of actions that will be rewarded. After that association is established, the trainer presents an observation – an utterance or a gesture – *simultaneously* with the onset of the desired action. During subsequent episodes the utterance gets associated with the action, and then can be used as a stimulus.

In “Clicker by Eire”, the user trains Duncan by first, reinforcing the desired behavior regardless of the observation context, and then, training the dog to recognize the command utterance and associate it with the appropriate action. The algorithm, described in the next chapter, implements the final stage of the training. After the dog is trained to prefer one action over others, the user, wearing a microphone, produces a command and rewards Duncan if it responds with an appropriate action. If the response is correct, then the user “rewards” the dog, who quickly[2] builds a small repertoire of voice commands and learns to associate them with particular behaviors. With practice, the trainer could shape the dog’s action selection mechanism to reliably perform tricks as directed by trainer’s voice commands, regardless of the language or the gender of the trainer.

Figure 2-6: Duncan is an artificial animated creature that accepts voice commands. The shepherd, acting as the user’s proxy, directs Duncan to herd a flock of virtual sheep.

The algorithms described in this document pursue the goal of improving the selective ability of the agent as the amount of interaction with the trainer increases. The training procedure used in this thesis consists of a series of steps, during which the agent is presented with a stimulus, here and below denoted by x^n, to which it responds by selecting an action,a^n. If the trainer finds this behavior appropriate, he or she rewards the agent by delivering to it some scalar value of a reward, r^n. With this in mind, the task of learning the perceptual organization for the agent can be stated as follows:

> ***Given a new observation, x, select an action, a, collect a reward, r, and update parameters of the perceptual model to increase the probability of receiving reward in the future.***

Generally, the correspondence between observations and actions can be arbitrarily complex, as well as the dependency of the reward on the past performance of the

[2]A vocabulary of 7 utterances would typically take 5–10 minutes to train for, which included playing out all the actions selected by Duncan to the end, keeping the training setting more-less realistic.

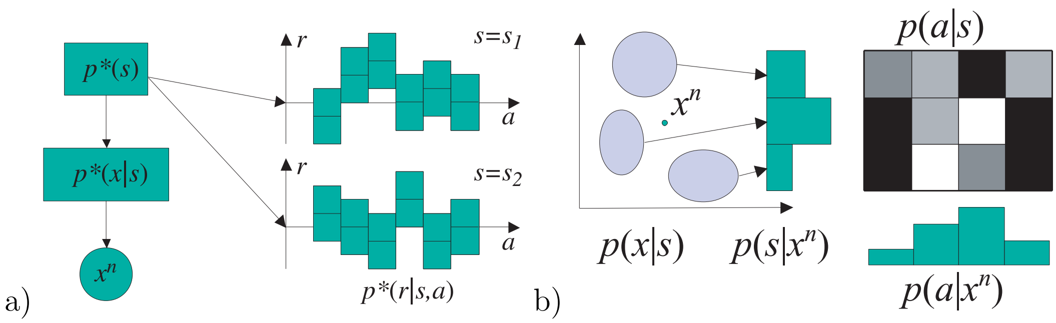


Figure 2-7: a) Generative model of the environment. Nature (shepherd) randomly selects a state, s_i and draws an observation (voice command, gesture, etc.), x^n, from a distribution $p^*(x|s = s_i)$. Upon observing the agent taking an action a^n, nature produces a reward (a treat) from the distribution $p^*(r|s = s_i, a = a^n)$. b) The solution which allows estimating the states (perceptual categories) along with the policy of action selection for the task of estimating the structure in a).

agent or past observations. However, for the task of associative training, the following generative model of production of observations and reward will be assumed (see figure 2-7a)):

Environment Model

1. Nature can be in one of a discrete set of states $S = \{s_i\}_{i=1}^K$;
2. Nature selects a state, s_i, from a probability distribution $p(s)$;
3. From a conditional distribution $p(x|s = s_i)$ nature generates an observation x^n;
4. Upon observing the agent taking an action a^n, nature produces a reward from the distribution $p(r|s = s_i, a = a^n)$.

The goal of the agent then is to estimate parameters of the model shown in figure 2-7b). In that figure $p(x|s)$ is the probability model of a category s, which is subject to estimation; $p(s|x^n)$ is a *belief state*, calculated for the incoming observation from current model parameters; $p(a|s)$ is a *policy*, that is related to the past correlations between state, action and reward; and $p(a|x^n)$ is the probability distribution based on which action is selected upon observing x^n. With this estimator, the overall structure of the solution will be as follows:

Agent Model

1. `Receive an observation,` x^n`;`
2. `Using current perceptual model,` $p(x)$`, compute the probability of` x^n `belonging to each of the perceptual classes,` s`,` $p(s|x^n)$`;`
3. `With the current belief about the class membership,` $p(s|x^n)$`, and the policy,` $p(a|s)$`, construct the distribution` $p(a|x^n)$ `according to which an action is selected;`
4. `Collect reward,` r`;`
5. `Update the policy,` $p(a|s)$`, so that actions resulting in larger positive rewards are more likely given the current belief state;`
6. `Update the perceptual model,` $p(x)$`, so that the belief state, resulted in the action with higher value of the reward has stronger preference for the most prominent state.`

This structure will be the general structure under which the algorithms of this document operate. Specific assumptions or simplifications will be explicitly stated in the text of each chapter as appropriate.

2.6 Notation

In the remainder of this document I will follow the notation given below wherever possible:

C_i	class i
$C_{max}^{(i)}$	maximum allowed number of samples in the model $\mathcal{M}_i$
$\mathcal{M}_i$	i-th class model
$\mathcal{M}$	full model dataset, $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^{K}$
$p(a\|s)$	policy – a probability distribution, according to which an action is selected for a given state
$p(s\|x^n)$	posterior probability of the state given an observation
$p(a\|x^n)$	probability distribution according to which an action is selected for a given observation
$p(\lambda\|s)$	a micro-state model membership
$Q(s, a)$	state-action value function, here, an average discounted value of the reward
r	instantaneous reward
$\bar{r}$	expected (discounted) reward
$\bar{r}^{+}$	positive part of expected reward
$\bar{r}^{-}$	negative part of expected reward
x^n	n-th input sample
$x_k^{(i)}$	k-th sample of the model $\mathcal{M}_i$
λ	a macro-state

Chapter 3

Learning Memory Based Models

When solving a given problem, try to avoid solving a more general problem as an intermediate step.

Vladimir Vapnik, 1995

3.1 The Goal

This chapter describes the algorithm for on-line utterance learning under reward. Here, the trainer gives a voice command to the agent and waits for the agent to respond with some action. If the action that the agent performs is the desired one, the trainer rewards the agent. Otherwise, the selected action remains unrewarded.

In this problem, the action selected by the agent serves as a label to the perceptual category to which the utterance is assigned, so the policy, $p(a|s)$, (recall section 2.5 and fig. 2-7b)) is very simple. For the rest of the chapter it will be set to a diagonal and ignored, that is, $p(a|x^n) = p(s|x^n)$. The reward is considered to be binary and is simply an indication of the validity of the guess of the algorithm about each new utterance. Thus the main problem is that of incremental supervised building of the memory-based model.

3.2 Overview

This chapter presents an algorithm for learning an on-line memory-based classifier of utterances of segmented speech. It follows the strategy set in the previous chapters for assigning labels to classes based on the utility of the model to the agent with respect to performing an associative learning task.

In building classifier systems one inevitably faces the choice of formulating a rule, or a discrimination function, according to which a new query, x^n, gets assigned a label, y^n. These classification rules can take an arbitrarily complex form and be formulated in a variety of ways, striving for a minimum of some loss function. The complexity of the problem dictates the appropriate choice of the solution.

3.2.1 Two Approaches to Solving the Problem

There are primarily two approaches to choosing such a discrimination rule – distribution-based and distribution-free. Distribution-based techniques typically construct the classifier in two steps – first, a class-conditional density of the data is estimated for each class; and then the decision boundary between classes is established based on the likelihood ratio criterion. The first step is typically the harder of the two, as density estimation is inherently an ill-posed problem and is affected by the data dimensionality, the size of the data set, feature scaling etc.

Another approach to finding a discriminant function is to solve the problem directly, without estimating the class-conditional densities. One example of such an approach is a k-Nearest Neighbors classification rule. Even though it is typically presented in context of density estimation tasks and contrasted to Parzen density estimators, the solution given by it is essentially distribution-free. Nearest neighbor classifiers have traditionally been very popular for several reasons: firstly, they are simple and, therefore, intuitive and easily interpretable. Secondly, and most importantly, they are mathematically tractable, while in the limit of training data approaching performance of best possible schemes – it can be easily shown that under very mild constraints local neighborhood rules achieve at worst a double of the optimal error rate in the limit of data [13] (see appendix A.1).

Lately, attention in machine learning community has turned to the statistical learning theory [80], that focuses on solutions to classification and regression problems in a distribution-free setting with a finite sample set. This shift in attention from traditional methods resulted in the development of kernel methods, in particular, support vector machines (SVM). SVMs implement a principled mechanism for solving distribution-free problems, that includes model complexity as a regularization parameter.

In general, distribution-based methods give better solutions if such a solution is possible to obtain. However, in a developmental framework (e.g. [84]) the distribution-based techniques are initially not applicable due to the necessity of obtaining a large amount of data.

3.2.2 What Is Here

This chapter will concentrate on solutions to a practical problem of training and developmental learning of class models in a multi-class classification scheme. Initially the models will have no information about the input space, and the agent will not have observed any part of the data prior to training.

The learning will proceed incrementally, allowing the algorithm to collect the data necessary to represent each class, while using a trial and error for class label assignment. Each class in the algorithm will be represented by a collection of "memorized", previously observed samples that were correctly classified by the classifier.

One complication to building such memory-based representation is that the model grows in size as more and more observations are seen by the agent. A large part of this chapter is dedicated to exploration of different methods of finding *compression sets* – subsets of the data that represent the rest of the data in a manner that is optimal in some respect.

3.3 Sampled Classification Rules

Constructing classifiers on-line is different from batch learning in one respect – learning starts with no data available at hand in the beginning of the process. Imagine the following incremental scheme – a classifier decides on the class membership of the query point based on the class, which is the nearest to it in some respect, and gets a

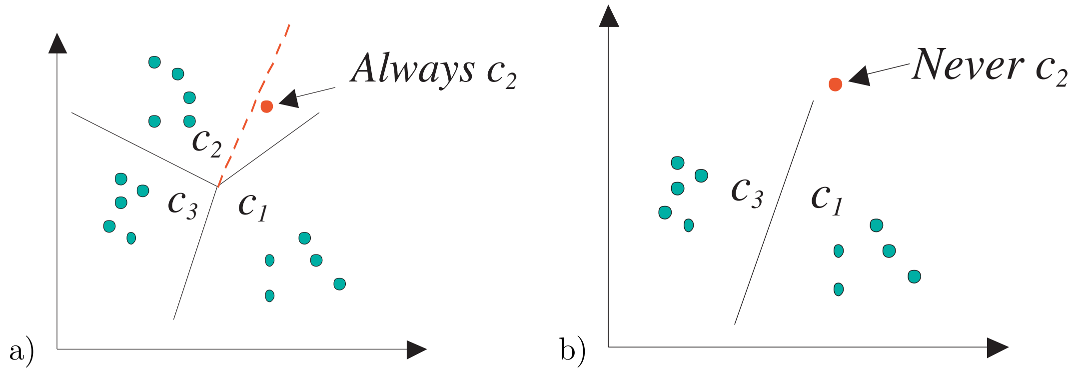


Figure 3-1: Illustration of effect of greedy label selection. The exploration strategy addresses the issues arising in incremental building of a classifier when little data are available: a) on the finite sample class boundaries are being re-estimated. The true class boundary, shown by the red dashed line, is different from the current empirical estimate (solid black line). A greedy procedure disregards the error rate and, therefore, queries falling inside the wedge between true and empirical boundaries always receive the same label; b) when no data have been seen for a class c_2 the query point is never assigned a label c_2.

feedback from an oracle indicating if the assignment was correct. If it is correct, then the observation is used to update the nearest class model, otherwise it is discarded. Such a scheme will be called *greedy*.

Starting with a small data sample, the greedy scheme encounters two problems, as illustrated in figure 3-1. Figure 3-1 a) shows the situation where the empirical boundary between classes is significantly different from the true one. In this situation all the points landing in the wedge between them will be given an incorrect label, regardless of how frequently the labels in that region have been rejected by the environment. Another problem is illustrated in the figure 3-1b). It shows a situation where one class has not received any data to fill its model to be a contender for some area of the input space. In this case, the model is never discovered and the data is simply rejected. These situations are routinely solved in reinforcement learning. An important insight that the incremental setting of reinforcement learning problems brings to the table is the trial-and-error approach.

3.3.1 Incremental Learning Algorithm

The the learning algorithm for the memory-based state representation needs to select samples from the set of the previously observed such that the misclassification rate is minimal. There are three problems that need to be explored for such setting: a) how to select the samples to include in the class models; b) how to use the trial-and-error approach to avoid the problems of greedy label assignment; and c) how to keep the computational complexity in check. This section addresses the first two, while the last problem is extensively explored in the following section.

Exploration Strategy in On-Line Classification

The trial-and-error paradigm to some degree quantifies the uncertainty of the agent about the quality of its action selection, allowing it to experiment with results of decisions that it currently finds sub-optimal. This "mistrust" is expressed via a probability distribution over actions given the current perceptual state. The action is selected by drawing a sample from this distribution. This strategy allows the estimated optimal action to be selected more frequently than the sub-optimal ones, not excluding the latter until some criterion is met [1].

Being an arbitrary construct, the probability distribution over actions given the state, $p(a|s)$, reflects the designer's strategy of balancing *exploration* and *exploitation*. Explorative strategy takes sub-optimal actions frequently, "exploring" the space of actions, or labels, to see if a higher reward would unexpectedly result. This strategy is appropriate for early stages of learning, while few observations are available and the model bias is large. On the other hand, when enough data has been seen by the algorithm and "surprise rewards" are rare, the exploration provides no additional information. In this case it is more beneficial to go with the knowledge and make optimal decisions at every step, or "exploit" the knowledge.

A form of the constructed probability distribution defines the exploration strategy. For instance, in a K-class classification problem, setting the probability of the class currently estimated to be optimal to the value of $1-\varepsilon$ and of every sub-optimal one to $\varepsilon/(K-1)$ for some small ε results in an ε-greedy strategy. A Boltzmann distribution with a decreasing temperature parameter is another popular choice. In this thesis a

[1] Typically this criterion is just "some time later".

Boltzmann distribution parameterized by an expected total reward is used to allow for "self-regulating" balancing of exploration and exploitation.

Label Assignment

To implement the exploration strategy, the learning algorithm uses a probabilistic label assignment, while incrementally building the model for each class. Imagine that the classifier needs to assign a label from a set $S = \{s_i\}_{i=1}^{K}$ to a query point x^n. Using a correct model of a distribution over the labels, $p(s|x)$, one can simply plug in the value of the query into the model and find a probability distribution over labels, as a value of $p(s|x)$ at the point x^n: $p(s|x = x^n)$. Then, the classification decision about the membership of x can be found to minimize some risk criterion (Bayes, Neyman-Pearson, minimax, etc.).

Each class in the algorithm is represented by a collection of previously observed samples that have been correctly classified by the classifier. The full classifier model $\mathcal{M}$ is represented by a collection of class models, $\mathcal{M} = \{\mathcal{M}_k\}_{k=1}^{K}$, each of which is a collection of correctly classified samples, $\mathcal{M}_k = \left\{x_i^{(k)}\right\}_{i=1}^{C_{max}}$. That is:

$$\mathcal{M} = \left\{x : x \in \left\{x_i^{(k)}\right\}_{i=1}^{C_{max}}\right\}_{k=1}^{K} \tag{3.1}$$

where $0 \leqslant C_k \leqslant C_{max}$ – number of samples added so far, and C_{max} is the maximum number of samples in the model, if applicable. The probability distribution over the set of classes is computed based on a pseudo-distance between the query point, x^n and each of the class models[2]. For a discriminant function for a class k, $g_k(.)$, that separates the set of samples that belong to the class from the rest of the data the pseudo-distance is calculated by simply evaluating the function at the query point, x^n. Computing the value of the discriminant for each class at x^n results in the vector of pseudo-distances:

[2]It is known that the choice of the distance metric can have a large impact on the performance of the algorithm.

$$\mathbf{d}(x^n) = \{g_k(x^n)\}_{k=1}^K \tag{3.2}$$

In particular, for the nearest neighbor type rule, the distance vector can be computed as a set of Euclidean distances to the nearest representative of each class:

$$\mathbf{d}(x^n) = \left\{\min\left\{\min_i(\|x^n - x_i^{(k)}\|), d_{max}\right\}\right\}_{k=1}^K \tag{3.3}$$

which is just a set of k distances from x^n to the closest sample of each class, capped by d_{max}. The algorithm assigns a label s_i to a class with a probability $P(s = s_i|x^n)$, which is computed in proportion to these pseudo-distances to each of the classes that are present in the model:

$$p(a|x^n) = p(s|x^n) = \frac{e^{-\zeta\mathbf{d}(x^n)}}{\sum\limits_s e^{-\zeta\mathbf{d}_s(x^n)}} \tag{3.4}$$

where parameter ζ controls the entropy of the resulting distribution. It is convenient to make it proportional to to the expected discounted reward: $\zeta = \beta\bar{r}$. This allows varying the effect of the computed distances on a resulting distribution, $p(a|x)$, such that entropy of it $\mathcal{H}(p(a|x)) \rightarrow 0$, as $\bar{r} \rightarrow \infty$, resulting in a "winner-take-all" nearest neighbor rule:

$$p(a|x^n) = p(s|x^n) = \frac{e^{-\beta\bar{r}\mathbf{d}(x^n)}}{\sum\limits_s e^{-\beta\bar{r}\mathbf{d}_s(x^n)}} \tag{3.5}$$

where, β is a free parameter that can be set to a small value in the beginning stages of the algorithm, and then slowly increased to result in sharper decisions when enough data is acquired.

Given a new data sample, the algorithm computes $p(a|x^n)$, and samples it to assign a label, $s^n = s_\ell$, and select an action a^n. If the action of the agent is rewarded, the sample x^n is added to the corresponding class model, $\mathcal{M}_\ell$. Subsequently, the value of the expected discounted reward is updated:

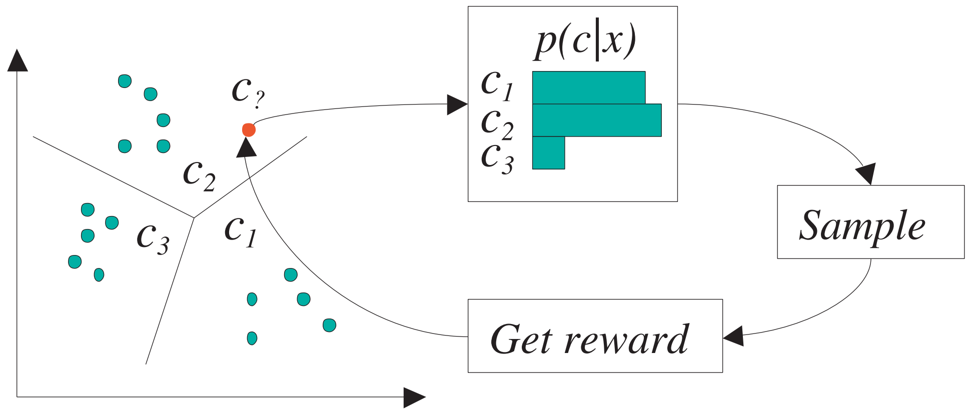


Figure 3-2: Illustration of the sampled incremental labelling procedure. A new query point, x^n, is used to compute a set of distances to the model classes. A label probability distribution, $p(c|x^n)$ is constructed based on these distances. The action is selected by sampling $p(c|x^n)$ and reward is collected. The reward dictates whether or not to include the query x^n into the model.

$$\bar{r}_t = \bar{r}_{t-1} + \alpha(r - \bar{r}_{t-1}) \tag{3.6}$$

where r is the momentary reward and α is a learning rate set to some small value, $0 \leqslant \alpha \leqslant 1$. This equation is equivalent to incrementally estimating a "drifting mean", or a baseline rate of reward. The parameter α is an exponential decay parameter that assigns lower weights to older values of reward in computing the sum of rewards up until now. It can be easily seen that setting α to be inversely proportional to the number of steps so far results in a sample mean.

The procedure is illustrated in the figure 3-2. Starting with $\mathcal{M}_i = \{\varnothing\}$ and $d_i = d_{max}$, the algorithm collects model samples by attempting to classify them with labels drawn from $p(a|x^n)$ and examining outcomes of this procedure. Sample assignments that resulted in receiving positive values of the reward lead to incremental construction of class models as summarized below:

Label Selection Algorithm

1. Receive an observation, x^n;
2. Compute distances to each of the class models, $\mathcal{M}_i$ (eqn. 3.3);
3. Calculate distribution over actions, $p(a|x^n)$ (eqn. 3.5);
4. Select an action $a^n = a_\ell$ by drawing a sample from the distribution $p(a|x^n)$;
5. Collect reward, r;
6. Update expected reward with the value collected (eqn. 3.6);
7. If $r = 1$ add sample x^n to the model $\mathcal{M}_\ell$;
8. Proceed to step 1

This algorithm allows to build models of states incrementally. Some parallels can be drawn between this algorithm and other well known on-line learning algorithms, such as a multi-class Winnow algorithm, for example. The focus here is on building a good model of the *state*, rather than the *decision boundary.* This goal is achieved by selecting examples to store in the model, from which the decision boundary is constructed when necessary.

3.3.2 Classification with Compression Sets

In a memory-based algorithm, where the model includes actual data samples, both memory and processing power requirements grow with increase of the number of stored samples. This necessitates additional methods to make the models sparse. The problem of finding *compression sets*[3] has been addressed in the case when all the data is available at once. Then, a subset of the data may be selected such that the performance of the classifier degrades the least. These techniques include the one proposed by Hart [33], and later reviewed by a number of researchers. The idea

[3]After Floyd and Warmuth [27].

of the technique is as follows. Given N data samples, find a subset of size $M < N$ such that the error committed on the remaining $N - M$ samples is minimal. The variations on this technique proposed in a variety of papers, eg. [11, 17, 31] included set reduction rules allowing for finding prototypes that do not belong to the original set, or re-labelling the subset data to achieve minimum error rates on the remaining set.

In the incremental setting, all data are not available at every step, but only the data kept in class models. At each step of the algorithm a decision needs to be made a) whether to include the successful query point in the prototype set; and b) which prototype to discard from the model to free up space for the new sample.

In this respect the proposed algorithm compares several strategies of sample compression – a) based on utility; b) based on minimization of the empirical risk; c) based on class boundaries; and d) based on margin samples.

Utility Compression Sets

Each prototype in the model is augmented with a *usage count*, Z_k, maintaining a count of how many times the prototype has been found to be the closest to a query, x^m; and a *lifetime count*, L_k which is a count of queries applied to the classifier since the prototype k was inserted into the model. Then the *utility index* of the prototype, U_k, is computed as a ratio of the corresponding usage count to its lifetime:

$$U(x_j^{(i)}) = \frac{Z(x_j^{(i)})}{L(x_j^{(i)})} \tag{3.7}$$

When the sample x^n is considered for addition to the model $\mathcal{M}_i$, and the model contains C_{max} prototypes - a maximum allowed number for the model, then the prototype with the lowest utility index is discarded and replaced with x^n.

Utility Compression Rule

1. `Increment the count` $L(x_j^{(i)})$ `of every model sample;`
2. `If the reward` $r > 0$
   (a) `Increment the usage count,` $Z(x_j^{(i)})$ `of the sample` $\mathcal{M}_i$`;`
   (b) `If` $|\mathcal{M}_i| > C_{max}$ `discard the sample with the smallest utility index;`
   (c) `Append the query` x^n `to the set` $\mathcal{M}_i$`;`

Minimum Empirical Risk Compression Sets

After the sample x^n is classified into the class C_i and added to the model $\mathcal{M}_i$ the algorithm removes a sample which is found to be redundant. That is, the k-th sample of the i-th set, $x_k^{(i)}$, which is the best predicted by the remaining samples in the model. Misclassification risk for the sample $x_k^{(i)}$ is computed as the probability of classifying it into a class $j \neq i$, given all the present model samples with $x_k^{(i)}$ removed from the set $\mathcal{M}_i$:

$$\begin{aligned} r(x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) &= p(C_k \neq i | x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) \\ &= 1 - p(C_k = i | x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) = 1 - \frac{e^{-\zeta \left\| x_k^{(i)} - x_{NN}^{(i)} \right\|^2}}{\sum_{j=1}^K e^{-\zeta \left\| x_k^{(i)} - x_{NN}^{(j)} \right\|^2}} \end{aligned} \tag{3.8}$$

The sample $x_k^{(i)}$ minimizing the risk in eqn. 3.8 is removed from $\mathcal{M}_i$:

$$\begin{aligned} \mathcal{M}_i &= \mathcal{M}_i \backslash \arg\min_{x_k^{(i)}} \left[r(x_k^{(i)}, \{x_{NN}^{(j)}\}_{j=1}^K) \right] \\ &= \mathcal{M}_i \backslash \arg\max_{x_k^i} \left[\frac{e^{-\zeta \left\| x_k^{(i)} - x_{NN}^{(i)} \right\|^2}}{\sum_{j=1}^K e^{-\zeta \left\| x_k^{(i)} - x_{NN}^{(j)} \right\|^2}} \right] \end{aligned} \tag{3.9}$$

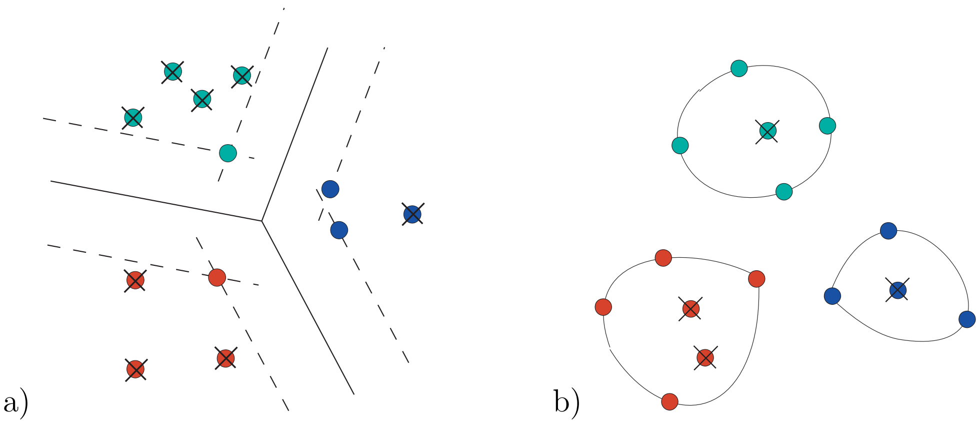


Figure 3-3: a) Margin compression sets are found by discarding the data lying away from the margins; b) Boundary compression sets are the data points lying on the cluster boundary of the classes.

Recall (from eqn. 3.4) that the last term of eqn. 3.9 is equal to the posterior probability of the class i given the sample, $x_k^{(i)}$. Hence the removed sample is the one with the maximum value of the posterior probability, $p(a|x_k^{(i)})$.

Minimum Risk Compression Rule
`If the reward` $r > 0$`:` 1. `If` $\|\mathcal{M}_i\| > C_{max}$ `compute the risk for every sample in` $\mathcal{M}_i$`;` 2. `Discard the sample with the smallest value of risk;` 3. `Append the query` x^n `to the set` $\mathcal{M}_i$`;`

Margin Compression Sets

Another approach to finding useful compression sets for the given set of models is based on the observation that the areas of the space where the classification boundaries are constructed are more important to be included in the models than the data lying in the areas distant from the decision boundary. The figure 3-3 a) illustrates the intuition behind this scheme. It shows that for making a classification decision the data located far from the margin between classes does not contribute to the decision boundary, and, therefore, can be safely removed.

Support vector techniques provide the solution for the problem of finding the data that lies in the margin between classes. The following is a condensed version of the idea of support vector machines. For the full discussion, the reader is directed to the literature (e.g. [79]).

Support Vector Machines Support vector machines are the realization of two ideas: a) *generalized optimal separating hyperplane* ; and b) *convolution of an inner product*. The solution to the classification problem is based on finding a hyperplane that separates the data with minimum error. In order to do it the input data consisting of observations and their labels, $D_i = \{(x_k, y_k = I(x_k \in \mathcal{M}_i))\}_{k=1}^{N}$, is first mapped to some high dimensional space, such that $\Psi_i = \{(z_k = \Phi(x_k), y_k = I(x_k \in \mathcal{M}_i))\}_{k=1}^{N}$ is the new dataset, where $\Phi(.)$ is a non-linear transformation.

Then, the optimal separating plane is determined by a vector w that minimizes the margin functional:

$$F(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i-1}^{N}\xi_i \tag{3.10}$$

subject to the constraint:

$$y_i(w \cdot z_i + b) \geqslant 1 - \xi_i \tag{3.11}$$

where C is fixed and ξ_i is a slack variable, making the second term of eqn. 3.10 a penalty for the data being inside the margin. The solution to eqn. 3.10 is found via quadratic optimization of the following *dual* problem: find $\{\alpha_i\}_{i=1}^{N}$ that maximize the Wolfe dual:

$$W(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (z_i \cdot z_j) \tag{3.12}$$

subject to constraints:

$$\begin{gathered} 0 \leqslant \alpha_i \leqslant C, \qquad i = 1, ..., N \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{gathered} \tag{3.13}$$

After the solution for the set $\{\alpha_i\}_{i=1}^{N}$ is obtained, w is found as:

$$w = \sum_{i=1}^{N} \alpha_i y_i z_i \tag{3.14}$$

while the classification decision is made based on the sign of the discriminant function:

$$g(x^n, D) = \left(\sum_{i:z_i \in \{SV\}} \alpha_i y_i (z_i \cdot \Phi(x^n)) - b \right) \tag{3.15}$$

An important insight of support vector algorithms is that in the calculations above, one never needs to consider the feature space explicitly, since all feature data is only used in the context of its inner product. However, the feature space can be chosen in a special way, so that the inner product in that space is represented by a kernel function in the input space, $(z_i \cdot z_j) = (\Phi(x_i) \cdot \Phi(x_j)) = K(x_i, x_j)$. In order to represent an inner product the kernel has to satisfy Mercer conditions. Known classes of Mercer Kernels include RBF, polynomial and sigmoid kernels. That is, equations 3.12 and 3.15 can be re-written as follows:

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{3.16}$$

$$g(x^n, D) = \sum_{i:x_i \in \{SV\}} \alpha_i y_i K(x_i, x^n) - b \tag{3.17}$$

The most important fact for the application of compressing the model dataset is that only the so called *support vectors* - data points lying on the margin boundary - will have their corresponding Lagrange multipliers, $\alpha_i \neq 0$. These are the vectors that are important for making the classification decision that the algorithm needs to keep on a long run.

The Margin Compression strategy estimates the new set of support vectors at every step when the model includes a new sample and possibly reduces it. In summary, the algorithm for determining the margin compression set is as follows:

Margin Compression Rule

`If the reward` $r > 0$`:`

1. `Append the query` x^n `to the set` $\mathcal{M}_i$`;`
2. `Find a set of support vectors,` $\mathcal{M}_i^{SV} \subseteq \mathcal{M}_i$ `as a solution to the classification problem` $\mathcal{M}_i$ `vs.` $\{\mathcal{M}_j\}_{j=1,j\neq i}^N$`;`
3. `Replace the set` $\mathcal{M}_i$ `with the reduced set` $\mathcal{M}_i^{SV}$`.`

Boundary Compression Sets

Yet another approach to incremental finding the compression set is based on the *support vector clustering* algorithm by Ben-Hur *et al.* [2].

The approach is based on an observation that in the case of nearest neighbor or margin classification the interior points of the cluster do not affect the result of classification. In contrast to the margin-based technique of the previous section, the approach of this section identifies the cluster's exterior points and uses them as prototypes for each model. Thus, not only the margin support vectors relevant to separating existing classes are kept in the model, but also the points on the cluster exterior that do not belong to the margin support set (see fig. 3-3 b)). The intuition is that the cluster support set is the set of potential support vectors if more classes are added to the classifier in the future.

The cluster support set is found with the use of a "kernel trick" - mapping the data to some higher-dimensional (feature) space and finding the smallest enclosing sphere for it. Then, the points lying on the surface of the sphere in the feature space can be shown to lie on the cluster's exterior in the original space.

Support Vector Clustering The support vector clustering algorithm solves the problem of finding the cluster boundaries by mapping the data from the input space to some high dimensional feature space and then finding a minimal enclosing sphere in that space.

The setting of the problem is very similar to that of the previous section. Since finding cluster boundaries only involves one class at a time, class index will be dropped for the remainder of this section. The data, $D = \{x_k\}_{k=1}^{N}$, is first mapped to some high dimensional space, such that $\Psi = \{z_k = \Phi(x_k)\}_{i=1}^{N}$ is the new dataset. The points enclosed in a sphere of radius R in the feature space satisfy the inequality constraint:

$$\|\Phi(x_j) - a\|^2 < R^2 + \xi_j \tag{3.18}$$

with slack variables $\xi_i \geqslant 0$ and a the sphere center. The problem is now to minimize R^2 subject to the above constraints. Introducing the Lagrange multipliers for the constraints, Wolfe dual and a kernel, $K(x_i, x_j)$, to express the inner product as before, the objective functional is given by:

$$W(\alpha) = \sum_{i=1}^{N} \alpha_i K(x_i, x_i) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \tag{3.19}$$

Again, the solution for Lagrange multipliers α_i is found by quadratic optimization, where $\alpha_i \neq 0$ indicates that the point lies on the surface of the minimal enclosing sphere in the feature space, and, thus, on the cluster boundary in the input space.

The Boundary Compression strategy estimates the new set of cluster boundary points at every step when the model includes a new sample, and possibly reduces it. In summary, the algorithm for determining the boundary compression set is as follows:

Boundary Compression Rule
`If the reward` $r > 0$`:` 1. `append the query` x^n `to the set` $\mathcal{M}_i$`;` 2. `Find a set of boundary points` $\mathcal{M}_i^B \subseteq \mathcal{M}_i$ `as a solution to the minimum enclosing sphere problem;` 3. `Replace the set` $\mathcal{M}_i$ `with the reduced set` $\mathcal{M}_i^B$`.`

	Compression Set of $\mathcal{M}_i$	Decision Metric
KNN	$\mathcal{M}_i \setminus \arg\min\limits_{x_j^{(i)}} \left\{ \frac{Z(x_j^{(i)})}{L(x_j^{(i)})} \right\}$	$d_k = \left\lVert x^n - x_{NN}^{(k)} \right\rVert^2$
KNE	$\mathcal{M}_i \setminus \arg\max\limits_{x_j^{(i)}} \left\{ \frac{e^{-\zeta \left\lVert x_j^{(i)} - x_{NN}^{(i)} \right\rVert^2}}{\sum\limits_{k=1}^{K} e^{-\zeta \left\lVert x_j^{(i)} - x_{NN}^{(k)} \right\rVert^2}} \right\}$	$d_k = \left\lVert x^n - x_{NN}^{(k)} \right\rVert^2$
SVC	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : \left\lVert \Phi(x_j^{(i)}) - a_i \right\rVert^2 < R_i^2 + \xi_j \right\}$	$d_k = -\left(\sum\limits_{j=1}^{\lvert\mathcal{M}_k\rvert} y_j \alpha_j K(x_j^{(k)}, x^n) - b_k \right)$
SVS	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : \left[w_i \cdot \Phi(x_j^{(i)}) + b_i \right] > 1 - \xi_j \right\}$	$d_k = \left\lVert x^n - x_{NN}^{(k)} \right\rVert^2$
SVL	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : \left[w_i \cdot x_j^{(i)} + b_i \right] > 1 - \xi_j \right\}$	$d_k = -\left(\sum\limits_{j=1}^{\lvert\mathcal{M}_k\rvert} y_j \alpha_j (x_j^{(k)} \cdot x^n) - b_k \right)$
SVM	$\mathcal{M}_i \setminus \left\{ x_j^{(i)} : \left[w_i \cdot \Phi(x_j^{(i)}) + b_i \right] > 1 - \xi_j \right\}$	$d_k = -\left(\sum\limits_{j=1}^{\lvert\mathcal{M}_k\rvert} y_j \alpha_j K(x_j^{(k)}, x^n) - b_k \right)$

Table 3.1: Summary of compression rules and sampling strategy. For each classifier, the table shows the set reduction technique in the second column, followed by the pseudo-distance metric based on which the probability of label assignment is computed.

Summary of Compression Set Rules

The summary of the rules for finding a compression set of the class model is shown in the table 3.3.2. The table lists six different set reduction techniques. The table shows the strategy of the set reduction for each classifier, followed by the pseudo-distance metric for computing the probability of action selection, $p(a|x^n)$, as shown in eqn. 3.5. :

1. KNN
   Utility-based sample reduction strategy. When a new sample is added to the model $\mathcal{M}_i$ and the model size exceeds the maximum allowable, $C_{max}^{(i)}$, the sample with the lowest utility index, U_k (see eqn. 3.7, is discarded from $\mathcal{M}_i$. The set of class distances for eqn. 3.5 is computed as Euclidean distances to the closest representatives of each class model, $x_{NN}^{(k)}$.

2. KNE
   Minimum risk strategy. First, a sample is added to the rewarded model, $\mathcal{M}_i$, then all samples in $\mathcal{M}_i$ are examined in turn calculating the posterior probability of the class i with the sample removed. The one having the least uncertainty

about its label (highest value of the posterior probability, $p(a = i|x_k^{(i)})$) is removed from $\mathcal{M}_i$. Decision metric is the nearest neighbor Euclidean distance, as in KNN;

3. SVC

   Cluster Boundary rule. When a new sample x^n is added to the model $\mathcal{M}_i$, the support vector clustering is performed on the model in order to identify points not lying on the cluster boundary, which are subsequently removed. An action is selected with probability based on the value of the discriminant function of eqn. 3.17 at the query point, x^n;

4. SVS

   "Silly" SVM with the Margin rule and a vector norm decision metric in the input space[4]. After a new sample is added to the model $\mathcal{M}_i$ a support vector classification "one against the rest" is solved to find the set of Support Vectors lying on the margin boundary. The remaining samples are removed from the model $\mathcal{M}_i$. The decision metric is the nearest neighbor Euclidean distance, as in KNN and KNE models;

5. SVL

   Linear SVM with the Margin rule. In the case of the Linear SVM input and feature spaces coincide. The set of support vectors is identified after addition of a new sample, as in SVS. An action is selected with probability based on the value of the discriminant function of eqn. 3.17 at the query point, x^n;

6. SVM

   RBF Kernel SVM with the Margin rule. The set of support vectors is identified after addition of a new sample, as in SVS and SVL models. An action is selected with probability based on the value of the discriminant function of eqn. 3.17 at the query point, x^n.

[4]It is "silly" because the decision metric does not match the metric used in the compression rule.

Figure 3-4: The sheep.

3.4 Experimental Results

Initially, the algorithm of this chapter was developed for the "Trial by Eire" installation by the Characters group. The installation was used with a large number of users in a variety of noise conditions and showed a fairly robust performance even with the simplest utility-based compression. While only qualitative results have been collected from "live" runs of the algorithm with KNN compression set rule, it was tested on three standard data sets from the UCI Machine Learning Repository [5], to get more definite quantitative results.

In all experiments reported in sections 3.4.2, 3.4.3 and 3.4.4 the results are averaged over 120 runs with data randomly reordered. For the KNN and KNE set compression methods, the parameter C_{max} was set to 20.

3.4.1 Trial by Eire

In the live system, the user would have a goal of training the artificial dog, Duncan, to respond to a set of 5-7 sheep herding voice commands. Although users varied in their gender, and native tongue, the training would normally take 2-3 minutes for the whole set. The main difficulty in the training was that the initial guess of the first utterance of each class was typically time consuming. It is explained by the fact that the distribution over actions from which the action label is sampled would be almost uniform for a new utterance of each class. This emphasizes the importance

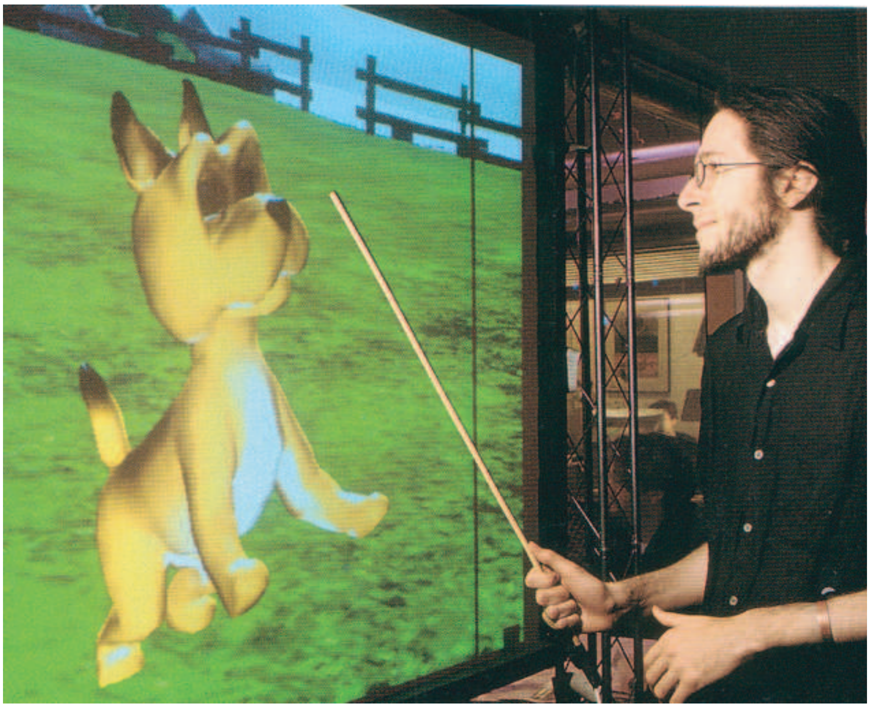


Figure 3-6: Training Duncan.

for practical applications of the action pre-conditioning before the perceptual and associative training takes place, providing a nice parallel to the initial discussion of lessons from animal training in section 2.3.3.

While the algorithm used in Duncan was very simple, it raised many interesting questions mainly concerned with economizing the memory available for the model storage. Exploration of different options lead to formulation of the compression set rules shown above and, subsequently, the technique of SVM-based sequence classification, presented in full in the following chapter.

Figure 3-5: Duncan.

A more thorough examination of the implications of different compression rules and decision metrics is necessary to properly evaluate the algorithm. The evaluation is performed on three data sets collected from the University of California, Irvine, Machine Learning repository [5]. Three data sets were used: a) Iris data set; b) Wine data set; and c) Japanese Vowel data set. The following presents them in order.

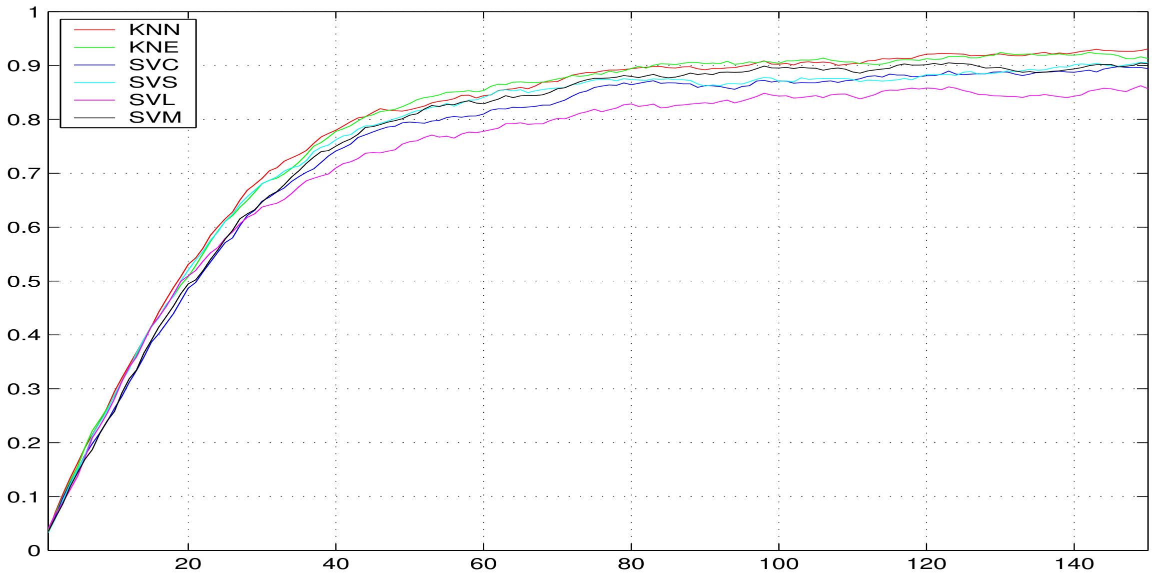


Figure 3-7: Results of running the algorithms on the Iris dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far.

3.4.2 The Iris Set

The Iris data set consists of 150 samples of measurements of 3 species of the Iris plant - Iris Setosa, Versicolour and Virginica. One class is linearly separable from the other 2, while the latter are not linearly separable from each other. 4 characteristics of each plant are measured: 1) sepal length in cm.; 2) sepal width in cm; 3) petal length in cm; 4) petal width in cm.

	Accuracy	Prototypes
KNN	0.927362	19.397222
KNE	0.918276	19.580556
SVC	0.893316	18.625000
SVS	0.902188	9.927778
SVL	0.853878	3.561111
SVM	0.899894	9.630556

Table 3.2: Results on the Iris set.

The results are presented in the figure 3-7 and the table 3.2. The second column of the table shows the accuracy attained by the incremental classifier after seeing the whole data set. The third column shows the number of retained prototypes per class. All results are averaged over 120 runs.

This proved to be a simple data set where nearest neighbor classifiers achieved marginally best performance, while retaining the maximum number of prototypes (∼38% of the data). Linear SVM performed the worst but only used 3.6 prototypes per class (7.2% of the data).

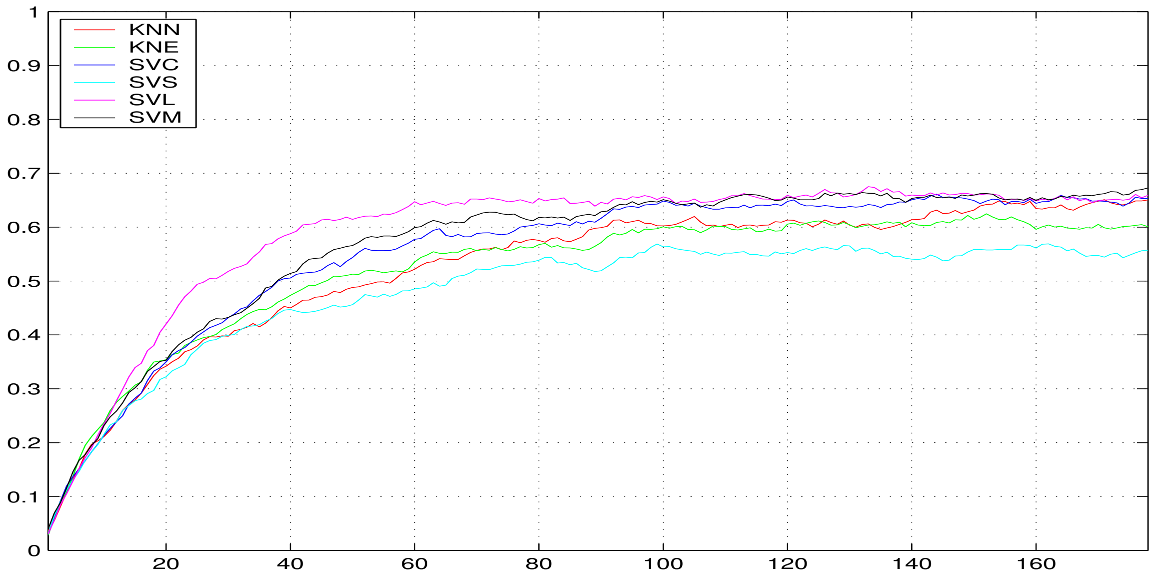


Figure 3-8: Results of running the algorithms on the UCI Wine dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far.

3.4.3 The Wine Set

	Accuracy	Prototypes
KNN	0.646044	19.363889
KNE	0.600730	19.205556
SVC	0.648654	16.575000
SVS	0.549376	14.013889
SVL	0.652421	4.547222
SVM	0.663934	9.341667

Table 3.3: Results on the Wine set.

The Wine data set consists of 178 samples of measurements of 13 parameters of 3 types of Italian wine.

The results are presented in the figure 3-8 and the table 3.3. The second column of the table shows the accuracy attained by the incremental classifier after seeing the whole data set. The third column shows the number of retained prototypes per class. All results are averaged over 120 runs with random re-ordering of samples.

The "silly" SVM showed noticeably the worst performance of the set, while retaining 23.5% of the data. In contrast, the RBF and linear SVMs turned out to be the best, at the expense of storing away 15.6% and 7.62% of the data respectively. The latter, as well as a good performance of the boundary rule indicate that the classes were well separable.

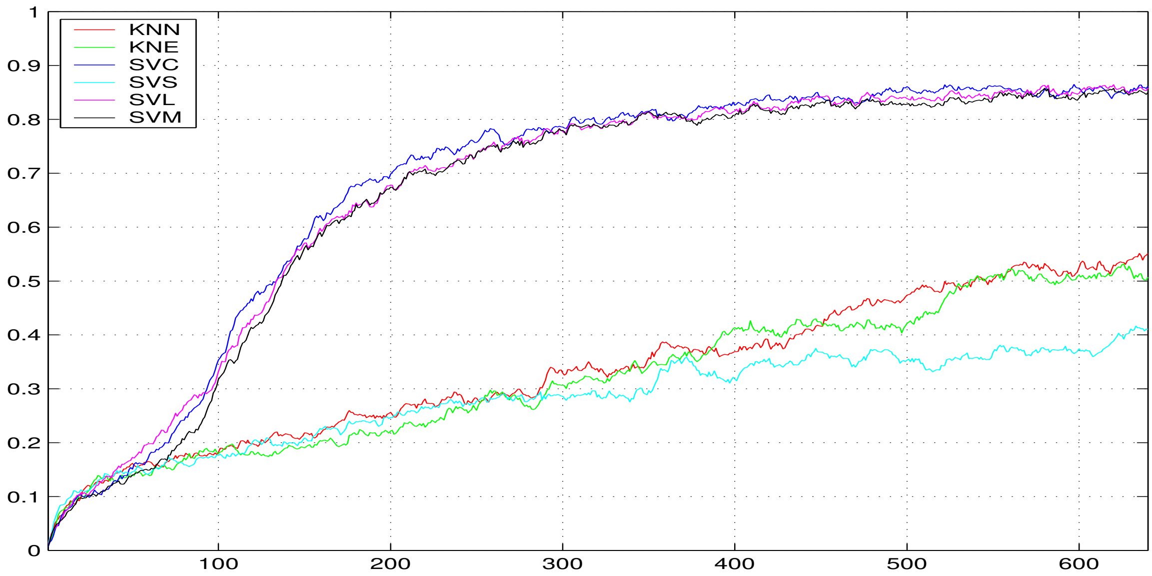


Figure 3-9: Results of running algorithms on the Japanese Vowel dataset. The average accuracy attained by the algorithm is plotted against the number of observations seen so far.

3.4.4 The Vowel Set

The last experiment was run on a Japanese Vowel data set. The set consists of 640 sequences of 12 LPC cepstral coefficients. The sequences consist of 7 to 29 coefficient vectors each and are collected from 9 male speakers, pronouncing the Japanese vowel "æ".

	Accuracy	Prototypes
KNN	0.545064	16.715741
KNE	0.508990	16.468519
SVC	0.858571	17.693519
SVS	0.411129	12.515741
SVL	0.855106	14.337963
SVM	0.849584	18.772222

Table 3.4: Results on the Japanese Vowel set.

The experiments were run on time-normalized data, as will be shown in the next chapter, which resulted in it occupying 252-dimensional space. Even though time-normalization presents a violation of the no-lookahead principle, it will be shown later that there exists a method, which allowing to avoid previewing the data in practice, while delivering at least the same performance. That allowed the experiments to be conducted with normalization.

The results are presented in the figure 3-9 and the table 3.4. The second column of the table shows the accuracy attained by the incremental classifier after seeing the whole data set. The third column shows the number of retained prototypes per class. All results are averaged over 120 runs with random re-ordering of sequences.

The "silly" SVM again showed the worst performance while storing 17.6% of the data. Linear, the boundary rule, the RBF and the Linear SVMs and this time showed dramatically better performance than other contenders (24.9%, 20.1% and 26.4% of data stored).

3.4.5 Discussion

While being so simple the algorithm contains a sketch of a powerful idea of using whatever little information that is available from the environment to refine perception. This insight will be developed further in the subsequent chapters, however, what is most interesting in this chapter is the different ideas of forming the reduced memory-based models and the performance that the algorithm achieves using them incrementally.

Theoretically, two questions still remain open - a) what can be said about the convergence *rate* of the nearest neighbor classifier? and b) what can be said about the convergence rate of the classifier using a compression set? Unfortunately, currently not much can be concluded - some details are given in appendices A.2 and A.3. Those results are definitely not the final word of theory, however, they require much more effort before they become useful for practical purposes.

The Iris data set gave no surprises. Classification rates of all algorithms were relatively high - around 90%. In this situation, only the linear SVM showed a slightly poorer performance taking the most of the performance loss on the two non-separable classes of the set. KNN and KNE showed the best accuracy. The low dimensionality-to-sample size ratio provided good conditions for the nearest neighbor rules.

Despite the fact that the Wine data set is usually referred to as "easy", the incremental classification of it was not very successful. The main problem that the all classifiers had to deal with was the uneven scale across different dimensions. While whitening[5] the data would have improved the results, it would be a violation of the desired property of not having to see the data before trying to classify it. In the end, the main loss was sustained by the "silly" SVM, combining the nearest neighbor rule with the Support Vector reduction rule.

On the data set of the most interest - the Japanese Vowel set - the dramatic improvement in performance achieved by the SVM-based techniques illustrates a good generalization ability that the SVM can provide. The high dimensionality of the space with barely 3 samples per dimension for all 9 classes showed to be an obstacle for the local neighborhood rules. The situation with neighborhood rules would probably not be so grim had they not run out of data. On the larger dataset, the convergence of these rules can be traced further, however, for the purposes of this application the data set provided about 70 repetitions of each class of utterances. This is not satisfactory for a live training application. SVM-based compression rules achieved an accuracy of about 70% after 20 repetitions each, which is much closer to the target convergence rate.

The performance of the "silly" SVM proved that it is almost never a good idea to use an SVM compression set with a nearest neighbor decision rule. The explanation for this is very simple - the neighborhood rule used for label assignment was computed based on the Euclidean metrics of the input space, while the margin is computed in the feature space, which imposes a non-Euclidean metrics on the input space.

Perhaps the biggest surprise of all the experiments was the excellent performance of the boundary compression rule. Indeed, unlike with margin rules, to find a sparse subset each class was examined independently of the rest. In addition, this strategy tends to pick outermost samples of the class, without any concern for outliers. Despite these two shortcomings, on the examined data sets the performance of the boundary compression rule was on par with, if not better than that of margin rules. The performance seems to be a property of the data set. However, in higher dimensions, this technique has a much better chance of succeeding.

The conclusion of this chapter is that in problems with low dimensionality-to-

[5] That is, transforming it to occupy a unit sphere.

sample size ratio, neighborhood rules perform well and are worth considering at the lower computational cost. As the ratio grows higher, margin-based rules begin to gain their edge. In that situation, as well as when there is a reason to believe that the classes are reasonably well separated, boundary rules, being computationally simpler than full margin rules, should provide faster convergence, especially in situations where some of the classes are not yet represented.

Chapter 4

Support Vector Classification of Sequences

So far nothing has been said about the implications of the sequential character of the data. All the data, sequential or not, has been treated in the previous chapter as if it all lived in the same space and that every data sample was easy to plot and compare to another. The reality of utterance processing is that this is not true. The input data set typically consists of multivariate short-time spectral sequences of variable length. Examples of the input to the classification algorithms presented in this chapter are shown in figure 4.

The figure shows example waveforms of utterances belonging to two classes, their short-term spectral representation (second column) and the time-aligned spectral sequences (third column).

Support vector machines have been successfully used in many fields for classification and regression tasks. However, there is a difficulty of using it with sequences of observations, since they "live" in a space with variable dimension. In order to apply support vector machines to find solutions in the space of sequences, either the input data has to be time-aligned in such a way that all input sequences are mapped onto a space with a fixed dimension, or an algorithm needs to be developed that allows direct treatment of sequences. Some solutions for time-normalization has been shown by Rabiner and Juang, [61], which is the basis for the approach taken in section

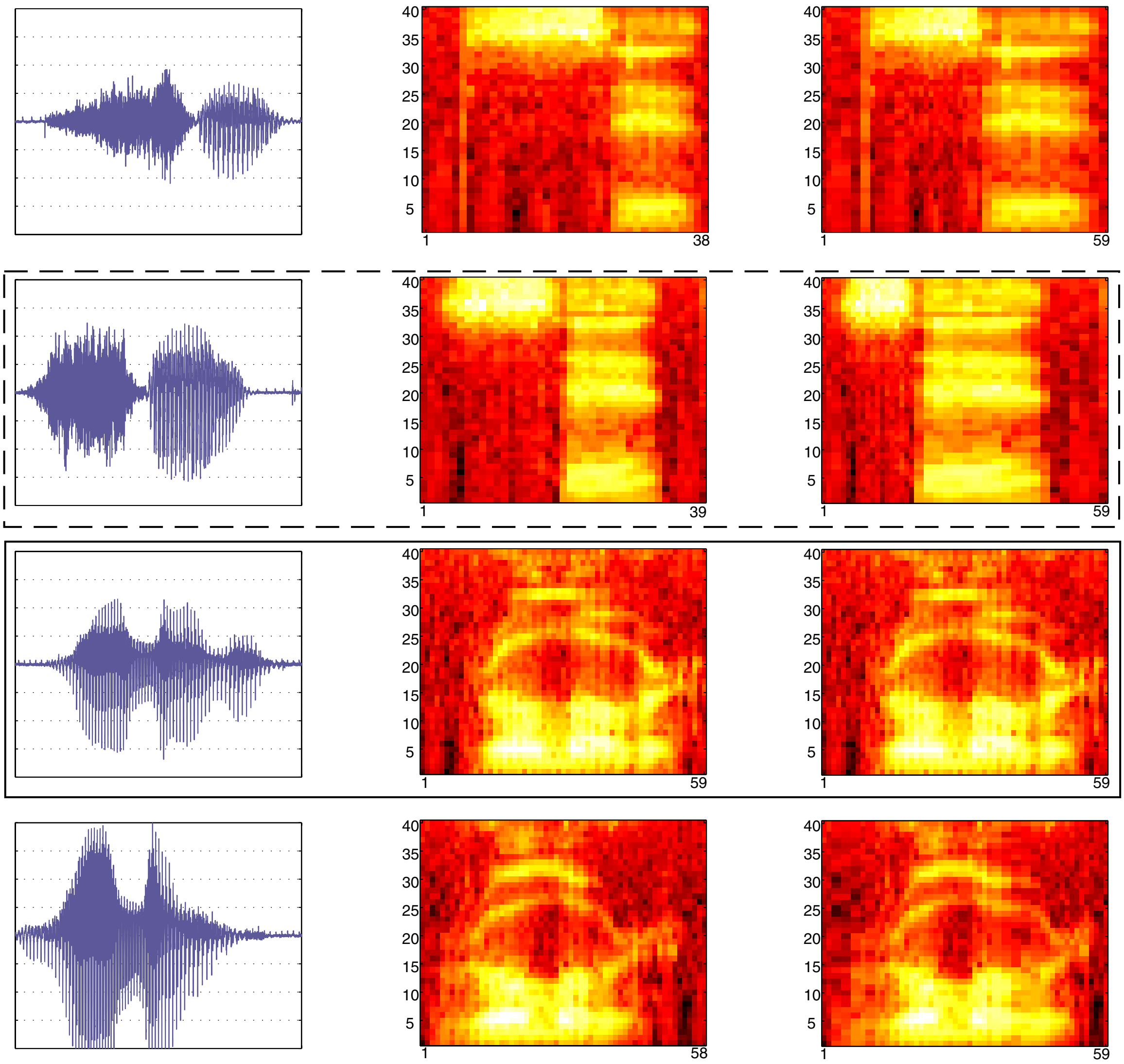


Figure 4-1: An example of the input data for classification. The left column shows waveforms of utterances coming from two classes - “Sit” (first two rows), and “Roll over” (second two rows). The middle column - their corresponding Mel-scale cepstral coefficients. The waveforms and coefficient sequences are of different lengths, as can be read off the horizontal axis. The right column shows the output of the time-alignment algorithm. Minimum distance spectrograms are outlined. The one outlined with a solid rectangle is the master prototype to which all remaining data are aligned. After the dimensionality normalization has been applied the spectrograms have the same dimensionality and can be used with standard classification techniques. In contrast, the support sequence machine, described in the text, can work with unnormalized spectrograms of the second column directly.

4.1.1. Support vector algorithms for treating sequential data have also deserved some attention. For instance, Jaakkola and Haussler, [38] use the gradient space of fixed size induced by the parameters of an HMM trained on input sequences to derive the kernel to be used within the SVM. However, developmental incremental setting of the problem addressed in this thesis disallows direct application of these techniques.

The following two sections describe the time-normalization and the dynamic alignment kernel solutions in turn.

4.1 Dimensionality Normalization

One approach to Support Vector classification of sequences is to time-align the data in the data set, which effectively makes the data have equal dimension across the data set. For the purposes of time alignment and dimensionality normalization, a technique of *Dynamic Time Warp* (DTW, eg. [61]) is frequently used.

Dynamic Time Warp procedure aligns (re-indexes) one sequence of observations with another such that the total per-sample distortion measure is minimized.

4.1.1 Time alignment

Consider two patterns, $\mathcal{X}$ and $\mathcal{Y}$, represented by sequences of vector-valued features, $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T_x})$ and $(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{T_y})$, where sample indices, i_x and i_y in the sequences $\mathcal{X}$ and $\mathcal{Y}$ range from 1 to T_x and T_y, respectively. The general time-alignment between the two sequences involves the use of two warping functions that relate the indices of the two patterns to some common time axis, that is:

$$i_x = \phi_x(k), \qquad k = 1, 2, ...T \tag{4.1}$$

$$i_y = \phi_y(k), \qquad k = 1, 2, ...T \tag{4.2}$$

The global dissimilarity between the patterns can be measured considering the pair of the functions $\phi = (\phi_x, \phi_y)$ as the sum of T local dissimilarities, $d(\mathcal{X}[i_x], \mathcal{Y}[i_y])$, weighted by the cost of re-indexing at k-th step, $m(k)$:

$$\mathcal{E}_\phi(\mathcal{X}, \mathcal{Y}) = \sum_{k=1}^{T} d\left(\mathcal{X}\left[\phi_x(k)\right], \mathcal{Y}\left[\phi_y(k)\right]\right) m(k)/M_\phi \tag{4.3}$$

where M_ϕ is a normalization factor, depending on the form of the slope weighting constraint, $m(k)$. The form of $m(k)$ is purely heuristic, expressing the designer's preferences for the local slope of the warped path. This slope weighting constraint can take many forms. The interested reader is encouraged to look at the literature on speech recognition, such as Rabiner and Juang, [61], for further details. The distortion measure, $\mathcal{E}_\phi$ and the warping function, ϕ can be easily computed via dynamic programming, where DTW algorithm is one possible realization.

4.1.2 Batch Time-Normalization

Using the DTW algorithm, the data in the full dataset is transformed to a common length sequence. The heuristic for locating a common sequence to which all data are aligned is based on finding a set of sequences that provide the least distortion within each class (a "mean" prototype), and then picking the longest prototype of the set. The data is transformed as follows:

1. The DTW error matrix is computed for each class of the training data by aligning each pair of sequences in the dataset and computing the corresponding error:

   $$E_{i,j}^k = \mathcal{E}_\phi(\mathbf{s}_i^k, \mathbf{s}_j^k)$$

   where $\mathcal{E}_\phi(\cdot, \cdot)$ designates the DTW error between the sequence $\mathbf{s}_i^k$ warped to the sequence $\mathbf{s}_j^k$, as in eqn. 4.3.

2. Minimum error prototypes are found for each class as sequences producing the least amount of within-class error:

   $$\mathbf{s}^{*,k} = \arg\min_i \left(\sum_j E_{i,j}^k \right)$$

3. The longest prototype is found among the set of k minimum error prototypes:

$$\mathbf{s}^* = \arg\max_k \left\{|\mathbf{s}^{*,k}|\right\}_{k=1}^{K}$$

4. All data in the set are warped to s^* with corresponding warp functions, $\mathcal{P}_\phi(\mathbf{s}_1, \mathbf{s}_2) = \phi_{\mathbf{s}_2}^{-1} \circ \phi_{\mathbf{s}_1}$:

$$\hat{\mathbf{s}}_i^k = \mathcal{P}_\phi(\mathbf{s}_i^k, \mathbf{s}^*)$$

   where $\mathcal{P}_\phi(.)$ designates the warp of the sequence $\mathbf{s}_i^k$ to the sequence $\mathbf{s}^*$ with the pair of the warping functions, computed by DTW algorithm.

5. If $\hat{\mathbf{s}}_i^k$ is a multivariate sequence, it is converted to the vector form:

$$\hat{\mathbf{x}}_i^k = vec(\hat{\mathbf{s}}_i^k)$$

4.2 Support Sequence Machine

The above procedure for the dimensionality normalization is not convenient in practice for use in a developmental setting, where model parameters are updated at every new data sample. Indeed, the sample used as the "master sample" is found in the dataset, which has to be observed in its entirety before the normalization can be made. While incremental learning algorithms for training Support Vector Machines are available, the normalization of the data needs to be performed at every step. A different method, which does not require pre-processing of the data, ought to be found in order to make the support vector technique useful in the incremental setting for sequence classification.

One approach to classification with the use of kernel methods was proposed by Watkins [83]. The author shows that scores produced by some dynamic alignment algorithms form valid kernels. The report proves the validity of the *conditional symmetric independence kernels* for *pair hidden Markov models*. Here another, model-free technique is presented, which is based on dynamic programming.

Gaussian radial basis functions are a popular choice of SVM kernels. Gaussian RBFs satisfy Mercer conditions for representing an inner product in a Hilbert space and are easy to compute from the data in the input space:

$$K(x_i, x_j) = e^{-\zeta \|x_i - x_j\|^2} \tag{4.4}$$

To be useful, the kernel only requires an error norm to be defined between any two data points in the input space. On the other hand, the Dynamic Time Warp technique provides some distortion measure between two sequences, $\mathbf{s}_i$ and $\mathbf{s}_j$. However, in order to induce a metric on the space of sequences, the distance function $\mu(.)$ has to satisfy four requirements:

1. $\mu(\mathbf{s}_i, \mathbf{s}_j) \geqslant 0$

2. $\mu(\mathbf{s}_i, \mathbf{s}_j) = 0$ if and only if $\mathbf{s}_i = \mathbf{s}_j$

3. $\mu(\mathbf{s}_i, \mathbf{s}_j) = \mu(\mathbf{s}_j, \mathbf{s}_i)$

4. $\mu(\mathbf{s}_i, \mathbf{s}_j) + \mu(\mathbf{s}_j, \mathbf{s}_k) \geqslant \mu(\mathbf{s}_i, \mathbf{s}_k)$

The DTW distortion, $\mathcal{E}_\phi$ clearly satisfies condition 1, as it is a linear combination of nonnegative local distortions with non-negative wights, eqn. 4.3. The requirement of condition 2 is also trivially true when $m(k)$ is chosen and no 0-weight coordinate moves are allowed ([61]). The same applies to the triangle inequality, 4.

However, condition 3 is not satisfied by $\mathcal{E}_\phi$. Using the superscript xy to denote the warping transformation from the sequence x to the sequence y, the following symmetrization solves this problem while retaining properties 1, 2 and 4:

Classifier	Error rate	Accuracy	Prototypes
KNN	5.4%	(94.6%)	270
MCC	5.9%	(94.1%)	N/A
HMM	3.8%	(96.2%)	N/A
SVM	1.8%	(98.2%)	58.2
SSM	1.8%	(98.2%)	127.3

Table 4.1: Batch classification results on the Japanese Vowel dataset. SVM with the DynA kernel is shown in the last row of the table.

$$\begin{aligned}\mathcal{D}(\mathbf{s}_i, \mathbf{s}_j) &= \frac{1}{2}\left[\mathcal{E}_\phi(\mathbf{s}_i, \mathbf{s}_j) + \mathcal{E}_\phi(\mathbf{s}_j, \mathbf{s}_i)\right] \\ &= \frac{1}{2}\sum_{k=1}^{T}\Big[d(\mathbf{s}_i(\phi_i(k)), \mathbf{s}_j(\phi_j(k)))m^{ij}(k)/M_\phi^{ij} \\ &+d(\mathbf{s}_j(\phi_j(k)), \mathbf{s}_i(\phi_i(k)))m^{ji}(k)/M_\phi^{ji}\Big] \\ &= \frac{1}{2}\sum_{k=1}^{T} d(\mathbf{s}_i(\phi_i(k)), \mathbf{s}_j(\phi_j(k)))\left[m^{ij}(k)/M_\phi^{ij} + m^{ji}(k)/M_\phi^{ji}\right]\end{aligned} \tag{4.5}$$

Using eqn. 4.5 in place of the error norm in eqn. 4.4 allows us to use the dynamic time warp distortion measure to compute the kernel matrix:

$$K(\mathbf{s}_i, \mathbf{s}_j) = e^{-\zeta \mathcal{D}(\mathbf{s}_i, \mathbf{s}_j)^2} \tag{4.6}$$

Clearly, the dynamic alignemnt kernel of eqn. 4.6 (DynA kernel) satisfies the Mercer conditions to represent an inner product and can trivially be used within the support vector algorithm.

The table 4.2 shows the results of application of the DynA kernel to the set of sequences of 12 LPC coefficients, collected in the UCI Japanese Vowel dataset. The data set consists of 270 training and 370 test samples of 9 speakers pronouncing the Japanese vowel “æ”. The task is to correctly identify the speaker.

The table compares performance of different classification algorithms on the dataset. KNN and SVM are the traditional 1-nearest neighbor and RBF Support Vector machine respectively. These two algorithms are tested on the data that underwent the time normalization, as discussed in section 4.1. That results in the data being mapped to the fixed 250+-dimensional Euclidean space. MCC, HMM and SSM are run on sequences directly. Of these three algorithms, MCC, is the Multidimensional Curve Classification algorithm, proposed by Kudo *et al.* [44], for which the above data set was collected; HMM is a continuous 5-state Hidden Markov Model; and SSM is the support Vector Machine with the Gaussian RBF kernel, using the symmetrized DTW error metric as the basis for the DynA kernel (eqn. 4.6).

The SVM techniques attain the error rate of 1.8%, misclassifying 7 utterances of the test set of 370 with perfect classification of the training set of 270 utterances and outperforming the best reported results achieved with other techniques.

The SVM with the traditional kernel retains a smaller number of support vectors while running on the batch-normalized data while attaining the same performance as the SSM. However, the inherently heuristic technique of data alignment might result in higher distortions of the utterances that are significantly different from the "master" prototype, resulting in a drop in performance, whereas the DynA kernel is only computed based on pairwise relations and does not result in systematic error.

SVM is also at a disadvantage in the situation where no data is seen beforehand, that is in developmental situations, which are the topic of interest to this dissertation.

Chapter 5

EM for Weak Transduction

If a man will begin with certainties, he shall end in doubts; but if he will be content to begin with doubts, he shall end in certainties.

Sir Francis Bacon (1561 - 1626)

5.1 The Goal

The chapter presents an algorithm that implements both perceptual and associative learning with the Expectation Maximization framework. It relieves the constraints on the policy, $p(a|s)$ (see section 2.5 and fig. 2-7 b)) imposed in the previous chapter. This chapter will introduce an additional stage for estimating the mapping from states to actions.

The goal of this chapter is to develop a memory-conscious approach to learning a statistical model of perceptual organization simultaneously with learning associative action selection for applications on longer-range tasks than the technique described in chapter 3.

The parametric statistical model provides less flexibility in the cluster assignments than the memory-based as the cluster models are localized, hence, we can no longer assume a direct mapping between states and actions, that is $p(s|x^n)$ is no longer the same as $p(a|x^n)$.

In addition, the reward is not assumed to be binary and can vary from unspecified negative to unspecified positive values. This takes the problem of learning perceptual organization out of the domain of supervised learning and into the territory of transductive, and of yet uncharted, *weakly transductive* learning, as will be shown in the later sections. From the point of view of reinforcement learning paradigm the problem addressed in this chapter falls into the category of associative learning with simultaneous estimation of the hidden state.

5.2 Overview

Keeping in mind that the goal now is to learn parameterized models of input categories, imagine again the situation where an agent is trained to respond to a set of voice commands. After hearing an utterance the agent performs an action. The trainer would like to train the agent to respond correctly by providing (possibly noisy) rewards and punishments after seeing actions that it performs in response. In this scenario the agent needs to learn two things: a) parameterized equivalence classes in the space of utterances; and b) what action to select upon observing a sample from one of these classes. The first problem is that of *clustering under reward*, while the second is the typical *policy learning* task of reinforcement learning.

There exist a number of algorithms permitting learning of policies of action selection given that the perceptual system provides a good set of features. But how can such features be efficiently estimated while the policy learning is taking place? This chapter focuses on the solution to the problem by embedding a simple associative search algorithm into an Expectation Maximization paradigm.

In addition, this chapter introduces the notion of *weak transduction* in order to properly place the solution among the existing techniques used for unsupervised and transductive problems.

The task of *on-line* estimation of the perceptual organization and the policy of action selection is cast here as a problem of a multi-armed bandit with hidden state and solved iteratively within the Expectation Maximization framework. The hidden state is represented by a parameterized probability distribution over states tied to the reward. The parameterization is formally justified, allowing for smooth blending between likelihood- and reward-based costs.

The chapter proceeds as follows: after introducing the multi-state bandit problem, section 5.4 describes a modification of the reinforcement pursuit algorithm that allows to include the hidden state. Section 5.5 justifies modifications to the EM algorithm that allow to include the reward information into the parameter estimation and to solve the problem of learning perceptual organization along with policy learning. Experiments with this algorithm showing the results for different objectives are presented in section 5.7. The chapter concludes with section 5.8 pointing out contributions and some of the problems with the algorithm.

5.3 Introduction

Let us return for a moment to the original problem statement. The problem, again, is to find a mapping between continuous observations and a set of discrete actions that the agent can execute. In the short run, as the previous chapters show, some of these observations could be simply memorized and then a generalization rule could be invoked to assign an action to a new observation.

One of the main advantages of the memory based-techniques of the previous chapters is their insensitivity to the distribution of the data within each category. In such a formulation, each cluster can potentially contain a set of disjoint clusters, collected within one model. This allows us to keep the policy matrix square and diagonal, since in these circumstances the many-to-one mapping from states to actions is often unnecessary. The drawback of the technique is that it is largely memory and computation bound.

5.3.1 Long Term Perceptual Learning

In the long run, memorization of the prototypes does not lead to a satisfying solution. Indeed, if the number of observations that the algorithm is allowed to keep is not bounded, then very soon the problem becomes intractable, as the computational complexity of it is typically quadratic in the number of prototypes. In addition, the memory requirements make it in efficient to store even small subsets of the data. In contrast, it is often beneficial to learn a distribution-based model of the input space when the amount of data allows, as distribution-based methods typically perform better than the distribution-free.

For a longer term learning task one would like to seek a parametric solution that allows discarding the data altogether, while keeping only their statistics. Now, estimating the parametric statistics necessitates using far simpler models for the data (e.g. Gaussians, etc.) than the data itself. Now one needs to trade the simplicity of the input representation with the necessity of learning to associate these simple unimodal categories with actions, building up multimodal perceptual models which were essentially available for free with memory-based techniques. In other words, in a statistical setting, part of the complexity is now shifted to the policy, such that $p(s|x^n)$ is no longer the same as $p(a|x^n)$.

Associative Search

The policy in the problem addressed here has a special and simple character - since all observations are assumed to be equally likely and independent - the trainer can produce any utterance or gesture at will - there is no need to keep the history as a part of the action context. In other words, the agent's behavior does not change the state of the world, which only changes at random. The task of the policy estimation in such a simplified setting is termed *associative search* and requires no historical data to be solved. However, it is complicated by the necessity of perceptual learning, or state estimation.

State Estimation

The difficulty with policy evaluation comes from the fact that the action selection, performed by sampling the distribution over actions, $p(a|x^n)$ is no longer the same as the distribution over states, $p(s|x^n)$. In order to construct $p(a|x^n)$ one needs the entire distribution $p(s|x^n)$, as will be shown in subsequent sections, and not just its value at a particular state. In this setup one can never commit to a state and the states need to be estimated without the direct knowledge about exact state attribution of the observation x^n. At worst - this is a problem of unsupervised clustering. However, if an action is rewarded, the fact that selecting it based on a particular value of $p(s|x^n)$ was beneficial provides additional information. Thus, the problem of state estimation is weakly supervised and one needs to solve clustering under reward.

5.3.2 Related Work

This chapter is based on the Expectation Maximization algorithm [16, 62, 89] extended to situate it within the context of reinforcement learning and to take advantage of the additional information that is available as a result of interaction with the environment. Neal and Hinton [54], show a view of the EM algorithm that makes the extensions made in this chapter possible.

At a certain parameter setting and binary reward the algorthm shown here can be viewed as an on-line version of the λEM, presented by Nigam et al. [55], for learning with partially labeled data (albeit for a Gaussian Mixture) and transductive inference [80]. However, the problem solved by the algorithm described here is in general more difficult than the problem of transductive clustering, which Nigam's algorithm solves as it does not have access to exact labels for the input data.

To learn the policy of the action selection the learning algorithm developed here uses an N-armed bandit model (see for example Sutton and Barto, [74]). The multi-state policy is estimated with the aid of the reinforcement pursuit algorithm of Thatachar [76], which is applied to a set of states simultaneously.

A problem similar to the one presented here was explored by Likas [45], who used a variant of the REINFORCE algorithm [85] to learn Vector Quantization on the batch data, aided by a reward signal.

The technique of probability matching for reinforcement learning used here is similar to that shown by Sabes and Jordan [66]. Using this technique, the algorithm presented here constructs a reward-dependent probability distribution to guide the algorithm towards the configuration resulting in higher value of the expected reward.

5.4 Estimation of the Associative Policy

The problem of agent training, as described earlier, falls into the category of *associative learning with hidden state*. If the state space is modeled with a mixture distribution, then the problem can be described as follows: given an observation, estimate the state of the world from a finite set of states, $S = \{s_i\}$. *Given the belief* about the state membership, select an action (label), which will result in the maximum amount of expected payoff received once the action is performed. With that payoff, update the parameters of the policy of the action selection and of the input distribution. This section will deal with solving the problem of policy estimation for such a setup.

5.4.1 Multi-State Bandit Problem

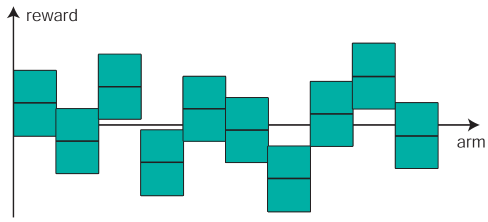


Figure 5-1: 10-armed bandit model. Each of the 10 arms produces a reward by drawing a sample from a corresponding distribution. Each box signifies the reward distribution with some mean (horizontal bar) and variance (height of the box).

Due to the previously stated assumption of the observations being independent of each other, this problem can be thought of as a multi-state N-armed bandit [74]. The N-armed bandit is a gambling device with a set of N arms (see fig. 5-1). Each arm has a probability distribution associated with it, according to which the sample reward is drawn every time the arm is pulled. Most frequently the reward generating process is assumed to be stationary or, at most, slowly varying.

Now imagine that the bandit can be in any one of M states, each of which have different distributions of the reward. Before each trial the bandit switches to a new state and produces an observation, x^n, from the distribution associated with this state (see fig. 5-2). The player's goal is to identify this state and perform action selection and model update for that state. When the state is perfectly known the problem reduces to M independent N-armed bandits. It is more difficult when the state is

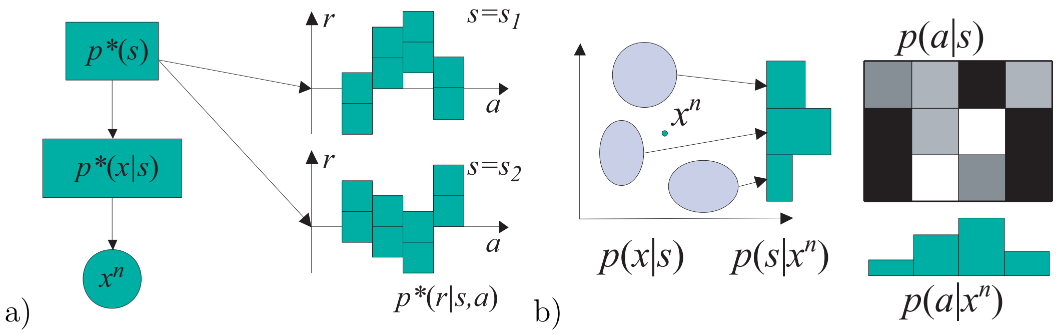


Figure 5-2: a) A generative model of the environment, shown as a 2-state 4-armed bandit. The bandit randomly switches between two states, according to a sample drawn from $p(s)$. After selecting the state, s, an observation, x^n is produced from a distribution $p(x|s)$. b) The estimator consists of two parts - a perceptual model and a policy. Upon receiving the observation, x^n, the distribution $p(a|x^n)$ is constructed and an action is selected by drawing a sample from it. Upon delivery of the reward parameters of both the policy and the perceptual model are updated.

hidden and must be estimated.

5.4.2 Solutions with Known State

When the state is exactly known, then the solution for the multi-state bandit is achieved by independently solving a set of single-state bandits. A variety of *action-value* methods, such as *sample average*, *reinforcement comparison* and *reinforcement pursuit*, have been proposed to solve the single-state bandit problem[1]. The general idea is to stochastically search the action space while updating the estimate of the reward function. A probability distribution over the action space (*action preference*) is built based on this estimate and action selection is done via sampling from this distribution.

The simplest pursuit method, adapted for the multi-state agent, maintains an estimate of the payoff structure of the bandit via action value function, $Q_t(a, s)$. This function is updated at each step based on the reward received from the bandit after pulling the arm a by, for example, an exponentially-weighted sample-average method:

[1]Another classical method for solving bandit problems, which includes balancing of exploration with exploitation involves computation of the so called Gittins indices. This method provides an optimal solution to a large class of problems, but assumes the knowledge of prior distribution of possible problems.

$$Q_t(a, s) = Q_{t-1}(a, s) + \alpha(r - Q_{t-1}(a, s)) \tag{5.1}$$

Based on the value of $Q_t(a, s)$, the pursuit method updates its action preference model, $\hat{p}_t(a|s)$, such that the action with the highest value of $Q_t(a, s)$ increases the probability of being selected by a small fraction, γ. Actions that are currently found to be suboptimal decrease their probability correspondingly. Let $a^*_{t+1} = \arg\max_a(Q_t(a, s))$, then:

$$\begin{aligned} \hat{p}_{t+1}(a^*|s) &= \hat{p}_t(a^*|s) + \gamma(1 - \hat{p}_t(a^*|s)) \\ \hat{p}_{t+1}(a|s) &= \hat{p}_t(a|s) + \gamma(0 - \hat{p}_t(a|s)), \ \ \forall a \neq a^* \end{aligned} \tag{5.2}$$

The convergence of the pursuit method is dependent upon values of α and γ, which in all experiments of this chapter are set to be $\alpha = 0.1$ and $\gamma = 0.01$. In addition, it is readily combined with ε-greedy techniques to allow for non-stationary environments.

5.4.3 Solutions with Hidden State

In the presence of the hidden state the problem of estimating the optimal action becomes more difficult. The uncertainty about the state can be dealt with by distributing the reward proportionally to the current belief about the state membership of the observation x^n.

Most of the bandit search algorithms allow for formulating a policy, or a probability distribution over actions, given a state, $p(a|s)$. This is an arbitrary distribution which only expresses the current estimate of "action preferences". The action is selected by sampling the conditional probability distribution $p(a|x^n)$, which can be calculated from the belief state and the policy, by marginalizing the joint, $p(a, s|x^n)$:

$$
\begin{aligned}
p(a|x^n) &= \sum_s p(a,s|x^n) \\
&= \sum_s p(a|s,x^n)p(s|x^n) \\
&= \sum_s p(a|s)p(s|x^n)
\end{aligned} \tag{5.3}
$$

The action selection now takes into account the uncertainty about the state, encoded in the state posterior. For the purpose of bandit updates, the reward is distributed among M bandits in proportion to their contribution to $p(a|x^n)$:

$$
Q_t(a,s) = Q_{t-1}(a,s) + \alpha(rp(s|x^n) - Q_{t-1}(a,s)) \tag{5.4}
$$

The action preference update equations, eqn. 5.2 are left unchanged.

5.5 Clustering Under Reward

State estimation under reward can be performed with the aid of the Expectation Maximization algorithm, which is often used for unsupervised clustering. This section introduces a technique for including the reward function into the EM re-estimation procedure. The new objective function is simply implemented in the EM framework while allowing the algorithm to "fall back" to the unsupervised mode if no reward is provided.

5.5.1 Weak Transduction

The EM algorithm is a powerful tool for solving unsupervised and transductive problems. It is often used as a clustering algorithm with the objective of maximizing the likelihood of the data. This is a good heuristic to use for learning perceptual organization when no other evidence is available. However, by using an *unsupervised* technique for learning the perceptual organization, one disregards its utility for the agent.

The utility of a perceptual configuration is measured by the reward that the agent

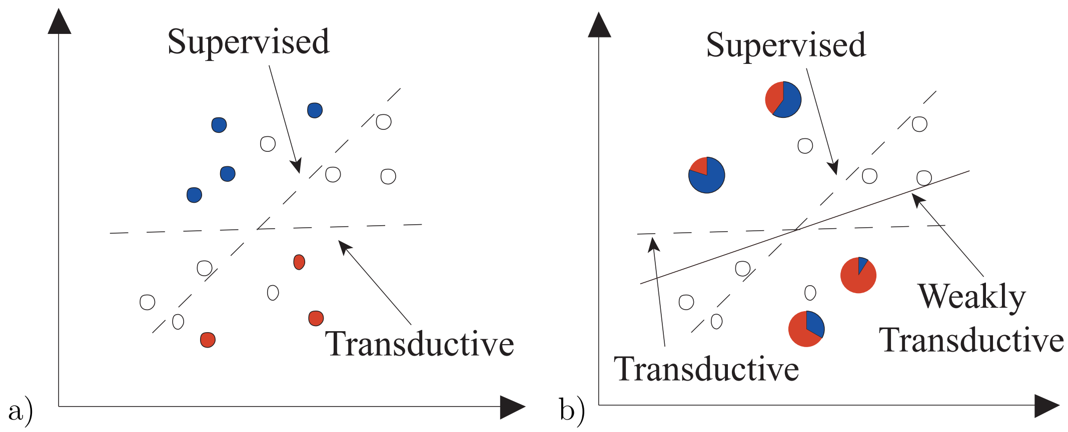


Figure 5-3: a) Illustration of transductive inference for classification. Empty circles indicate the data with unknown class membership. Classification boundaries are different for estimators with and without unlabeled data. b) Weak transduction has no direct indication of the class label, but a probability distribution over labels.

collects while using it. Therefore, an algorithm is sought, which while capable of learning from patterns in the input data alone, can be "directed" with the reward to choose a different configuration providing higher payoff. That is, the solution should be an EM-type algorithm, which would allow the inclusion of reward into its objective for state estimation, while learning the policy of action selection.

The Expectation Maximization algorithm is frequently used for *unsupervised* clustering of data by spatial proximity in the space of features. For a given number of clusters the algorithm proceeds iteratively to first, calculate from the current cluster statistics the probability of data to be generated by each cluster, a state posterior, $p(s|x)$; and then to average the data, weighted by this posterior to update cluster statistics.

When the data comes with the known state attribution, $s^n = z$, then the posterior of each data point turns into a deterministic function, having 1 at the slot of the corresponding state and 0 elsewhere:

$$\begin{aligned} p(s^n = z|x^n) &= 1 \\ p(s^n \neq z|x^n) &= 0 \end{aligned} \tag{5.5}$$

Averaging with respect to this posterior, let us call it $p_{01}(s|x)$, results in the parameter estimation to decompose into several independent *supervised* estimation problems.

When only part of the data has an exact label, $s^n = z$, then the solution to the clustering problem results in a mixture of the supervised solution for the labeled data and the unsupervised solution for the unlabeled set. This is an example of *transduction*, where the knowledge from the labeled data is transduced onto the unlabeled. The setting of a transductive classification is illustrated in the figure 5-3 a). In the figure the supervised solution does not follow the valley in the density of the complete data set, while transductive inference takes into account all available data.

Supervised, unsupervised and transductive learning methods view the label information in a binary fashion - it is either present or absent. In contrast, under the circumstances of the problem where the knowledge about the label is inexact and subjective, the situation is a bit worse than in the transductive setting, but better than unsupervised. With the current setting of the model parameters the posterior $p(s|x^n)$ is computed as the state membership of the query point. If, consequently, some value of reward results from this assignment, it indicates the quality of the posterior given the current parameter settings. This is the situation, which the algorithm being described encounters in the task of estimating a state. That is, in line with the above taxonomy, the data is labeled with a *probability distribution*, as illustrated in figure 5-3 b). It is convenient to call data labeled in this fashion *weakly labeled*, and the problem - a *Weak Transduction*. These terms properly place the problem among traditional machine learning tasks and emphasizes its relation to already existing techniques for learning with labeled, unlabeled and partially labeled data (e.g. [55]).

5.5.2 Reward-driven variational bound

Typically, the EM algorithm for density estimation is used for unsupervised maximization of the likelihood function of a parametric density model when obtaining an analytical solution for the gradient in the parameter space is difficult. This is the case when we need to learn parameters of a mixture density. In the algorithm of this chapter the input space is represented by a mixture density, $p(x; \theta) = \sum_i p(s_i)p(x|s_i; \theta_i)$, parameters of which, θ, need to be estimated. The goal of the algorithm, however, is to not simply maximize the likelihood of the data, but also take into account the external reward signal if such is present. To do so, in this section a new cost function is justified, which allows for inclusion of the reward in the traditional EM framework.

EM as a variational bound optimization. The main idea of EM is based on simple geometric reasoning - if instead of maximizing some difficult function one maximizes its convex lower bound that touches the function at a current parameter value, then a step in the direction of the gradient of this bound is also a step in the direction of the local maximum of the function. For a likelihood function, $f(\theta) = p(x, \theta)$, where x is the data set and θ is the vector of parameters, the EM algorithm is based on the following bound (geometric-arithmetic mean inequality):

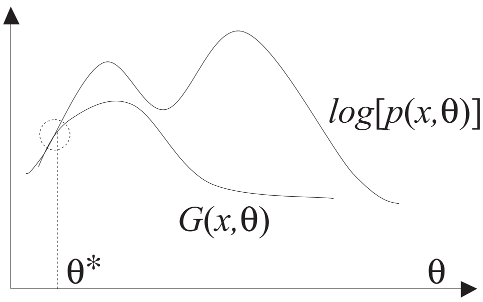


Figure 5-4: The step in the direction of the gradient of the lower bound bound of eqn. 5.8 is the step in the direction of the gradient of likelihood.

$$
\begin{aligned}
f(\theta) &= \int p(x, s, \theta) \frac{q(s)}{q(s)} ds \\
&\geqslant \prod_s \left(\frac{p(x, s, \theta)}{q(s)} \right)^{q(s)} = g(x, \theta)
\end{aligned}
\tag{5.6}
$$

Here, $g(x, \theta)$ is a lower bound of the likelihood, $f(\theta)$, and $q(s)$ is some positive function of s, integrating to 1. Typically, for the purposes of optimization of $f(\theta)$, the logarithm of $g(x, \theta)$ is optimized:

$$
G(x, \theta) = \int q(s) \log p(x, s, \theta) - q(s) \log q(s) ds \tag{5.7}
$$

It follows from eqn. (5.6) that for any density $q(s)$, $G(x, \theta)$ is a lower bound on $\log f(\theta)$. Now the density $q(s)$ needs to be found, which touches $\log f(\theta)$ at θ. The cost function in eqn. (5.7) can be re-written as follows [54]:

$$
G(x, \theta) = -D\left(q(s) || p(s|x, \theta)\right) + \log f(\theta) \tag{5.8}
$$

where $D(p||q)$ is a Kullback-Leibler Divergence between distributions p and q. From here it is easily shown that $G(x, \theta) = \log f(\theta)$ when $q(s) = p(s|x, \theta)$, that is, the bound will be touching the likelihood function at the current θ, as shown in figure 5-4.

Augmented reward bound. In order to let EM include the expected reward into the optimization, the EM bound shown above needs to be augmented with a reward-dependent term. It is easy to do using the probability matching technique [66].

To learn preferred cluster configurations, one can consider observation-state pairs and construct a reward-dependent probability distribution, $p^*(s|x; r)$. The task of the learning algorithm is to select from a set of conditional distributions $p(\mathcal{S}|\mathcal{X}, \theta)$, aided by rewards that are provided by the environment for some of the data points. These rewards can be thought of as inverse energies - pairs (s, x) receiving higher rewards correspond to lower energy states. Energies can be converted to probabilities via the Boltzmann distribution, which represents the ideal observation-state mapping - (s, x) pairs receiving higher rewards being more likely than pairs receiving low reward. If the parameters of $p(s|x, \theta)$ are adjusted so that it is close to $p^*(s|x; r)$, then the output of the algorithm will typically result in higher rewards.

Following this line of reasoning $p^*(s|x; r)$ is made *proportional* to the Boltzmann distribution as shown later in the text. Starting with the equation (5.8), an additional term penalizes the estimator for being different from this distribution in the posterior:

$$\begin{aligned} F(x, \theta) = & -D\left(q(s)||p(s|x, \theta)\right) + \\ & E_{q(s)}\left[\log \frac{p^*(s|x; r)}{p(s|x, \theta)}\right] + \log f(\theta) \end{aligned} \tag{5.9}$$

When $q(s)$ is set to the posterior distribution, $p(s|x, \theta)$, the expectation term turns into negative divergence between the posterior and, $p^*(s|x; r)$:

$$E_{q(s)}\left[\log\frac{p^*(s|x;r)}{p(s|x,\theta)}\right]\Bigg|_{q(s)=p(s|x,\theta)} = \\ -D(p(s|x,\theta)||p^*(s|x;r)) \tag{5.10}$$

In fact this term induces a different but very intuitive bound for the likelihood maximization, which is shown in the theorem 5.5.1.

Theorem 5.5.1. *$F(x,\theta)$ is a lower bound on* $\log f(\theta)$.

Proof. Starting from (5.9), one can write:

$$\begin{aligned}
F(x,\theta) = &-D\left(q(s)||p(s|x,\theta)\right) \\
&+ E_{q(s)}\left[\log\frac{p^*(s|x;r)}{p(s|x,\theta)}\right] + \log f(\theta) = \\
\int &q(s)\log\frac{p(s|x,\theta)}{q(s)}ds \\
&+ \int q(s)\log\frac{p^*(s|x;r)}{p(s|x,\theta)}ds + \log f(\theta) = \\
\int &q(s)\left[\log\frac{p(s|x,\theta)}{q(s)} + \log\frac{p^*(s|x;r)}{p(s|x,\theta)}\right]ds + \log f(\theta) = \\
\int &q(s)\log\frac{p^*(s|x;r)}{q(s)}ds + \log f(\theta) = \\
-&D\left(q(s)||p^*(s|x;r)\right) + \log f(\theta)
\end{aligned} \tag{5.11}$$

In the last line of eqn. (5.11) the divergence, $D(q(s)||p^*(s|x;r)) \geqslant 0$, from which follows that

$$F(x,\theta) \leqslant \log f(\theta), \qquad \forall q(s),\theta, s.t. \sum(q(s)) = 1 \tag{5.12}$$

with equality holding iff $q(s) = p^*(s|x;r)$. □

This function has the same form as eqn. (5.8), which implies that for practical purposes one may simply substitute the EM-induced posterior with the fictitious probability distribution, $p^*(s|x;r)$. It provides the traditional bound for the likelihood function in the absence of the reward. With the reward present, the algorithm performs only a *partial E-step*. However, the step in the direction of the gradient of

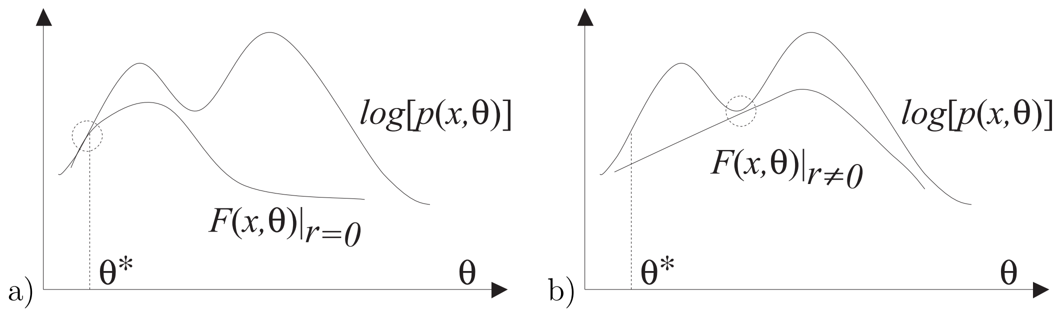


Figure 5-5: a) The augmented bound behaves just like the traditional EM bound when no reward is present. b) With the reward present the bound is no longer in contact with the likelihood at the current parameter setting, leading uphill in the expected reward.

this bound leads uphill in the future expected reward.

Now $p^*(s|x;r)$ needs to be constructed in a convenient form. The main constraint that should be imposed is that the additional term in eqn. (5.9) vanishes when after producing a label s for an observation x, the reward r received from the environment is 0. That is,

$$E_{q(s)}\left[\log \frac{p^*_{r=0}(s|x;r)}{p(s|x,\theta)}\right] = 0 \tag{5.13}$$

which implies that $p^*_{r=0}(s|x;r) = p(s|x,\theta)$. The distribution $p^*_{r=0}(s|x;r)$ can be set to be proportional to the Boltzmann distribution:

$$p^*(s|x;r) = \frac{p(s|x,\theta)\exp\left(\beta r p(s|x,\theta)\right)}{Z_\beta(x,\theta)} \tag{5.14}$$

This form of $p^*(s|x;r)$ is used throughout the rest of this chapter.

The resulting bound is illustrated in figure 5-5. The augmented bound behaves just like the traditional EM bound when no reward is present. With the reward present, the bound is no longer in contact with the likelihood at the current parameter setting, leading uphill in the expected reward. The point of contact with the bound is the value of parameter at which the posterior $p(s|x^n)$ equals $p^*(s|x;r)$.

5.6 Reward-Driven Expectation Maximization

Now the two parts of the estimation procedure can be joined to get the complete solution to perceptual learning under reward. The algorithm is shown below and is illustrated in the figure 5-6.

The algorithm folds the action-selection policy estimation into the Expectation step of the EM algorithm while using the immediate reward signal to control the entropy of the posterior for the Maximization step. The algorithm is iterative and incremental, performing one iteration per data point, keeping only the sufficient statistics about the density function of the input space. The goal of the algorithm is to estimate the structure shown in the figure 5-2. It proceeds as follows:

REM algorithm

1. **Initialize:** Set parameters of the M-state Bandit model to starting values; guess initial parameters of the distribution $p(x)$ and iterate the following Expectation and Maximization steps; for each new data point:

2. **E-step:**

   (a) calculate $p(s|x^n)$ using the Bayes rule and the current parameters of the observation model, $p(x)$;

   (b) **Forward pass:**

      i. compute $p(a|x^n)$ (eqn. 5.3);

      ii. select an arm by sampling $p(a|x^n)$

   (c) **Backward pass:**

      i. collect reward and distribute it among the states in fractions of $p(s|x^n)$;

      ii. calculate $p^*(s|x^n, r^n)$ (eqn. (5.14));

3. **M-step:** Maximize the resulting bound, eqn. (5.11), with respect to parameters, θ.

In the forward pass of the algorithm the processing breaks out of the EM's Expectation step to select an action and update the Bandit model as shown in the figure 5-6. The yielded payoff serves as a control parameter for the EM.

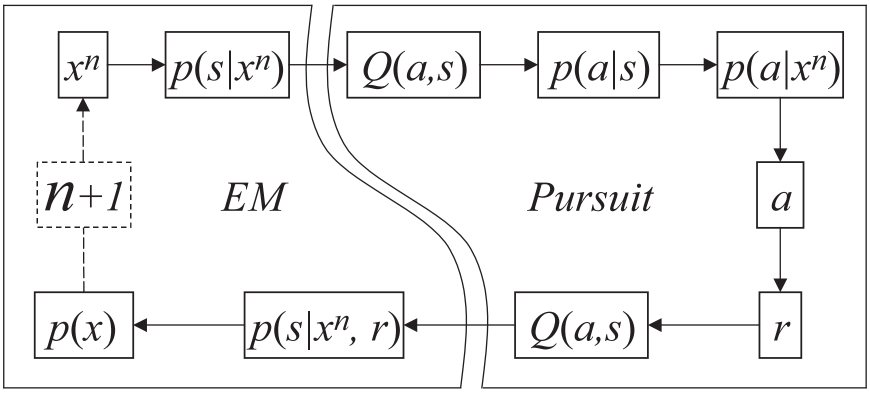


Figure 5-6: The reward-driven perceptual learning algorithm breaks out of the expectation step of EM to compute the improved posterior. Then the parameter estimation is performed with respect to $p(s|x^n, r)$

5.7 Experiments

The experimental analysis of the algorithm presented in this chapter is performed on a series of tasks of increased difficulty. The first experiment does not include policy learning and is designed to simply test estimation of the perceptual model alone for a fixed optimal policy. Next, two experiments are performed, which involve the policy estimation. In the first experiment the reward is delivered by a bandit with only one arm per state producing a unit of reward. In the second experiment the binary restriction on the bandit is removed allowing each arm to produce some value of reward, positive or negative. Finally, an experiment is performed with a variation on the reward structure such that the reward reinforces arbitrary objective, not related to the likelihood of the data.

5.7.1 EM for state estimation

The first experiment confirms the conclusions of the previous section, showing that it is in fact possible to use the EM framework for partially supervised tasks. It has to be shown that, given the context of the classification task, the algorithm will result in choosing the clustering configuration that provides a higher expected reward.

In the experiments of this section, the performance of the algorithm is compared with the traditional EM. However, it should be understood that this comparison is for reference only, as the EM is not designed to perform the task that REM is targeting and can only provide the "worst case" performance.

As a source of the data a Gaussian mixture, $q(x) = \sum q(s)q(x|s)$ is used. The algorithm estimates the density $p(x) = \sum p(s)p(x|s)$ by adjusting its parameters in an on-line fashion, upon seeing every data point, x^n. The reward is delivered after an attempt to classify x^n to be generated by a particular component of $p(x|s_i)$. The experiment proceeds as follows:

1. **Initialize the generator mixture,** $q(x)$**:** `for each state,` s_i`, randomly select a Gaussian observation model -` $\mu_i \sim N(\mathbf{0}, 2I)$ `and` $\sigma_i = I$`;`
2. **Iterate:**
   (a) `randomly choose a generator state,` s_k`;`
   (b) `generate an observation,` x^n`, distributed with` μ_k `and` σ_k`;`
   (c) `using current parameters of the model,` $p(x)$`, select a label` l^n`;`
   (d) `if` $l^n = s_k$`, deliver a reward of 1, otherwise,` -1`;`
   (e) `update parameters of the model`
      i. `compute` $p^*(s|x^n; \hat{r})$ `via eqn. (5.14);`
      ii. `perform the E-step of the EM algorithm using` $p^*(s|x^n; \hat{r})$ `in place of` $p(s|x^n)$`.`

The results of the incremental reinforced binary classification experiments are shown in the figure 5-7. The top plot shows the attained likelihood of the data after a number of randomly generated samples. The horizontal axis shows the number of iterations (data points seen so far) with the likelihood plotted along the vertical axis. It is curious to see that the unguided EM (with $\beta = 0$) attains the lowest likelihood. This is partially due to the fact that the EM is more likely to get stuck in the local maxima, while the reward signal delivers some extra energy for the algorithm to get out of it.

The intuition behind choosing the parameter β is that as it increases, the entropy of the probability distribution from which a label is selected drops. Characteristic

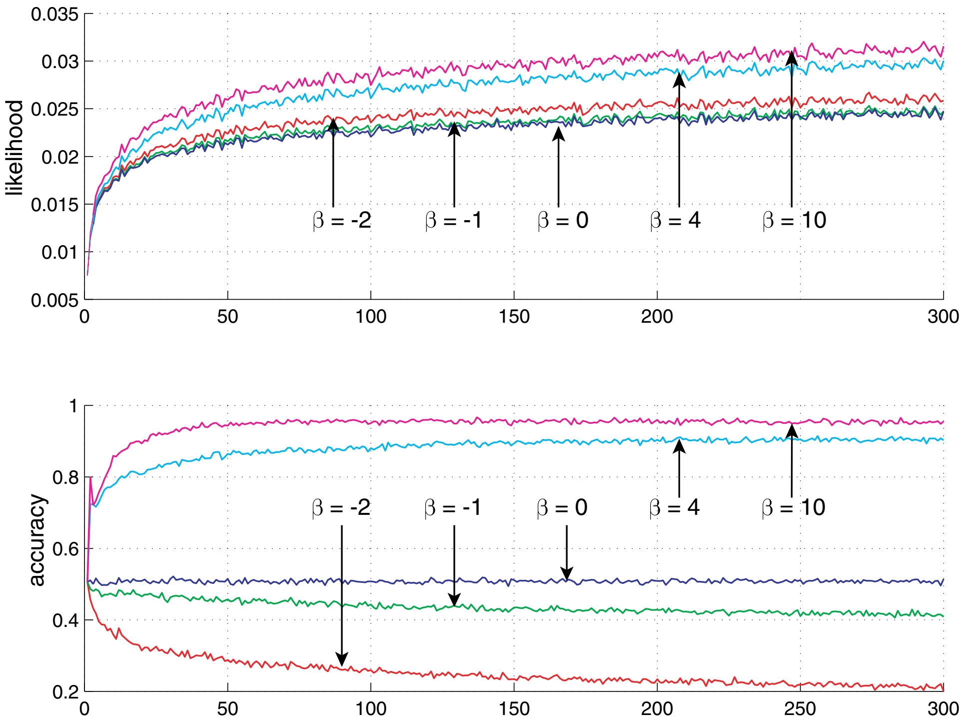


Figure 5-7: Performance of the REM averaged over 1000 runs for different values of the parameter β as compared with EM. Curiously, even negative values of β result in higher likelihood than that attained by EM.

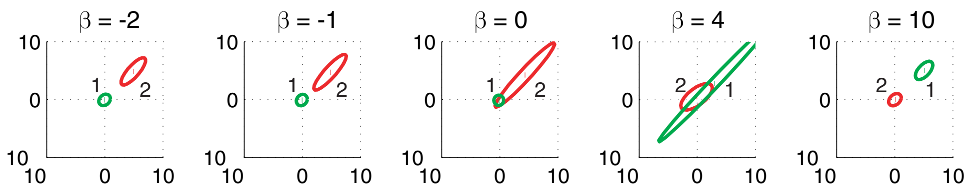


Figure 5-8: Results of a run of the algorithm for different values of β starting from the same initial conditions. For coefficients with opposite signs the labeling is reversed, while the EM produces the labeling by chance. In this run the source distribution has the component 1 (green) at the position $(5, 5)$ and component 2 (red) at $(0, 0)$.

behavior of the algorithm can be observed at extreme values of β: with $\beta = -\infty$, the distribution over labels is uniform and the label selection is performed purely by chance, with no regard to neither the reward nor mixture parameters. At $\beta = 0$ the distribution over labels exactly equals to the mixture posterior, that is the algorithm disregards the reward completely, performing the unsupervised parameter estimation as mixture parameters dictate. Setting β to $+\infty$ results in a "winner-take-all" label assignment.

The second plot in figure 5-7 complements the likelihood plot by showing the classification accuracy of the algorithm at different values of the parameter β. It is expected that the accuracy of the EM used for classification should not be better than chance, since even when EM converges to the correct set of classes it does not care which source cluster corresponds to which estimated component. Positive values of the parameter β drive the extended EM towards correct labeling, while negative β drives the algorithm away from it, as can be seen in the accuracy plot. It is interesting that none of the settings of the parameter β result in the optimal accuracy of 1. There are two reasons for this. First, any fixed value of β less than ∞ will result in sub-optimal label to be selected, albeit with small probability. The second reason is related to the fact that even optimal Bayes classifier will not achieve the perfect classification rate as randomly placed source Gaussian components may significantly overlap.

The influence of β is further illustrated in the figure 5-8. The figure shows the resulting clustering attained with different values of β. It can be seen that the clusters for positive and negative values of β have opposite labeling while zero-valued β is labeled by chance. In this run the source distribution has the component 1 (green) at the position $(5, 5)$ and component 2 (red) at $(0, 0)$, which is correctly identified by the algorithm with large positive value of β.

5.7.2 Multi-State Bandit with Hidden State

In contrast to the previous experiment a policy estimation is now introduced. The estimation of the perceptual state has to be performed on the basis of *indirect* reward attribution, that is, the state now becomes hidden.

Maximization of the likelihood - Binary Bandit

This section shows the results on problems in which the reward function is well aligned with the likelihood, that is, the problems where maximization of the reward results in maximization of the likelihood. Results for this task are shown in figure 5-9. Unlike in the experiments of the previous section, the cluster identity is not important, as long as they correctly partition the input space. The multi-state Bandit essentially implements the mapping from clusters to labels.

It is particularly interesting to see if the reward-based estimator of the input density results in a better fit of the resulting observation density to the one that gets reinforced than the regular EM. In the case of a Gaussian mixture density with a known number of components (known number of states), the fit can be measured with the symmetrized KL Divergence (see appendix B):

$$\begin{aligned} S(p||q) = \frac{1}{4} \Big[& (\mu_q - \mu_p)^T (\mathbf{\Sigma}_q^{-1} + \mathbf{\Sigma}_p^{-1})(\mu_q - \mu_p) \\ & -tr \left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p + \mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q - 2\mathbf{I} \right) \Big] \end{aligned} \tag{5.15}$$

For a lack of a better analytical method, this quantity is computed for every combination of source and estimator components and the minimum value is selected.

The experiment with a 2-state 10-arm Bandit is performed as follows:

1. **Initialize:** `for each state, randomly select a Gaussian observation model:` $\mu_i \sim N(\mathbf{0}, 2I)$, $\sigma_i = I$;
2. **Iterate:**
   (a) `randomly choose a generator state,` s_k;
   (b) `generate an observation,` x^n, `from` $\mathcal{N}(\mu_k, \sigma_k)$;
   (c) `using current parameters select an action` a^n;
   (d) `if` a^n `is the same as the optimal arm deliver a reward of` 1, `otherwise` -1;
   (e) `update parameters of the model;`

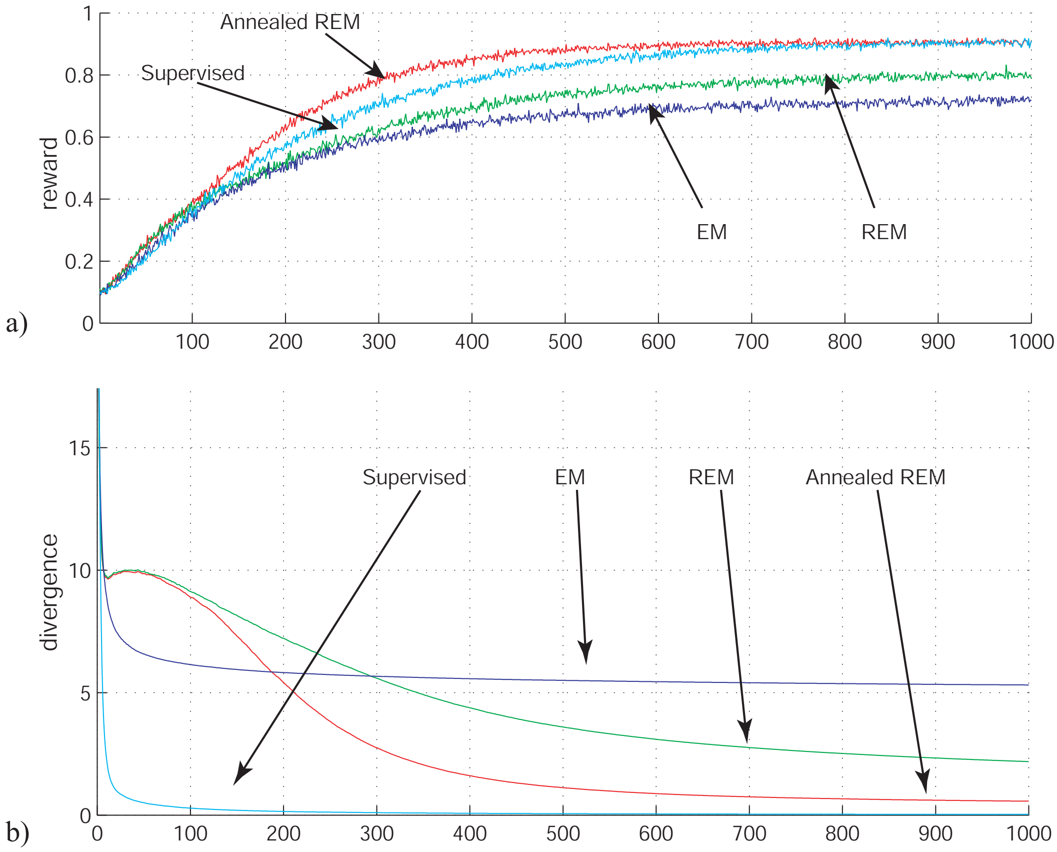


Figure 5-9: a) Performance on the 2-State 10-Armed binary bandit. b) Divergence between estimated and true source distributions.

One variation on the algorithm described in this chapter is the REM with the parameter β changing over time. For example, slowly increasing β, starting with the value of 0 will result in the algorithm to not pay any attention to the reward initially, while slowly shifting towards the "winner-take-all" mode after some period of time. Let us call it annealed REM.

The figure 5-9a) shows the average amount of reward collected by Bandits trained with the EM, REM and annealed REM algorithms compared to the case where the input space is estimated via a supervised estimator. As the goal is an accurate reproduction of the source mixture, these plots need to be considered along with the divergence plots (eqn. 5.15), given in figure 5-9b). The annealed REM algorithm, which slowly increases the value of the parameter β, performs very well, converging even faster than the supervised case. It is somewhat puzzling, but easily explained by the fact that the annealing amounts to simultaneous exploration of all states of the

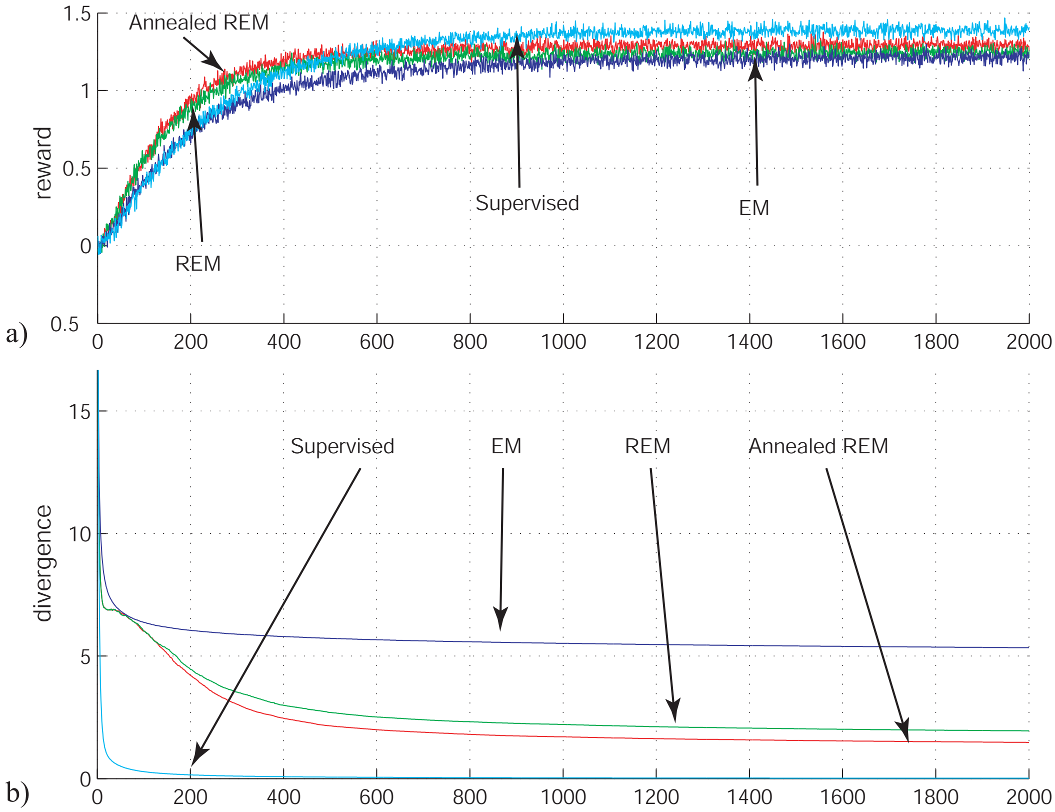


Figure 5-10: a) Performance on the full 2-State 10-Armed bandit. b) Divergence between estimated and true source distributions.

bandit in the initial stages. This gives a good set of initial conditions for subsequent search in each bandit when β increases.

Maximization of the likelihood - Full Bandit

The algorithm works with the full bandit, where each action taken by the algorithm results in some value of the reward – positive or negative, with no modifications. The results are shown in the figure 5-10a). As in the case with the binary bandit, the initial convergence of both REM and Annealed REM is faster than the supervised case. The advantage, compared to EM, however, seems less spectacular than in the binary case. The divergence plots (figure 5-10b)), as before, show better fit of REM and Annealed REM to the source distribution.

This experiment shows the worst case scenario for the algorithm. The reward structure here has many local maxima and is "distracting" for the on-line search. The

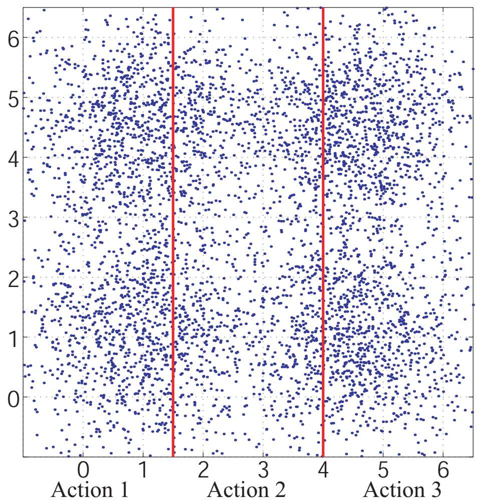


Figure 5-11: Source and the reward structure for the reward bound maximization task. The data forms four strong clusters, while the reward is delivered for selecting action 1 if the data comes from the area marked "Action 1", etc.

search becomes more difficult and the limitations of the search algorithm become the deciding factor in the achieved performance. However, despite the inconsistencies in the reward, the perceptual system captures the input distribution better when aided by the reward than when no feedback is given.

Maximization of the reward

It is interesting to see how this model performs on a problem in which the reward function is not aligned with the likelihood. The problem in this section is as follows - the input data is generated from 4 2-dimensional Gaussians. However the reward is delivered in such a way that action a_1 is rewarded when $x_1^n < 1.5$, a_2 - when $1.5 \leqslant x_1 < 4$ and a_3 when $x_1 > 4$, as shown in figure 5-11.

The performance of the model on this task is shown in the figure 5-12. After 2000 iterations the EM estimator yields an average reward of 0.58, Annealed REM - 0.82 and supervised estimator - 0.96 with the maximum possible reward of 1.

Figure 5-13 shows results of a single run of the algorithm. The left column of the figure shows the resulting positions and outlines of the mixture components. The middle column shows the classification decision regions corresponding to the clustering shown on the left. The right column shows the "cluster assignment" -

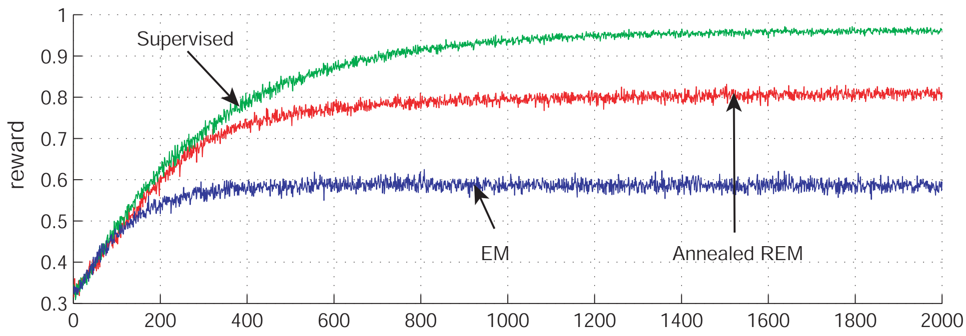


Figure 5-12: Performance of EM, REM and a fully supervised estimator on the problem where reward structure does not coincide with the likelihood (averaged over 2000 runs).

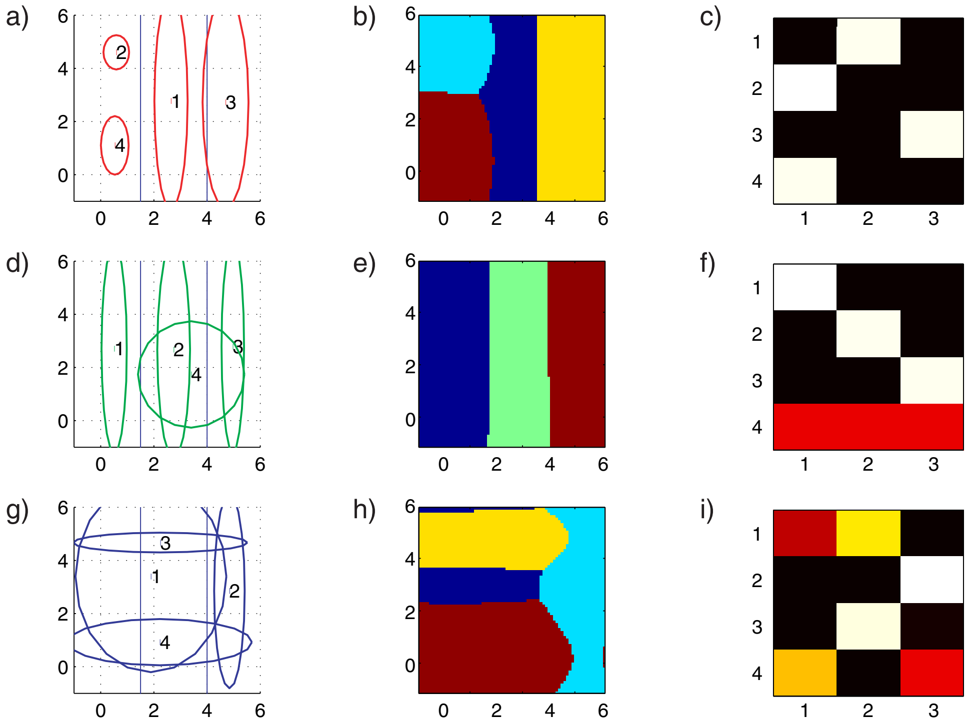


Figure 5-13: Final cluster positions (left column), decision regions (middle column) and cluster assignment matrices (right column) for REM (top row), supervised (middle row) and EM (bottom row) estimators after a single run.

matrices that map states to actions, $p(a|s)$. A value in k-th position of l-th row of the matrix indicates the probability of selecting an action k once the point x^n is classified as belonging to the cluster l. Figures (a–c) demonstrate the performance of the annealed REM algorithm, (d–f) - that of the supervised model, and the bottom row (g–i) - the performance of the unguided EM. The supervised case gives the best possible partitioning of the input while using 3 Gaussians (component 4 is never used and therefore has a mixing coefficient 0). The REM uses all 4 components and aligns them with the reward partitioning. Note that both clusters 2 and 4 select action a_1.

5.8 Discussion

This chapter presented an extension to the EM algorithm that allows for solving a range of learning tasks - from fully unsupervised, to fully supervised, including the partially and weakly labeled data. The justification for entropic variations of the posterior to achieve arbitrary component assignment goals is provided in the text. The algorithm allows for smooth blending between likelihood- and reward-based costs.

The algorithm is applicable to the longer range learning tasks when the amount of data available to the algorithm is relatively large. One can imagine a system where this algorithm is combined with one of the memory-based solutions presented in earlier chapters. The system would use the short term saliencies of the memory-based algorithm in combination with the long range model shown in this chapter, which would keep track of all data seen so far.

One of the problems of the algorithm is the appropriate choice of the parameter β. In some cases it is convenient to have an asymmetric schedule for positive and negative rewards, which adds another parameter to the set.

In other cases special care must be taken about the fact that both reward signal for the clustering algorithm and the state assignment for the action selection are non-stationary.

As a larger shortcoming, both this and the previous chapters required the input to be pre-segmented. This is not always convenient. For instance, in this framework learning command gestures remains difficult.

Chapter 6

Learning Markov Models

The subtlety of nature is greater many times over than the subtlety of the senses and understanding.

Sir Francis Bacon (1561 - 1626)

6.1 The Goal

This chapter is dedicated to an algorithm for on-line learning of gesture models from unsegmented input stream. The gesture models are built based on their utility in the context of sequential testing and label estimation. In the process of learning, the algorithm builds a global observation model, which is then factorized into independent gesture models with the aid of reward. This chapter shows a technique for a clean-slate learning of the global model and a way to bootstrap extraction of a set of simpler, more compact and efficient primitive recognizers.

Following the formulation of the problem of perceptual learning, (section 2.5), the problem of on-line learning of unsegmented gesture in this chapter is slightly reformulated in order to account for the temporal character of the input observations, as shown in figure 6-1. In this reformulation the notation is changed to reflect the multi-state form of the input models. In this formulation the state, s, previously associated with "the state of the world", will be renamed to a "macro-state", λ, to reflect that it is now more complex and consists of a set of states itself. These

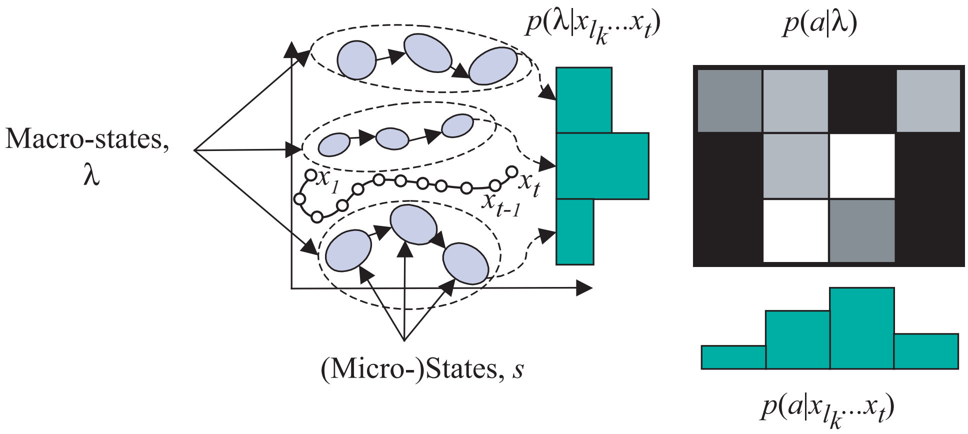


Figure 6-1: The input models (macro-states) are modeled by a collection of states with their own state dynamics. Macro-states, λ are then associated with actions.

component states will be called micro-states, or simply "states", and denoted by s.

For the purposes of this chapter, again, the policy, $p(a|\lambda)$ will be assumed square and diagonal, so that $p(a|\mathcal{X}) = p(\lambda|\mathcal{X})$, where $\mathcal{X} = (x_1, x_2, \ldots, x_t)$ is a sequence of observations. In the setting of agent training, where the agent is trained to perform an action in response to a gesture, the input is a continuous trajectory in some feature space. This trajectory contains the stimulus, which is assumed to be reliably segmented at its end-point, as specified in the later sections. From a series of such inputs accompanied with rewards once the correct action is taken by the agent in response, the algorithm needs to estimate a set of individual models - one per gesture.

With this in mind, the goal of the chapter is to develop a mechanism for clean-slate learning of a set of sequential Macro-states, λ (see fig. 6-1). For a given sequence of measurements, $x_1 ... x_t$, the algorithm needs to find a set of subsequences, $x_{l_k} ... x_t$, best modeled by each of the K models λ_k, select an action, and update parameters of the corresponding macro-state that define its temporal extent, state composition and intra-state dynamics to maximize future reward.

6.2 Overview

The previous chapter presented an algorithm for long term learning of segmented observations. The drawbacks of the algorithm include difficulties in using it with

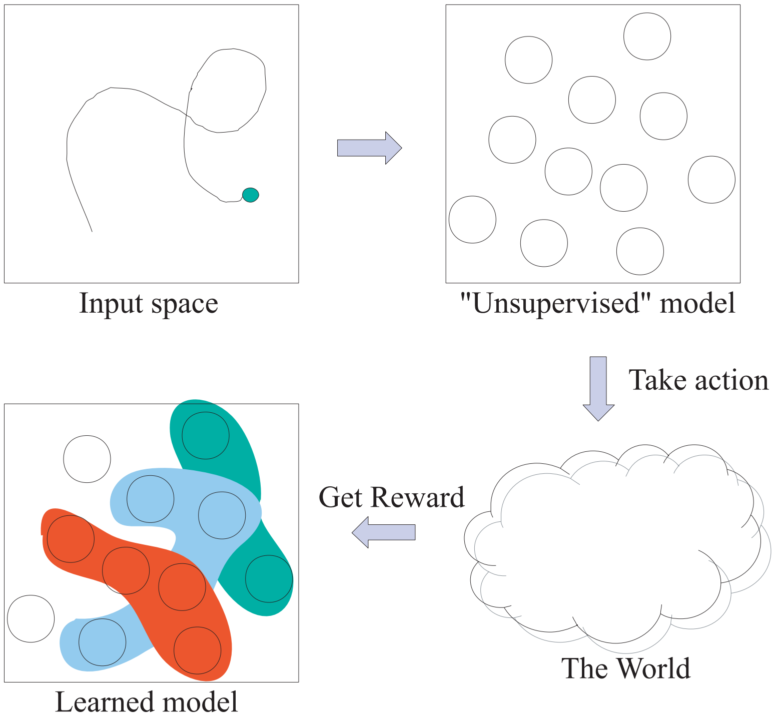


Figure 6-2: Learning gesture models. The observations for an infinite trajectory in the input space. The unsupervised model is built incrementally by adding states as necessary. State occupancy and state transition counts give rise to the Hidden Markov Model which is factorized into separate (possibly overlapping) gesture models or em macro-states as a result of interaction with the environment.

sequences and the fact that it is hard to model unsegmented data. In this chapter another algorithm is developed to continuously learn from an unsegmented gestural input. The idea of it follows in the steps of the previous chapter - it draws heavily on the unlabeled data and unsupervised experiences in order to maximize the utilization of the rare reward.

The main insight that is explored in this chapter is that the reward helps in extracting recurrent gesture models that have provided high utility to the agent. The learning algorithm developed here approximates the input sequences of observations by a set of radial basis functions used as micro-states and collects micro-state Markov statistics. After some period of time the model will be asked to assign a label (identify the macro-state) to the observed sequence. If the label is correct, then the model will be rewarded. After a while the reward structure will help to decompose the collection of micro-states into groups (possibly overlapping) of states that form a a macro-states.

These macro-states will be extracted from the full model to form a learned "gesture vocabulary". The illustration of the process is given in figure 6-2.

6.3 Introduction

Practical aspects of learning from unsegmented gesture are inspired by tricks of animal trainers. The development of this chapter uses formalisms that have been developed for learning and inference in graphical models. A brief introductory exposition of operations with graphical models necessary for understanding of the rest of this chapter is given in this section.

6.3.1 Tricks of Animal Trainers

The immediately apparent difficulty of learning command models from an infinite sequence is that the part of the sequence that is actually relevant to the the model is not identified. This difficulty persists unless some convention about delivery of the reward is adopted and understood by both the learner and the trainer.

The inspiring observation comes from studies in animal behavior and training tricks adopted by animal trainers to aid animals in solving just the problem described above. It has been shown (e.g. [46]) that the reward is the most effective in training when the time interval between the onset of the reward and the behavior selected by the animal is minimal. This shows that animals experience the segmentation problem which is central to the goal of the algorithm that this chapter develops.

Dog trainers have adopted a method, named "clicker training" to help dogs distinguish between a relevant and an irrelevant behavior. The trick is based on preconditioning the dog on a sound of a device, called a clicker, which produces a sharp loud click. First, the click is associated with reward - every time the clicker is used, the dog receives a treat. With such conditioning the sound of the clicker becomes a reliable predictor of a reward - a secondary reinforcer. Now, when the reward needs to be delivered at that exact moment that the desired behavior in the dog is invoked, the clicker is used to indicate that the dog will be rewarded for selecting the current behavior, which allows the dog to precisely segment the observation on which it is being conditioned.

Thus, the algorithm presented here is largely based on the assumption that the mechanism of detecting the end-point of the observation and onset of the desired action is available. This allows us to simplify the goal by limiting the infinite input sequence by the endpoint time index, t.

6.3.2 On-line Gesture Learning

In context of gesture learning and recognition, one would like to develop algorithms that are capable of "clean slate" life-long learning. These learning circumstances are best characterized by two main features: a) the agent has no indication when the next useful observation will arrive, and hence, needs to learn incrementally, after observing each new datum; and b) observations form an infinite sequence with only the latest part being relevant. These two features make the problem of on-line gesture learning difficult - the agent needs to update parameters upon seeing an observation, but it cannot reliably infer which part of the sequence is relevant.

The on-line character of the estimation also brings up another interesting aspect of the learning problem - the labels (or, rather, a subjective measure of their correctness) are only rarely available. That means that transductive inference should be the defining framework for parameter estimation. That is, the algorithm needs to be based on an unsupervised estimator, while leaving the option open for rare but valuable labels. In this case, by the time a label is given, strong unsupervised support for it already exists.

6.3.3 Graphical Models

In motion analysis, gestural input is frequently modeled by a Hidden Markov Model (HMM). An HMM is a sequential probability model, or a probabilistic finite state machine, that, much like its non-stochastic siblings, encodes the assumption that the process under review is memoryless. HMMs received their share of attention in the

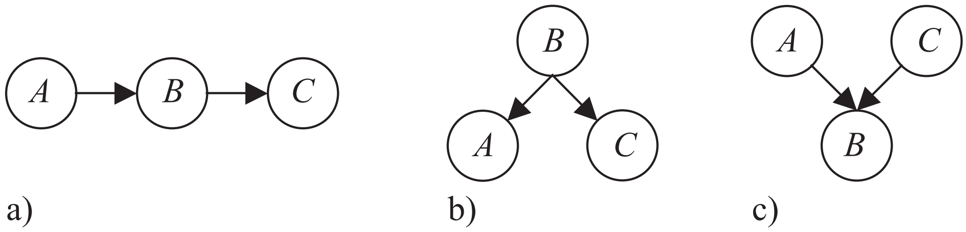


Figure 6-3: a) C is independent of A given B; b) A is independent of C given B; c) A is independent of C, and A is NOT independent of C given B.

community due to the tractability of inference and learning algorithms that have been developed in the speech community.

Recently, HMMs caught their second wind with advances in graph-theoretic approaches to belief propagation and Bayes nets. Graphical models, such as Bayes nets, allow us to view HMMs in context of larger tasks and help formulate many kinds of variations while keeping track of complexity of resulting solutions.

Graphical models help present the problem of probabilistic inference clearly and concisely and formulate all known independencies between all variables involved in the solution. In short, a graphical model represents a random variable by a node in the graph where edges connect the variables between which there might exist a dependency. A directed edge between two nodes represents a *conditional* dependency, encoding an assertion of probability of the child node given its parent.

The probability of any group of nodes (variables) in the network, possibly conditioned on any other group of nodes, can be easily computed from the joint probability distribution over all nodes of the graph. A graphical model represents a family of factorizations of the joint probability as a product of all nodes conditioned on their immediate predecessors:

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i | x_{\pi_i}) \tag{6.1}$$

where x_{π_i} is the set of nodes that are parents of x_i.

Three canonical graphs, shown in figure 6-3, encode three main types of conditional independencies that can arise in a problem represented by a Bayes net. Without much detail, while often representing causality or conditional independence between the variables the configuration in 6-3 a) asserts that C is independent of A given B. As follows from eqn 6.1, the joint probability distribution for this configuration is calculated as follows:

$$p(A, B, C) = p(A)p(B|A)p(C|B) \tag{6.2}$$

The configuration in figure 6-3 b) encodes the fact that A is independent of C given B with the following joint:

$$p(A, B, C) = p(B)p(A|B)p(C|B) \tag{6.3}$$

The last canonical graph in figure 6-3 c) indicates that A is independent of C, and A is NOT independent of C given B. The following computes the joint distribution of this configuration:

$$p(A, B, C) = p(A)p(C)p(B|A, C) \tag{6.4}$$

These three canonical graphs serve as building blocks in modeling independencies between variables in the model.

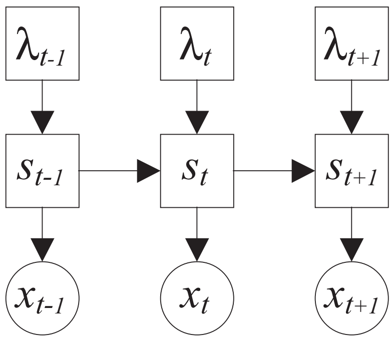


Figure 6-4: Independence structure of the model. An observation, x_t is generated by a micro-state s_t which is conditioned on the gesture model, λ_t.

Using graphical models makes it easy to trace dependencies between variables in the graph. For instance, the problem addressed in this chapter can be graphically represented by a graph shown in the figure 6-4. The graph shows the presence of Markov assumption in the process of state switching - each state is only dependent on its predecessor. It also asserts that the transition from state to state is affected by the current gesture model, λ_t. Notationally, everywhere in this chapter where a graphical model is used a node shown by a circle will represent a continuous random variable, while the one shown by a square will represent a discrete one.

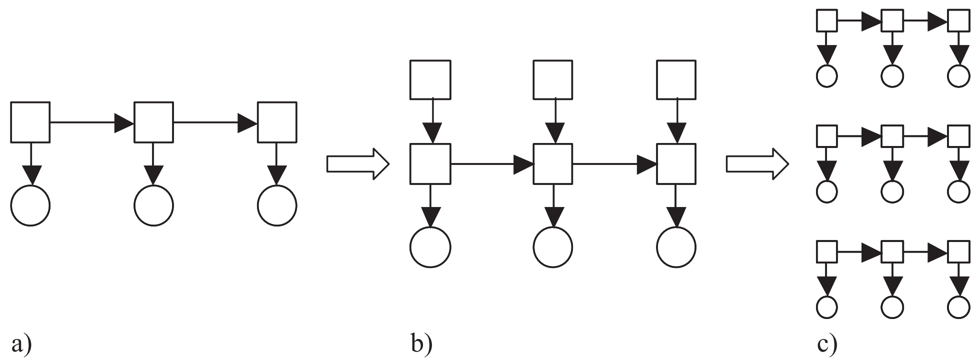


Figure 6-5: Reward-driven on-line decomposition. a) A large hidden Markov model is estimated on-line by counting state occupancy and state transitions. b) State attribution is found with reward which with some probability assigns each micro-state state to a gesture model. c) Monte-Carlo sampling fits independent models to each macro-state

In terms of graphical models the structure of the process of decomposition of the infinite trajectory representation into a set of individual models (states) is schematically shown in figure 6-5. The process is based on a long term unsupervised estimation of the transition model in the visual field, fig. 6-5 a). When rewards are delivered, each micro-state is given a gesture model attribution, fig. 6-5 b), which allows to factorize the full model into a set of gesture-specific models. These models are subsequently sampled to build a set of independent primitives, fig. 6-5 c).

6.3.4 Previous Work

A large part of this work is built upon research done in gesture recognition and speech processing. The algorithm starts with the membership of each state being completely independent and proceed to implicitly discover causal dependencies in the membership structure.

A technique useful for on-line sequence processing, based on an "open-ended" Dynamic Programming algorithm was shown in [14]. This chapter uses the idea of this technique to formulate the on-line multi-model Viterbi approximation algorithm for evaluation of likelihood-based measures on sub-sequences of observations.

Wilson [86], devoted his dissertation to a study of techniques for modeling gestural input. The models used in his work are very similar to those used here. One important distinction is that while Wilson's work used models that have been physically instantiated at the run time, the current work focuses on techniques of building them on-line, perhaps at the expense of some simplifications.

To evaluate the probability structure of the model the algorithm of this chapter uses a fairly general view of probabilistic sequential machines grounded in graphical models and Bayes nets literature ([59, 40, 41, 60]). A study of several algorithms for learning and inference in Dynamic Bayes Nets with similar structure was presented by Pavlović and Rehg in [58]. The authors examined several approximate solutions to the "loopy" DBN problem with continuous state. As is the case here, the model is in a sense simpler, having a discrete state, and, more importantly, fixed switching states.

The work shown by Hong, Turk and Huang in [34] demonstrates an elegant solution to gesture learning via extraction of a finite state machine. The algorithm begins with collecting data and spatially segmenting it into an unspecified number of states with a given variance. Then temporal structure of state transition is learned and used for classification. In the work of this thesis, the algorithm has no access to the data beforehand and cannot pre-segment it. Instead, it performs an RBF approximation of the incoming sequences to form a set of micro-states. Micro-states, as well as new gesture models, are created on demand, when either the coverage of the input space is found insufficient, or some indication that a new gesture is being shown is given to the system.

6.4 Multi-Model Gesture Representation

In this chapter it is assumed that a single gesture consists of a trajectory in some feature space that is switching from state to state. Some of these states are related to a stimulus, (form meaningful gestures) while others are the "filler" states. The state's attribution to a particular gesture is represented by a probability distribution over possible models. For states having strong membership this distribution will have a very low entropy, while the distribution for "background" states will be close

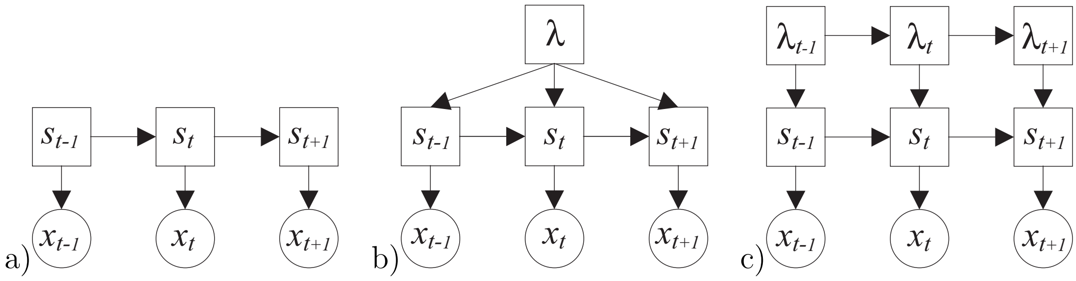


Figure 6-6: a) Hidden Markov Model; b) fixed and c) switching model of a sequence generator.

to uniform. The learning needs to accomplish the following a) for a sequence of observations extending infinitely into the past compute the probability of each of the models being responsible for producing at least the ending part of it; and b) update parameters of this model with a simple system of rewards and punishments.

6.4.1 Probabilistic Models of Gesture

The hidden Markov model has become the weapon of choice in modeling sequential data. The inferrential structure of the HMM is conveniently represented via a graphical model, shown in the figure 6-6 a). This model asserts that the sequence of observations is generated by a process that switches between a set of discrete states. Furthermore, the switching process is subject to the Markov assumption - the current state of the process depends only on the previous state.

There exists a variety of possibilities when it comes to representing the generation of a sequence of observations from a finite vocabulary of sequential models. Assigning a variable λ to represent a particular element of the vocabulary, one alternative is to condition all the states on the probability of the generator being in a particular model, as shown in figure 6-6 b). In a generative sense, this “fixed” model has one parameter that specifies the element of the vocabulary currently in effect.

This model does not model a process according to which a particular value of λ is chosen. Given a particular sequence of observations representing a single gesture the inference can be made as to which one of a set of gestures has been observed.

A switching model (figure 6-6 c)) allows queries of a different kind, modeling the process that switches between the gestures of the set according to some switching dynamics. Given a sequence of observations, the model allows to determine which gestures have been observed and in what order. This type of model has been extensively explored by Pavlović and Rehg [58].

Despite the fact that some progress in inference in loopy networks has recently been made (e.g. [90]), inference in both fixed and switching models is notoriously difficult due to presence of cycles in the network topology.

The fixed model is of the main interest to the work of this chapter since the goal of the on-line algorithm is to find the probability of a single gesture having just been performed. The estimation of this probability is typically simple on a fixed length sequence. It is more difficult when each model can potentially be responsible for a subsequence of different length. Naïvely, one can consider computing the likelihood of the best fitting sequence, then take its value, multiply by the model prior and normalize, hoping to get the posterior probability. Unfortunately this computation results in an estimate that is biased towards assigning higher probabilities to shorter sequences, since the likelihood function is exponentially decreasing with the length of the sequence.

An alternative measure is to compute the posterior based on the length-normalized likelihood of the sequence. Then the posterior probability can be approximated from a set of these normalized likelihoods. The difficulty of computing the likelihood in the presence of loops in the model can be alleviated by *cut-set* approximation (see Pearl [59] and Jensen [40]).

The cut-set method is based on the approximation that instantiates (gives a particular value) to a set of strategically chosen variables in the graph such that the instantiation turns the cyclic graph into a tree. Then the inference is performed on the tree for each value of the variable set. To get the final quantity of interest, the results are averaged. Of course, such approximation is difficult to apply to a continuous variable.

In the cut-set method it is often unclear which nodes to instantiate. In contrast, in the case of the problem at hand, not only does instantiation of λ break all cycles as

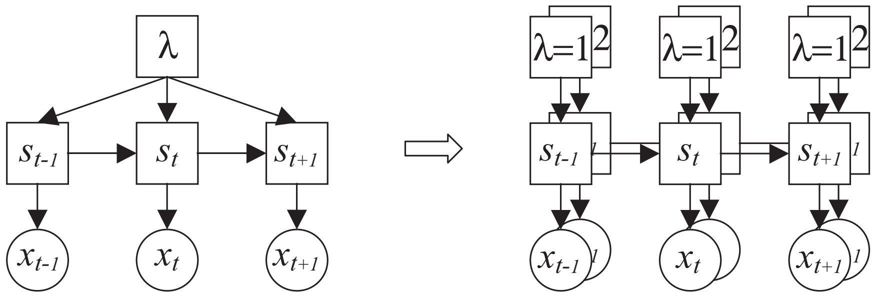


Figure 6-7: Illustration of the cut-set conditioning. In the model on the left a variable is instantiated such that the remaining network forms a tree. Then a traditional propagation algorithm can be used to query the network. The process is repeated for each value and the result is averaged.

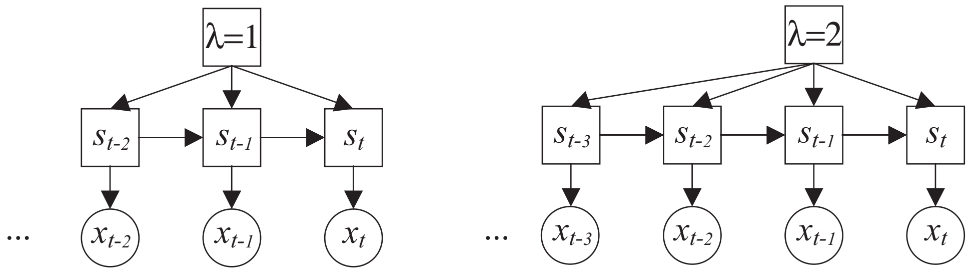


Figure 6-8: Illustration of the search in the cut-set tree. The model claims different subsequences of the input for different values of λ.

shown in figure 6-7, but it also allows us to solve the problem of subsequence search. The figure provides an illustration of how the loops can be treated. By assigning a particular value to λ, the graph turns into a tree, where inference is simple. Then the search is performed on the tree to find the best starting time index of the input sequence for which the model with the given value of λ can claim responsibility (see fig. 6-8). The following sections will describe the algorithm outlined above in more detail.

6.4.2 State Space Model

A "state space" is a set $\mathcal{S} = \{S, N, T, Q\}$, where S is a set of (Gaussian) states, N - a state sample count, T - a state transition count, and Q - is a state model membership function.

The state, $s \in S$, represented by a Gaussian RBF kernel, is parameterized by two parameters - the kernel center, c, and the state covariance, Σ:

$$s_k = e^{((x-c_k)^T \Sigma_k^{-1}(x-c_k))} \tag{6.5}$$

where Σ is chosen to be isotropic: $\Sigma = hI$, such that a state is represented by a spherical Gaussian with width h. The state sample count, N, is a number of samples that has been assigned to each state so far. An entry $T_{i,j}$ into the transition count matrix, T, indicates how many times a sample fell into the j-th state while its predecessor was assigned to the state i. A state model membership function, Q, is a count of rewards for attributing the state to a particular model. That is, for a given value of state and gesture indices, s and λ, the value of the state membership function, $Q_{k,i} = Q(s = i, \lambda = k)$, is the amount of reward (or average discounted reward), $\bar{r}(s, \lambda)$, that has been collected while the i-th state was attributed to k-th model such that:

$$Q(s, \lambda) = \bar{r}(s, \lambda) \tag{6.6}$$

State parameters allow us to calculate maximum likelihood estimates of probabilities of interest when necessary, as will be shown below.

The state space model is built on-line "as needed". The states represent a spatial distribution of the trajectory points that the system has seen so far. Once a new trajectory sample, x^n is received, the squared Mahalanobis distance from it to the closest kernel is computed:

$$D = \min_k \left\{ (\mathbf{x} - \mathbf{c}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{c}_k) \right\} \tag{6.7}$$

$$i = \arg\min_k \left\{ (\mathbf{x} - \mathbf{c}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{c}_k) \right\} \tag{6.8}$$

If that distance is found to be smaller than a threshold, d_T, both sample and transition counts, N_i and $T_{j,i}$, of that state are increased by one. If, on the other hand, the distance to the closest kernel is large, a new kernel, centered at the sample x^n, is added to the model, while its initial sample and transition counts, as well as the model membership vector, $Q_{k,i}$ are set to one.

As a result of this process, a model consisting of a large number of states linked by the transition count matrix is incrementally built. The state counts allow us to compute simple state occupancy and state transition statistics, which can be used to update the state model membership, such as:

$$
\begin{aligned}
p(\lambda_k|s_i) &= \frac{Q_{k,i}}{\sum\limits_m Q_{m,i}} && (6.9) \\
p(s_j|s_i) &= \frac{T_{i,j}}{\sum\limits_m T_{i,m}} && (6.10) \\
p(s_i) &= \frac{N_i}{\sum\limits_m N_m} && (6.11)
\end{aligned}
$$

The complete model and the dependency between model variables can be represented graphically , as shown in the figure 6-4. In this diagram, each state, s_t is responsible for an observation x_t and is conditioned on the macro-state, λ_t.

6.4.3 Inference and Testing

The algorithm is built for training an agent to correctly respond to an observation by selecting an action. Here the system is presented with an unsegmented gesture and is asked it to label it. The complication is that the starting point of the sequence is not indicated.

As before, the action selection in the algorithm is based on sampling from the posterior distribution, $p(\lambda_k|\mathcal{X})$, where $\mathcal{X}$ is all available observations. Hence, the goal of action selection is to compute the probability of a given model being responsible for the observation sequence so far. This computation needs to be performed for each new sample, x_t.

Typically, in the case of a finite segmented sequence one would use the Likelihood Ratio test guaranteeing the minimum Bayes risk of classifying the sequence into one of a set of classes:

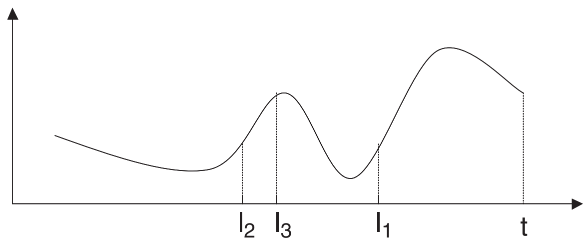


Figure 6-9: Problems of finding the "best" sequence - (1) the beginning of the sequence is unknown, (2) the best sequence for each model can be of different length.

$$\lambda = \arg\max_k (p(\lambda_k | x_1 ... x_t))$$

However, the problem that the algorithm is facing is that only the final part of the whole sequence is related to the reward, while each of the models can provide support for sequences of different length. In other words, one would like to use a similarity measure that considers comparing sequences of different lengths. One such measure is the expected amount of agreement between the sequence of observations and the "best" state sequence, $s^*_{l_k} \dots s^*_t$, that the k-th model can provide for these observations, assuming that such can be extracted. Using the shorthand $\lambda_k \equiv (\lambda_{l_k} = k, ..., \lambda_t = k)$, with the usual Markov assumption and factorizations of the model shown in figure 6-4, the likelihood of the k-th model can be written as follows:

$$\begin{aligned} & p(x_{l_k} ... x_t, s^*_{l_k} ... s^*_t | \lambda_k) = \\ & \quad p(s^*_{l_k} | \lambda_{l_k} = k) p(x_{l_k} | s^*_{l_k}) \prod_{\tau=(l_k+1)}^{t} p(s^*_\tau | s^*_{\tau-1}, \lambda_\tau = k) p(x_\tau | s^*_\tau) \end{aligned} \tag{6.12}$$

Taking a logarithm of 6.12 gives the following relation:

$$\begin{aligned} & \log p(x_{l_k} ... x_t, s^*_{l_k} ... s^*_t | \lambda_k) \approx \\ & \sum_{\tau=l_k}^{t} \log \left[p(s^*_\tau | s^*_{\tau-1}, \lambda_\tau = k) p(x_\tau | s^*_\tau) \right] = \\ & (t - l_k) E \left[\log p(x_i, s^*_i | s^*_{i-1}, \lambda_i = k) \right] = (t - l_k) \eta^{*,k}_{l_k} \end{aligned} \tag{6.13}$$

where $(t - l_k)$ is the length of the "best" subsequence ending at the t-th sample, and the expectation, $\eta^{*,k}_{l_k}$ is the average "disagreement" between the sequence of observations including samples starting at the time index l_k and ending at t and the "best" state

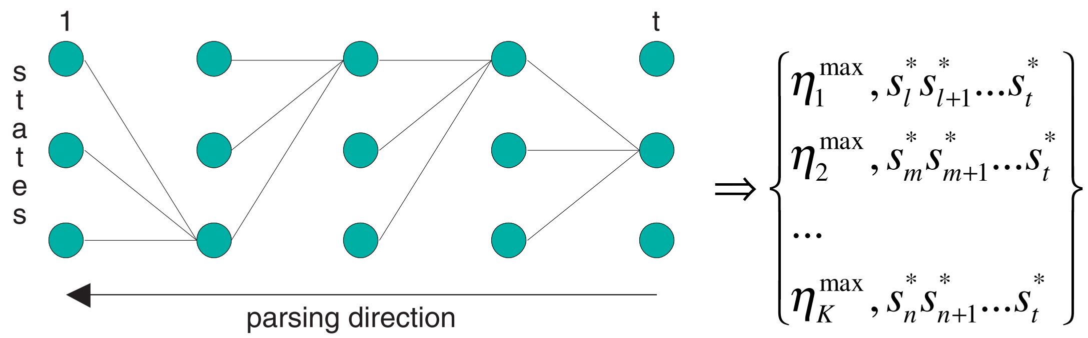


Figure 6-10: Viterbi parse

sequence that the k-th model can provide. The expectation is easily calculated from eqn. (6.13). For every sequence from l_k to t, the value of $\eta_{l_k}^{*,k}$ gives its time-normalized model score:

$$\eta_{l_k}^{*,k} \approx \frac{\log p(x_{l_k}...x_t, s_{l_k}^*...s_t^*|\lambda_k)}{t - l_k} \tag{6.14}$$

6.4.4 Multi-Model Viterbi Approximation

The equation (6.14) provides neither the means for computing the state sequence $s_{l_k}^*..s_t^*$ nor the starting time index l_k. Fortunately, both can be found via the approximate Viterbi algorithm, provided that the likelihood of equation (6.13) can be calculated for a given state sequence. This calculation requires a model-conditioned transition matrix, $p(s_t|s_{t-1}, \lambda_t)$, which is difficult to obtain on the small input sample, as it consists of MN^2 entries, where M is a number of gestures in the full model, and N - a number of states. Instead it can be approximated with quantities that are already available. Note that the following holds:

$$\begin{aligned} p(s_t|s_{t-1}, \lambda_t) &= \frac{p(s_t, \lambda_t|s_{t-1})}{p(\lambda_t|s_{t-1})} \\ &= \frac{p(\lambda_t|s_t, s_{t-1})p(s_t|s_{t-1})}{p(\lambda_t|s_{t-1})} \\ &= \frac{p(\lambda_t, s_{t-1}|s_t)p(s_t|s_{t-1})}{p(\lambda_t|s_{t-1})p(s_{t-1}|s_t)} \end{aligned} \tag{6.15}$$

The joint probability $p(\lambda_t, s_{t-1}|s_t)$ is, again, difficult to estimate, but it can be approximated in the maximum entropy sense, from its marginals, $p(\lambda_t|s_t)$ and $p(s_{t-1}|s_t)$:

$$p(\lambda_t, s_{t-1}|s_t) \approx p_{ME}(\lambda_t, s_{t-1}|s_t) = p(\lambda_t|s_t)p(s_{t-1}|s_t) \tag{6.16}$$

This approximation essentially neglects the dependency of λ_t on s_{t-1} given s_t and amounts to reversing the arrows directed from λ to s in the fig. 6-4.

Now, substituting 6.16 into 6.15, the approximation to the model-conditioned transition matrix can be calculated:

$$\begin{aligned} p(s_t|s_{t-1}, \lambda_t) &\approx \frac{p_{ME}(\lambda_t, s_{t-1}|s_t)p(s_t|s_{t-1})}{p(\lambda_t|s_{t-1})p(s_{t-1}|s_t)} \\ &= \frac{p(\lambda_t|s_t)p(s_{t-1}|s_t)p(s_t|s_{t-1})}{p(\lambda_t|s_{t-1})p(s_{t-1}|s_t)} = \frac{p(\lambda_t|s_t)p(s_t|s_{t-1})}{p(\lambda_t|s_{t-1})} \end{aligned} \tag{6.17}$$

As an aside, the computational savings achieved as a result of the approximation are actually quite significant. The number of parameters in both models grow with the number of states and the number of models. For N states and M models, the exact representation requires estimation of $E = MN^2$ parameters, while approximation reduces this number to $A = MN + N^2$. The ratio of these two numbers gives the relative computational savings as the number of states grows:

$$\frac{A}{E} = \frac{M+N}{MN} \tag{6.18}$$

that is, the the difference in growth in complexity between the two models is proportional to the ratio between a sum and a product.

It is important to say that the above approximation to the model-conditioned transition $p(s_t|s_{t-1}, \lambda_t)$ is not necessary. If enough data is available to reliably estimate it directly, one should always do so. However, in the face of the small sample this approximation provides a good starting point for the algorithm as it produces a *maximum entropy* model. The independence assumptions as stated above will result in presence of extra links between states, which can be removed with further training. In contrast, missing links are difficult to recover.

The inference algorithm presented in this chapter uses a time-normalized multi-model version of Viterbi approximation, which is shown below. Note that one needs to obtain a set of values of η - one per time step. This is clearly inefficient. However, since the Viterbi algorithm is recursive, it can be "reversed" and instead run from the current time step back into the past, providing the score, η incrementally, as proposed in [14]. Since η is time-normalized, it can be directly used for comparison. Using $\delta_\tau^k(i)$ to denote the highest probability of the k-th model along a single path through the state set, which accounts for the last $(t-\tau)$ observations and starts in state i, and $\psi_t^k(i)$ the argument that maximizes that quantity, the inference algorithm is formalized as follows:

Multi-Model Viterbi Parse

1. `Initialize` $(\forall i : 1 \le i \le N)$

$$\begin{aligned}\delta_t^k(i) &= \log p(x_t | s_t = i)\\ \psi_t^k(i) &= 0\end{aligned}$$

2. `Recurse` $(\forall i, \tau : 1 \le i \le N;\ (t-1) \ge \tau \ge 1)$

$$\begin{aligned}\delta_\tau^k(j) &= \max_{1\le i\le N}\Big\{\tfrac{1}{t-\tau+1}\big[(t-\tau)\delta_{\tau+1}^k(i)\\ &\quad + \log p(\lambda_{\tau+1} = k | s_{\tau+1} = i) + \log p(s_{\tau+1} = i | s_\tau = j) - \log C\big]\Big\}\\ &\quad + \log p(x_\tau | s_\tau = j)\\ \psi_\tau^k(j) &= \arg\max_{1\le i\le N}\Big\{\tfrac{1}{t-\tau+1}\big[(t-\tau)\delta_{\tau+1}^k(i)\\ &\quad + \log p(\lambda_{\tau+1} = k | s_{\tau+1} = i) + \log p(s_{\tau+1} = i | s_\tau = j) - \log C\big]\Big\}\end{aligned}$$

3. `Terminate`

$$\begin{aligned}\eta_1^{*,k} &= \max_{1\le i\le N}\left[\delta_1^k(i)\right]\\ s_1^{*,k} &= \arg\max_{1\le i\le N}\left[\delta_1^k(i)\right]\end{aligned}$$

4. `Recover the sequence` $(\forall \tau : 2 \le \tau \le t)$

$$\begin{aligned}s_\tau^{*,k} &= \psi_{\tau-1}^k(s_{\tau-1}^{*,k})\\ \eta_\tau^{*,k} &= \delta_\tau^k(s_\tau^{*,k})\end{aligned}$$

The values under the log in step 2 represent a logarithm of the conditional transition probability, $p(s_\tau | s_{\tau-1}, \lambda_k)$, calculated via the equation (6.17), where C stands for the normalization constant (denominator of eqn. (6.17)). Note the re-scaling of δ and ψ at each step by the current sequence length (L_k of eqn. 6.13).

After the last step of the algorithm for each model one has a sequence of most likely states, $S^{*,k} = s_1^{*,k}...s_t^{*,k}$ and a sequence of "per sample" scores, $\eta_1^{*,k}...\eta_t^{*,k}$. Within this sequence the search for the time index, l_k, of the "best" sequence corresponding to the k-th model is performed:

$$\begin{aligned} \eta_k^{max} &= \max_{1 \leq \tau \leq t-w} \left[\eta_\tau^{*,k}\right] \\ l_k &= \arg\max_{1 \leq \tau \leq t-w} \left[\eta_\tau^{*,k}\right] \end{aligned} \tag{6.19}$$

These equations allow us to calculate the time-normalized posterior using Bayes' rule:

$$\hat{p}(\lambda | x_{l_k}...x_t, s_{l_k}^{*,k}..s_t^{*,k}) = \frac{e^{\eta_k^{max}} p(\lambda)}{\sum_\lambda e^{\eta_m^{max}} p(\lambda)} \tag{6.20}$$

where $e^{\eta_k^{max}}$ is the per-sample likelihood, which is the result of the fact that η is the log-likelihood. The label is chosen by sampling from this distribution, which is essentialy an implementation of the following classification rule:

$$\lambda = \arg\max_k (\hat{p}(\lambda_k | x_{l_k}...x_t, s_{l_k}^{*,k}..s_t^{*,k})$$

As an aside - for the task of label selection one can always parameterize the expression for the posterior with the average discounted reward, similarly to how it was done in the chapter 3 (cf. eqn. 3.5):

$$\hat{p}_\beta(\lambda | x_{l_k}...x_t, s_{l_k}^{*,k}..s_t^{*,k}) = \frac{e^{\beta \bar{r} \eta_k^{max}} p(\lambda)}{\sum_\lambda e^{\beta \bar{r} \eta_m^{max}} p(\lambda)} \tag{6.21}$$

where β is an entropic parameter related to the sensitivity of the posterior to the reward, amd $\bar{r}$ is the "baseline" value of reward. The reward is maintained as follows:

$$\bar{r}_t = \bar{r}_{t-1} + \alpha(r - \bar{r}_{t-1}) \tag{6.22}$$

where α is, again, a learning rate. With a proper choice of β, the distribution in eqn.

6.21 becomes “self-regulated”. This parameterization allows for more exploration in the absence of the reward, while switching to exploitation mode when reward is high.

6.4.5 Reward distribution

With the full model the algorithm maintains the state-model membership function, $p(\lambda|s)$, which is readily computed from the state-model value, Q, (see eqn. 6.9).

The state-model value function is updated every time a model, λ_k, that has been selected by the sampler, based on the distribution of equation (6.19) is rewarded:

$$Q(s_\tau \in S^{*,k}, \lambda = k) = r + \gamma Q(s_\tau \in S^{*,k}, \lambda = k) \tag{6.23}$$

where r is the instantaneous reward. The value of the state being attributed to a model is increased by some value if the whole model has been rewarded or slowly decreased otherwise.

6.4.6 State Sharing

One problem with the above approach is that sharing states between different gestures is difficult. If a state can be a part of two gestures, then, since it is tied to the particular model by a posterior probability, $p(\lambda|s)$, increasing the probability for one state decreases it for others. To avoid that, an additional auxiliary binary vector variable, z can be added to each state introduced to indicate the state assignment. The binomially distributed variable z indicates the “coherence” of the relationship between a state and a model.

Let the k-th component of the variable z take the value 1 if the state belongs to the model k, and 0 otherwise. The graphical model representing the relation between a gesture models and states is shown in figure 6-11. With this variable, the model membership function, Q, is split into two components - one responsible for collecting the positive reward, and the other - for negative. Now the table Q is a function of state and a gesture model, such that:

$$
\begin{aligned}
Q(s, \lambda, z = 0) &= \bar{r}^{-}(s, \lambda) \\
Q(s, \lambda, z = 1) &= \bar{r}^{+}(s, \lambda)
\end{aligned}
\tag{6.24}
$$

Correspondingly to the change in the structure, the update equation for Q (eqn. 6.23) should be changed:

$$
\begin{aligned}
Q(s_\tau \in S^{*,k}, \lambda = k, z = 0) &= -r + \gamma Q(s_\tau \in S^{*,k}, \lambda = k, z = 0) \quad \text{if } r < 0 \\
Q(s_\tau \in S^{*,k}, \lambda = k, z = 1) &= r + \gamma Q(s_\tau \in S^{*,k}, \lambda = k, z = 1) \qquad \text{if } r > 0
\end{aligned}
\tag{6.25}
$$

The probability distribution over values of z can be computed, based on the current state of the table Q. The probability $p(z = 1|s_i, \lambda_k)$ specifies the degree of coherence between the state s_i and the gesture model λ_k. Introduction of the auxiliary variable z does not change the fundamental computations. It is clear from comparing the model in figure 6-11 with the one in figure 6-4, that the only computation affected by it is calculation of the transition probability, $p(s_t|s_{t-1}, \lambda, z)$:

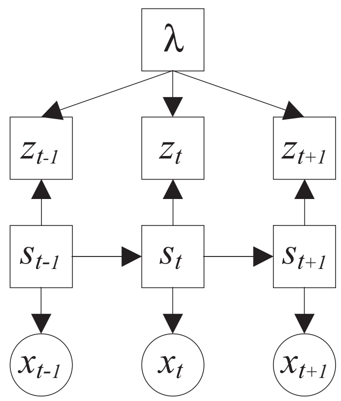


Figure 6-11: Independence structure of the state sharing model.

$$
\begin{aligned}
p(s_t|s_{t-1}, \lambda_t, z_t) &= \frac{p(s_t, z_t|s_{t-1}, \lambda_t)}{p(z_t|s_{t-1}, \lambda_t)} \\
&= \frac{p(z_t|s_t, s_{t-1}, \lambda_t)p(s_t|s_{t-1}, \lambda_t)}{p(z_t|s_{t-1}, \lambda_t)} \\
&= \frac{p(z_t|s_t, \lambda_t)p(s_t|s_{t-1})}{p(z_t|s_{t-1}, \lambda_t)}
\end{aligned}
\tag{6.26}
$$

With this relation, the likelihood function can be computed similar to equation 6.12. Using the shorthand $z_1 \equiv (z_{l_k} = 1, ..., z_t = 1)$:

$$p(x_{l_k}...x_t, s^*_{l_k}...s^*_t|\lambda_k, z_1) =$$
$$p(s^*_{l_k}|\lambda_{l_k} = k, z_{l_k} = 1)p(x_{l_k}|s^*_{l_k}) \prod_{\tau=(l_k+1)}^{t} p(s^*_\tau|s^*_{\tau-1}, \lambda_\tau = k, z_\tau = 1)p(x_\tau|s^*_\tau) \quad (6.27)$$

and, subsequently:

$$\eta^{*,k}_{l_k} \approx \frac{\log p(x_{l_k}...x_t, s^*_{l_k}...s^*_t|\lambda_k, z_1)}{t - l_k} \quad (6.28)$$

The approximation algorithm, shown in section 6.4.4 now uses the conditional transition probability given in equation 6.26 in place of $p(s_t|s_{t-1}, \lambda_t)$ while setting $z = 1$.

The definition of "coherence" is somewhat loose and leaves some room for interpretation of the form of the distribution $p(z|s, \lambda)$. Obviously, the Maximum Likelihood estimate of it is given by:

$$p_{ML}(z = 1|s, \lambda) = \frac{Q(s, \lambda, z = 1)}{Q(s, \lambda, z = 0) + Q(s, \lambda, z = 1)} \quad (6.29)$$

This form of $p(z|s, \lambda)$ is not always convenient. One shortcoming of this form is that the state that has never received any evidence is assumed to be 50% coherent with every gesture. One possible solution to this situation is to initialize Q unfairly and give $Q(s, \lambda, z = 0)$ a high initial value. Another possibility is to weigh the maximum likelihood estimate by some factor related to its entropy, such that high entropy estimate of $p_M L(z|s, \lambda)$ results in low value of $p(z = 1|s, \lambda)$.

6.5 Model Extraction

Since majority of the states in the full model that is learned will be the "filler" states, for the efficient model representation one would like to extract simpler and more efficient models of each gesture. The fundamental problem in the model extraction is that even though the algorithm might have a pretty good idea about which states belong to each particular model, at some point it would need to commit to a hard

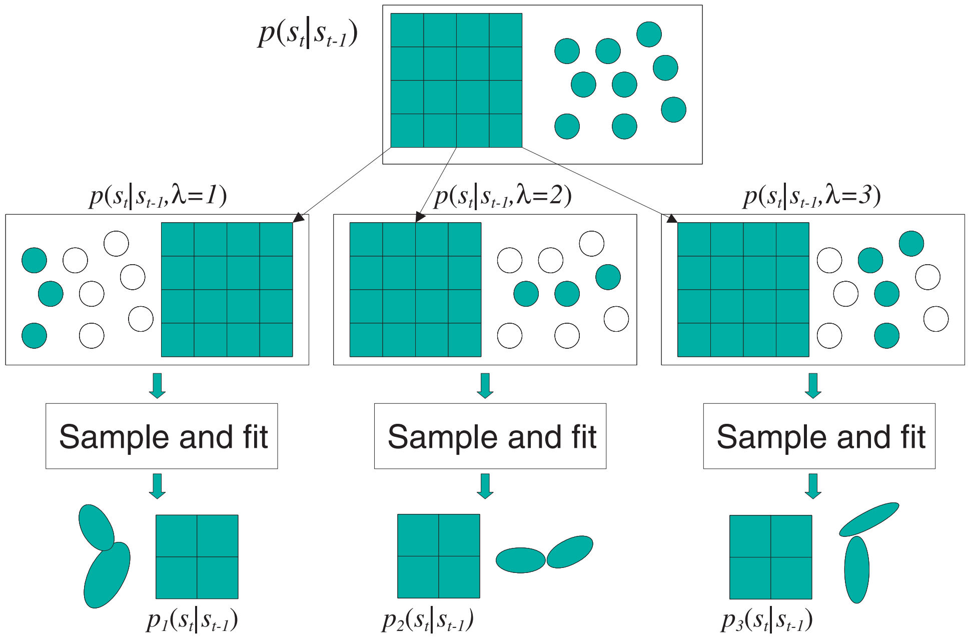


Figure 6-12: Sampling scheme

assignment of states to models. To avoid that a different route is chosen - it fits a simpler model to the set of states weighted by their corresponding probability of the full model. Short of the analytical solution, the problem is solved via a Monte Carlo simulation by sampling from it and generating large amounts of data to which the new model is fit. In order to build a sampler one needs a good estimate for the initial state of the sub-model. The corresponding probability distribution over all states, π_k, can be calculated from the transition matrix, $A_k = p(s_t|s_{t-1}, \lambda_k)$. Then, starting with a state chosen from π_k the algorithm can generate a sequence of states from A_k with observations generated at each state, until a final state is encountered.

Computations involved in recovery of the transition matrix, A_k, have already been shown in the equation 6.17. In order to run the simulation the algorithm needs a termination probability model, ϕ_k - a probability distribution over states that the state is final; and the starting state probability model π_k. The former is computed from the joint transition $p(s_t, \lambda_k|s_{t-1})$ as the *out-of-model* probability - a probability of transitioning into a state that belongs to any other model:

$$\phi_k(i) = 1 - \sum_j p(s_t = j, \lambda_k | s_{t-1} = i) \tag{6.30}$$

π_k is computed based on the heuristic[1] that the state has a higher probability of being the starting state if on average it results in a longer sequence if the process starts in this state. In other words for each state the algorithm needs to compute the expected number of hops induced by the transition matrix until reach the final state is reached. For a joint transition matrix $J_k = p(s_t, \lambda_k | s_{t-1})$ probability of reaching any state from any other in exactly n hops from every state in the model is computed by taking the power of the matrix J_k. Hence, the required expectation for this transition matrix is computed as the infinite sum of these powers multiplied by corresponding number of hops. Then, π_k are calculated from the row-wise maximums of J_k^∞:

$$J_k^\infty = \sum_{n=1}^{\infty} n J_k^n = J_k (I - J_k)^{-2} \tag{6.31}$$

$$\pi_k(i) = \frac{\max\limits_j (J_k^\infty(i,j))}{\sum\limits_i \max\limits_l (J_k^\infty(i,l))} \tag{6.32}$$

The model $\{\pi_k,\ A_k,\ \phi_k,\ \{\mu_l,\ \Sigma_l\}_{l=1..N}\}$ allows to sample the k-th model to produce enough data to train a simpler isolated model as shown in the experimental section.

6.6 Algorithm Summary

The following presents the summary of the algorithm described in this chapter.

[1] which is not always true

On-Line Gesture Learning Algorithm

1. Initialize the model to $S = N = T = Q = \mathcal{X} = \varnothing$, where $\mathcal{X}$ is the sample buffer;

2. For every new sample, x^n:

   (a) Append x^n to the sample buffer $\mathcal{X} = \mathcal{X} \cup x^n$;

   (b) Compute the minimal distance, D (eqn. 6.8);

   (c) If $D < 2.5$, add a state to the state set $S = S \cup s_i$;

   (d) Update occupancy and transition counts, N and T;

   (e) Compute model statistics (eqn. 6.9);

   (f) If queried:

      i. From the buffer $\mathcal{X}$ compute the posterior, $\hat{p}(\lambda | x_{l_k} ... x_t, s_{l_k}^{*,k} .. s_t^{*,k})$, (eqn. 6.20);

      ii. From $\hat{p}(\lambda | x_{l_k} ... x_t, s_{l_k}^{*,k} .. s_t^{*,k})$ select an action;

      iii. Collect reward, update Q (eqns. 6.23, 6.25);

      iv. Clear the sample buffer $\mathcal{X}$;

3. For every individual gesture model:

   (a) Compute π and ϕ (eqns. 6.32, 6.30);

   (b) Sample the model and produce the corpus, $\mathcal{D}_k$;

   (c) Train a Hidden Markov Model on $\mathcal{D}_k$.

6.7 Experiments

This section shows some experiments with the online model building and model extraction. All experiments use a simple cartesian feature space to simplify the presentation.

6.7.1 Mouse Gesture

In the first experiment the algorithm is used to learn a set of models for classification of a mouse gesture. The algorithm proceeds as follows:

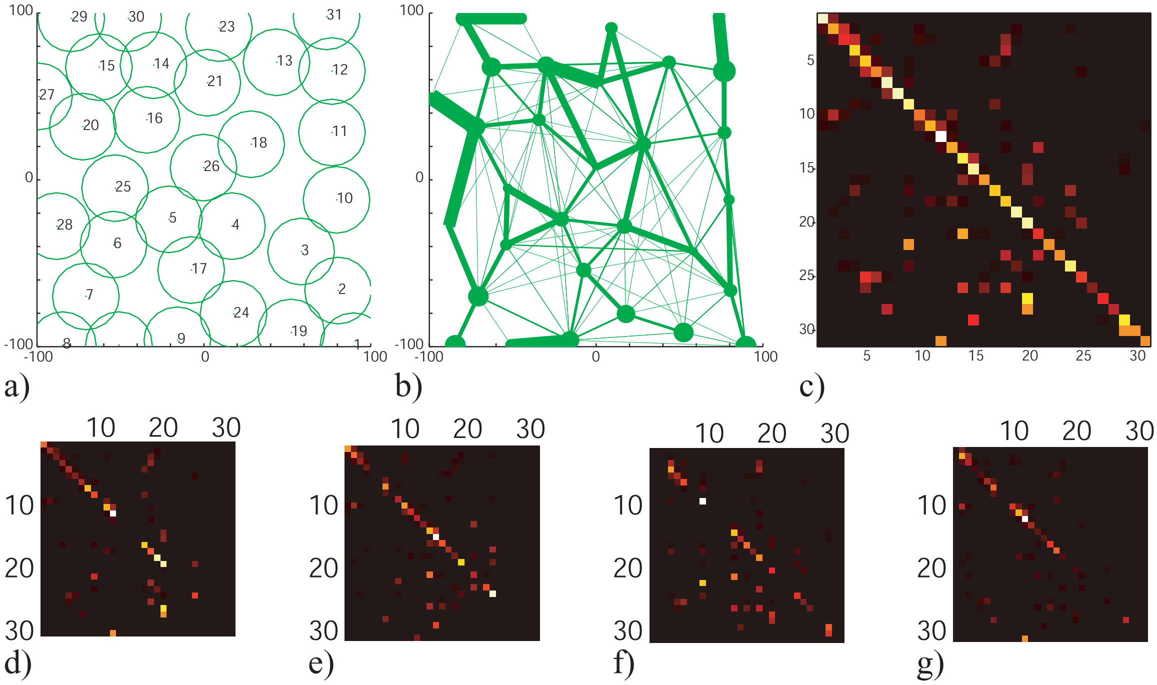


Figure 6-13: a)-c) a full HMM structure resulting after performing several unsegmented mouse gestures. a) A full RBF state space; b) resulting link structure. Vertices show means of RBF states, links - state transitions. The thicker ones correspond to higher probabilities of transitioning form one state to another. c) Full transition matrix. d)-g) Factored joint transition matrices, $p(s_t, \lambda_k | s_t - 1)$, for the 4 gestures that have been rewarded.

1. The user draws a continuous stroke with the mouse;
2. The user performs a gesture at the end of which the algorithm is queried to select a label;
3. If the user agrees with the label, then one unit of reward is delivered to the algorithm, otherwise - either no reward or a negative value of reward is delivered;
4. If the user is not satisfied, go to step 1;
5. For each model run a simulation to generate 100 synthetic data sequences;
6. Train a 4-state Hidden Markov Model on the synthetic data set.

The algorithm builds the state space on-line as necessary checking the distance

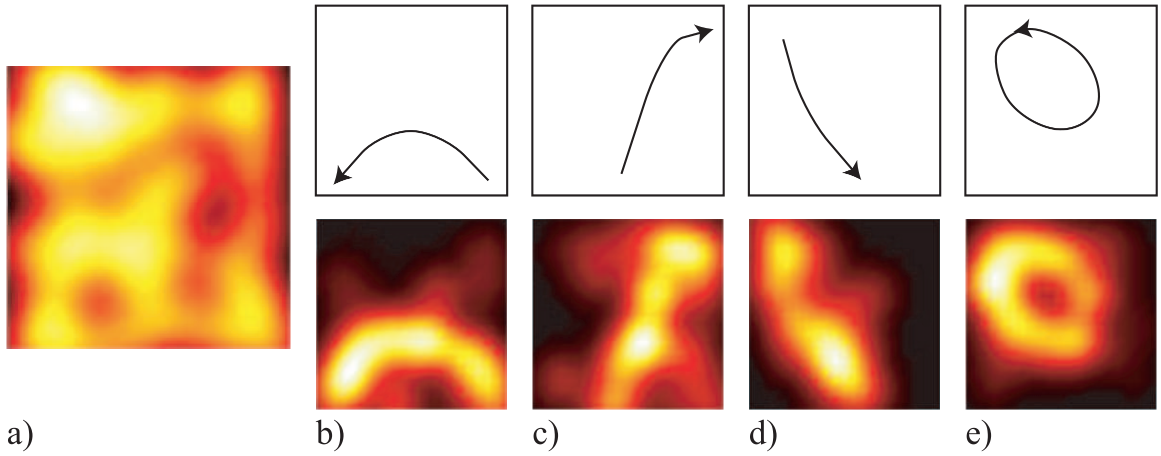


Figure 6-14: a) Contribution of all states to the state space. b)-e) State space factorization. Top row shows the individual gesture models for which the full model was trained. Bottom row shows the state space decomposition with the state membership function.

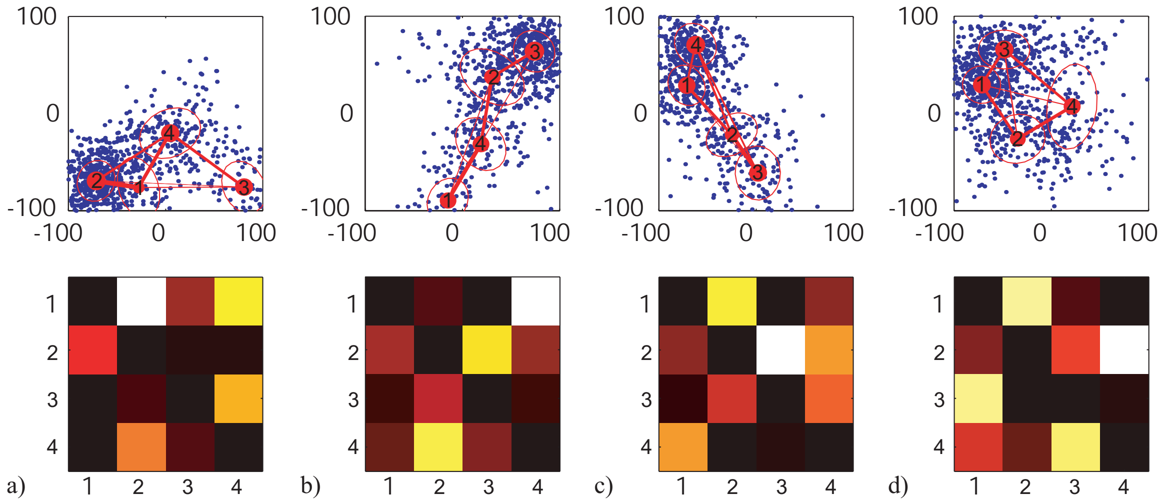


Figure 6-15: Individual HMMs and their transition matrices extracted from the full model by Monte Carlo simulation. (The dominant diagonals have been removed from the matrices to show the transition structure).

between every new sample and existing RBF kernels (eqn. 6.8). The algorithms learns models for 4 gestures, shown in the top row of figure 6-14 b) - e). After traversing each gesture several times and delivering a reward upon each correct guess that the algorithm makes, it arrives to the configuration shown in the figure 6-13a)-c). The figure 6-13 a) shows the resulting state space. Even though no reward or state attribution is reflected in the plot b) graphically showing only state transitions, one can see a strong hint at the structure to which the algorithm has arrived in an unsupervised fashion. After the estimated state membership was used to factor the full transition matrix it arrives to the set of gesture-specific state transition models, which "highlight" parts of the state space relevant to each gesture, shown in figures 6-13 d)-g).

The probability mass in the full model is evenly spread in the input space, as shown in figure 6-14 a). The state membership indicates its estimate of the state contribution to each of the macro-state estimates (bottom row of figure 6-14 b) - e)).

Since the state membership is a probabilistic assignment, all states are still used in each model even though probability of the majority of them to belong to a particular model is very low. These models are used to compute the starting and stopping state probability distributions for each gesture (eqns. 6.32 and 6.30), and sample sequences from resulting gesture models which include all states to fit smaller isolated models to each data set. Resulting 4-state HMMs and their transition matrices are shown in figure 6-15 a)-d).

The full model in this experiment instantiates 31 states with 14 "filler" states (55% state utilization). After the decomposition 4 4-state HMMs were used to model the gestures independently. The figure 6-15 a)-d) shows the resulting HMMs and their transition matrices with the main diagonal removed to expose the transition structure. The structure clearly reflects input gesture shapes.

6.7.2 Vision System

The algorithm is now applied to learning a simple gesture classification system. The input features are computed from a motion average image that allows some small

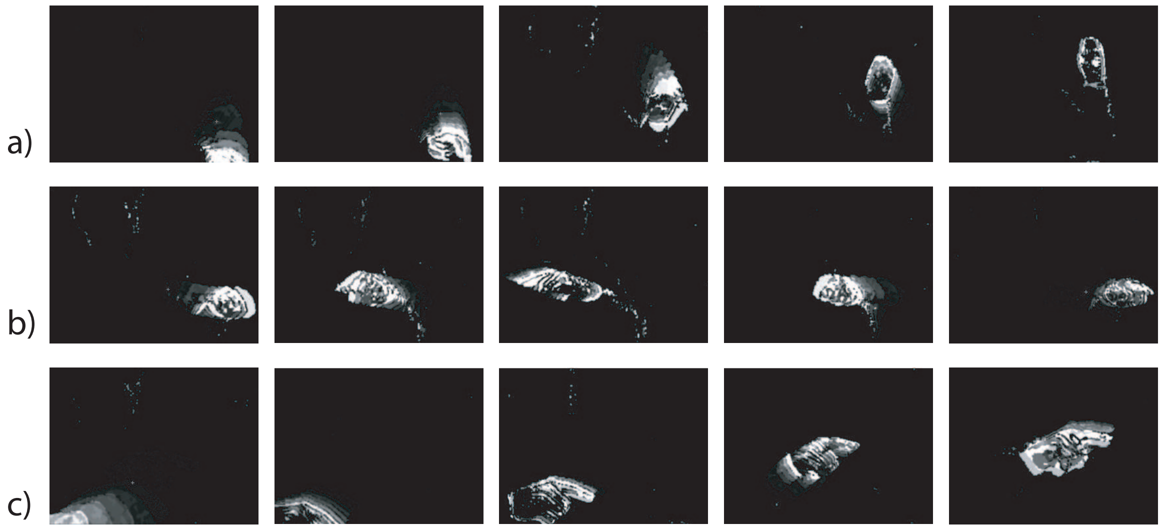


Figure 6-16: a) Attention gesture; b) pointing left with return gesture; c) pointing right gesture.

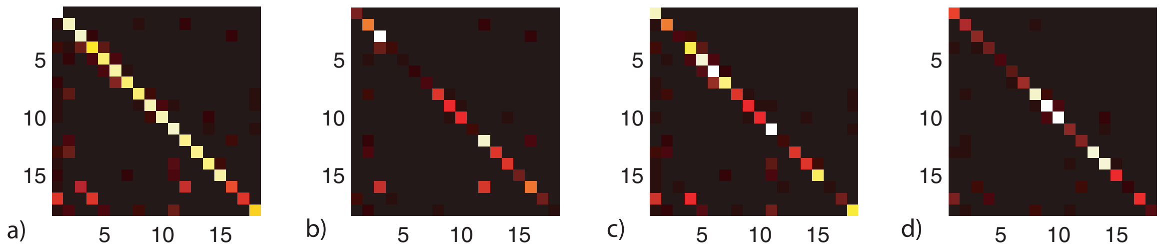


Figure 6-17: a) Transition matrix of the full model. b)-d) Transition matrices conditioned on state membership. At the stage shown in the plot matrices include 18 states.

degree of smoothing of the estimated motion trajectory. For each new image taken at time t, I_t, the motion average image is computed as a decaying running sum ($\alpha = 0.3$):

$$M_t = \alpha|I_t - I_{t-1}| + (1 - \alpha)M_{t-1} \tag{6.33}$$

The weighted mean of pixels in M_t is used as a simple estimate of the hand position. Using this feature as input, the algorithm is trained on-line for a set of three gestures shown in figure 6-16. It needs to be noted that the choice of input features is not extremely robust and is chosen only for convenience of presentation.

After a period of training (usually very short) delivering 5 units of reward for each correct classification the algorithm arrives to a configuration similar to that shown in figure 6-17. In the situation shown in the figure, the model has accumulated 18

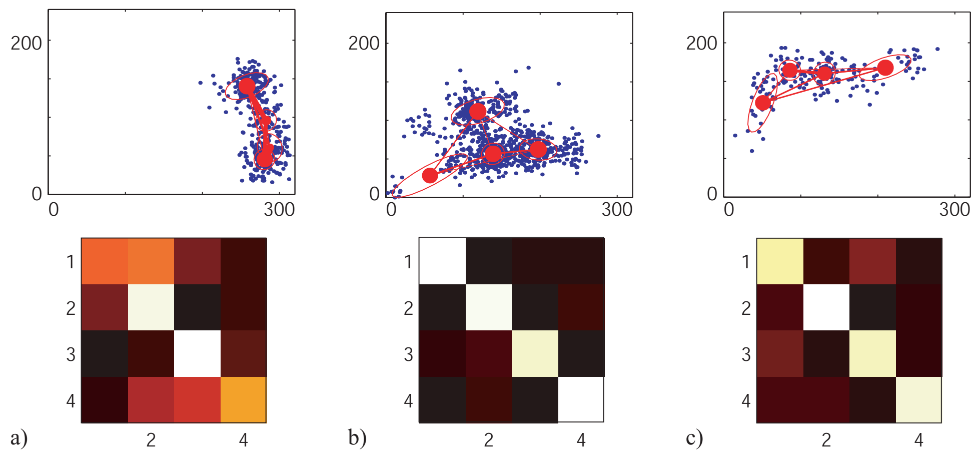


Figure 6-18: Individual extracted HMMs and their transition matrices.

states. The algorithm executes the sampling procedure to extract the set of 4-state HMMs, as shown in in figure 6-18.

The vision system runs on a dual processor 500 MHz Pentium III PC at the rate of 15 frames per second. Isolated gesture models are extracted off-line.

6.8 Discussion

This chapter showed an algorithm that allows for extraction of Markov gesture models from a larger HMM-based structure, estimated on-line. For clarity, results are shown on an intuitively understood set of features (absolute position). It is obvious that the easiest step that will make the system more robust is performing the modeling in the velocity space ([9]). But even with these simple features, the system shows good results in fast on-line learning of gesture models on demand.

The main problem of the algorithm comes from the cornerstone assumption about Markov structure of a gesture. This means, for instance, that a "figure eight" gesture will have problems at the self-intersection point. Even though this particular problem is easily solved with selecting different features (e.g. velocity instead of absolute position), this will remain an inherent problem in general unless some information about history is included with each state.

Chapter 7

Conclusions

The thesis is devoted to a study of algorithms for learning perceptual organization in the context of action selection for autonomous agents. The main bias of the thesis is to frame the problem such that the perceptual organization is directly related to the benefit of its use. The thesis presents three different settings in which the learning takes place:

- short-term on-line memory-based model learning (chap. 3 and 4);
- long-term on-line distribution-based statistical estimation;
- mixed on- and off-line continuous learning of gesture models (chap. 6).

The three methods proceed within essentially the same framework, consisting of a perceptual sub-system and a sub-system that implements the associative mapping from perceptual categories to actions. While the problems solved by each algorithms used in some way, only the Reward-driven Expectation Maximization explored the mapping mechanism fully (chapter 5). Nonetheless, the insight of this framework is that the semantic grounding of perception is derived by the agent in the outcome of the action selection.

In solving real problems of artificial intelligence it is important to look at human and animal learning - not for answers, but for questions. This thesis examines one aspect of learning - learning to find a better perceptual organization under reward as a result of interactions with the environment. This problem arises in the context of skill-related learning, where some frequently used aspects of perception get refined, while others are tuned out.

The main conclusion of the thesis is that in thinking about problems of adaptation of an autonomous system, one needs to think about the learning problem in its entirety. The formulation of the problem, which includes diverse sources of certainties, however insignificant, might make the solution span several areas of what is collectively called machine intelligence, but in a sense it makes the problem easier and the solution more efficient. Indeed, using reward to guide the otherwise unsupervised statistical estimation showed promising results in the experiments in chapter 5, while the problem of continuous learning in chapter 6 would not have even been possible without it. Similarly, the memory-based techniques benefited greatly from being situated in the context of action selection, where reward signal was almost as good as the direct supervision.

7.1 Contributions

This dissertation contributed to several areas, primarily focusing on making the algorithms shown here practical with respect to memory and computation requirements. In the process of developing the main topic, several problems have been encountered and explored in some depth, resulting in contributions listed below.

The main contribution of the thesis is formulating the problem of learning perception under indirect supervision. The setting of the problem allows one to use the reward that the agent receives from the environment to guide the process of learning a better perceptual organization. This formulation allows one to view the problem of perceptual learning as a problem of autonomous learning based on semantical grounding of perception in interactions of the agent with its environment. The outcome of interactions is expressed in the scalar subjective value of reward that the agent receives upon making a decision. Even though the action selection in this work is only associative, more complex representations and mappings could be developed.

The setting of all problems in this dissertation is strictly incremental. At first sight, it should make no difference to the learning algorithms, however, it is not so. The practical aspects of incremental learning and parameter estimation, especially in the initial stages of the algorithm when very little data is available cannot be underestimated. In particular, developing a classification scheme that would assign labels to new data points based on the data that has been observed so far requires additional thought to be given to situations when observed data does not yet contain members of some class. This requires using techniques of exploration, widely used in reinforcement learning, into procedures of otherwise traditional statistical estimation, as was shown in the earlier chapters.

Additionally, solving estimation and classification problems on such impoverished data sets initially rules out using distribution-based techniques, especially in spaces of high dimensionality. Building a learning algorithm for clean slate learning requires examining each sample of the observed data set individually and avoiding summarization of the set by its statistics. Under these circumstances, the computational expense of memory-based techniques can grow unbounded. This dissertation contributes to development of memory-based algorithms by formulating and examining different strategies of making the memorized sample sparse, while maintaining the best possible classification performance (chapter 3). The proposed techniques included a utility-, a margin- and a boundary-based set compression while comparing it to the known risk-based technique.

The problem of utterance classification calls for techniques for sequence analysis. In the presence of a large data sample, statistical techniques can be used to summarize data sets by sequential statistical models, such as Hidden Markov Models. However, the nature of the small sample learning again makes this solution infeasible. Developing a technique for sequence classification based on Support Vector Machine using only the pairwise distortion measures is another contribution developed in this document (chapter 4).

Introducing a solution to the problem of the long-term statistical estimation of perceptual input, this dissertation formulated the idea of weak transduction. The reward in this setting provided a scalar value that evaluated the "quality" of the posterior probability of the model components given the new observation. Including the reward into the problem of density estimation provides less information than direct supervision. Nevertheless, however small, this information provides subjective guidance to the algorithm convergence in the parameter space, as was shown in chapter 5.

Learning from unsegmented gestures provides an additional challenge since the unknown temporal extent of the gesture needs to be recovered. Chapter 6 contributed to the solution of the estimation task by formulating an on-line largely unsupervised estimation of transition dynamics of a sizeable Hidden Markov Model that is subsequently "teased apart" into model dependent representation with the help of a reward. The chapter developed the multi-model Viterbi approximation for online classification using the full model, while not separating the set of state into subsets explicitly. However, to derive a more efficient compact representation of the individual gesture vocabulary, a Monte Carlo sampling was used while solving the problem of finding starting and stopping sub-model states necessary for sampling.

Summary of the specific contributions of this dissertation to on-line state space exploration is shown in the list below:

1. Formulation of the problem of perceptual learning with indirect supervision;
   Identification of a reward that the agent receives from the environment as a result of an interaction can help to guide the process of learning a better perceptual organization.

2. Formulation of developmental classification with compression sets;
   The ideas of finding compression sets on a full sample of the data, as well as formulation of the optimal compression set for the case of a zero Bayes risk has been extended to the area of incremental on-line learning. This extension includes a mechanism for augmenting exploitative techniques with a degree of exploration.

3. Formulation of the margin-based compression rule;

The data in memory-based state representations does not have an equal value for the task of classification a new sample. The margin-based rule is based on an observation that in discriminative setting the data located away from the class margin is not used for classification. The rule results in finding a sparse subset of the samples in the memory model leading to significant memory and computational savings.

4. Formulation of the boundary-based compression rule;

   Similarly to the margin classification rule, the bound set compression rule discards the data located inside the class model. The advantages of this rule include its good performance even in cases where class margins cannot be estimated well in the initial stages of learning for classes that still have no representation in the model.

5. Development of a technique of Support Sequence classification;

   The contribution of the thesis includes the development of a technique for classification sequential data with Support Vector Machines. The Support Vector - based algorithm embeds a dynamic programming algorithm inside the Kernel calculations. That allows to treat sequences of variable length within the SVM framework directly without re-sampling.

6. Introduction of the idea of Weak Transduction;

   The dissertation introduces a notion of Weak Transduction and presents a solution to this problem in the context of perceptual learning under indirect supervision. Weak transduction emphasizes that the information given to the classifier in form of the reward is weak, yet contains information that can be utilized.

7. Exploration of incremental Reward-driven Expectation Maximization;

   The problem of weak transduction is solved in this thesis within the Expectation-Maximization framework as a part of perceptual learning under reward. The advantage of the technique is that an outside observer can use it to direct the learning process of an agent with naïve feedback.

8. Development of an algorithm for on-line reward-driven unsegmented gesture learning algorithm;
   The contribution of the last technical chapter of the thesis is in the development of a practical algorithm of a clean-slate learning from an unsegmented gestural input. The proposed model can quickly be trained to identify a gesture from a fixed set. The essence of the contribution is that the learning proceeds with a largely unsupervised model while using the feedback from the trainer to find a decomposition of the model dynamics into a set of independent models.

9. Development of the multi-model Viterbi approximation;
   The thesis contributed by developing and algorithm of gesture learning based on a well known Viterbi approximation. The Viterbi algorithm is modified to run simultaneously on a set of models approximated in a maximum entropy sense on demand.

10. Solution for the Monte-Carlo technique for compact vocabulary extraction.
    Finally, the thesis develops a solution for extraction of a set of independent gesture models from the full multi-gesture representation used for structural learning. This allows, when convenient, to replace the larger and less efficient model with a set of smaller and efficient ones to proceed with further model refinement.

7.2 Further Work

In this work many problems of autonomous learning were left unexplored. The main focus of this dissertation was devoted to the setting of associative learning where a) reward was immediate, albeit rare; and b) no state transitions were involved in policy estimation. To make the application of the ideas of this thesis more general one needs to address two problems: firstly, how to perform state discovery under delayed reward; and secondly, how to do it in the setting of the full reinforcement task. These two problems are both interesting and challenging, requiring more work to find an efficient solution.

Another issue, which is perhaps the most promising is combining the techniques of this dissertation with similar approaches to learning action space of the agent. Again, two problems exist and are only partially addressed in this thesis - the problem of action bias and the problem of learning in the action space.

Action bias. First, in all algorithms shown here in the beginning of the learning process all actions are presumed equally likely. In the face of a large set of actions, this can make training extremely inefficient and frustrating. In animal training this problem is solved by building up the action preference in a setting which is close to fully supervised. Then, gradually, the reward structure and the input are changed to facilitate building of the association between an observation, an action and a reward. In the system where the utterance learning algorithm was used, it was, in fact, being done outside of the utterance learning algorithm. Inclusion of the algorithm for building the action prior could significantly benefit its usability.

Learning in the action space. The second issue is related to exploration in the action space. In this dissertation all actions were presumed existing and no new skills were acquired by any of the algorithms during training. Again, this would not be extremely satisfying in a complete system. During training actions can and should be built up from some primitive representations. It can be done within the framework of presented algorithms by about the same means as the perceptual learning. One can imagine moving a robot arm to produce some meaningful action while estimating patterns of control voltages necessary to reproduce it. Here, again, the reward could help to separate the meaningful changes in the parameters of the arm from the transient ones in much the same way as it was done in perception. Furthermore, shaping actions and shaping perception could be done alongside to cross-validate formation of action and perceptual primitives.

Learning hierarchical sequence models. The algorithm of the last chapter extracts gesture models from an unsegmented sequential input with help of the reward. It is an interesting starting point for learning hierarchical models that initially sparked my interest in the topic of this dissertation. As a part of the motivation I showed two systems built on the basis of a hierarchical grammatical representation. These representations are possible to learn once a good set of the primitive components is discovered. This problem is made more difficult in the setting of this thesis when the amount of training data is very small.

There might be a number of different ways of attacking this problem. Using a bit of a speculation one might imagine approaching it from these directions:

1. Partitioning the training problem

   As a general note, the learning problem is made much easier by the presence of the trainer, whose goal it is to constrain the search space for the subject of training. One might imagine constructing a training situation such that reliable and accurate repetitions of the structural occurrences are provided to the agent;

2. Imposing structure

   As was stated in chapter 6, no switching dynamics were imposed on the gesture model, λ. In order to discover structure and hierarchy one would need to keep track of the macro-state transitions. In fact this is the simplest thing to do since gesture models are not unsupervised and are uniquely determined by the presence of the reward. Then one would only need to keep the transition statistics (with some continuity constraints) on the change in the value of λ as indicated by the reward independently from the input observations. Then, as more data becomes available, the switching model can contribute to the inference, much the same way as [58] propose;

3. Reinforcement learning solution

   The most obvious solution to the problem, once primitives are extracted, would be to use the posterior probability distribution over them as a belief state for the full-scale reinforcement learning technique. Some tree-based state representations (for instance, [50]) lend themselves to inducing the utile structure;

4. Model merging

   A technique of Bayesian model merging (see [71]) can be employed once the primitive components are extracted from the full input model. It is convenient to talk about the hierarchy in terms of a context-free grammar. In fact, learning hierarchical sequential representation is equivalent to inducing (at least) the context-free grammar. With this in mind, for any sequence of primitives the grammatical structure can be induced from the model transition counts when incrementally, a *chunking* operation is performed - for the highest co-occurrence pair postulate a non-terminal and form a re-write rule. The chunking operation is in a sense equivalent to establishing the length of the historical record of model transitions from the reinforcement learning solution from above. Typically, in learning from data, a *merging* operation is required in order to discover merged models (for instance, a model of an utterance "Stay" as a combination of a male speaker and a female speaker models). In the setting of this dissertation such a generalization is extracted from the policy;

5. Iterative refinement

   Perhaps a more immediate solution to the hierarchical learning problem from unconstrained on-line observations is the iterative refinement of structural hypotheses. The idea is to use full model as well as the extracted primitive set in order to learn the hierarchical structure. As a first step one would extract the independent primitives. Using these primitives one would produce several competing hypotheses about some aspect of local structure. At the next step the inference on the full model is done using the structure hypotheses and the winning ones would move to the next round. The process would repeat while necessary.

In conclusion, the problem domain of learning in the action-perception cycle turned out to be interesting and rich. Much more work is required to solidify the concepts explored in this dissertation. I can only hope to have provided ideas that can be extended and turned into practice.

Appendix A

Nearest Neighbor Performance Bounds

A.1 Asymptotic Bound

This section summarizes the derivation of the asymptotic bound as presented in Duda et al. [23]. The asymptotic analysis of a nearest neighbor procedure can be done starting with estimating Bayes risk in the simplest case of a single neighbor two-class classification problem. For the binary classification Bayes Risk, r^*, is a probability of misclassification of a query x^n. In order to classify a sample x^n into one of the classes C_1 or C_2, a nearest neighbor, $x_{(1)}$ needs to be found. Then x^n is assigned the to the class of $x_{(1)}$. An error occurs when the true label of x^n does not coincide with the label of $x_{(1)}$, that is:

$$
\begin{aligned}
r^* &= P\left((x^n \in C_1 \text{ AND } x_{(1)} \in C_2) \text{ OR } (x^n \in C_2 \text{ AND } x_{(1)} \in C_1)|x^n, x_{(1)}\right) \\
&= P(C_1|x^n)P(C_2|x_{(1)}) + P(C_2|x^n)P(C_1|x_{(1)})
\end{aligned}
\tag{A.1}
$$

The asymptotic behavior of the classifier can now be analyzed from the observation that when the amount of data is large, the nearest neighbor $x_{(1)}$ is so close to x^n that $P(C_i|x_{(1)})$ can be replaced with $P(C_i|x^n)$, and as a result, the error rate of a 1-NN classifier, $r^{(1)}$ becomes:

$$r^{(1)} = 2P(C_1|x^n)P(C_2|x^n) \tag{A.2}$$

Extending this reasoning to multi-class case, with K classes, the conditional risk can be written as follows:

$$\begin{aligned} r^{(1)} &= P(C_1|x^n)\sum_{i=1;i\neq 1}^{K} P(C_i|x^n) + \ldots + P(C_2|x^n)\sum_{i=1;i\neq K}^{K} P(C_i|x^n) \\ &= \sum_{i=1}^{K} P(C_i|x^n)\left[1 - P(C_i|x^n)\right] = 1 - \sum_{i=1}^{K} P(C_i|x^n)^2 \end{aligned} \tag{A.3}$$

On the other hand, $r^* = 1 - \max_i \{P(C_i|x^n)\} = P(C_\ell|x^n)$. From this and an application of the Schwartz inequality it follows that:

$$(K-1)\sum_{i=1;i\neq\ell}^{K} P(C_i|x^n)^2 \geqslant \left[\sum_{i=1;i\neq\ell}^{K} P(C_i|x^n)\right]^2 = \left[1 - P(C_\ell|x^n)\right]^2 = r^{*2} \tag{A.4}$$

Adding $(K-1)P(C_\ell|x^n)^2$ to both sides:

$$(K-1)\sum_{i=1}^{K} P(C_i|x^n)^2 \geqslant r^{*2} + (K-1)\left[1 - r^*\right]^2 \tag{A.5}$$

Substitution of A.5 into A.3 gives,[13]:

$$r^* \leqslant r^{(1)} \leqslant 2r^* - \frac{K}{K-1} r^{*2} \tag{A.6}$$

Computing expectation of both sides with respect to x we get the bound for the error rate, $L = E_x\left[r\right]$.

$$L^* \leqslant L^{(1)} \leqslant 2L^* - \frac{K}{K-1} L^{*2} \tag{A.7}$$

A.2 Finite Sample Bound

The main question regarding the nearest neighbor classification schemes is its rate of convergence. What can be said about the rate at which they achieve their limit performance? Unfortunately, to this day the answer is - absolutely nothing. In fact, it can be formally shown that in general, no universally best distribution-free classifiers exist (e.g. see [18]).

One can attempt to approach the problem from a different point of view - by examining error bounds achievable by a particular classification rule on a finite sample of data. Unfortunately, even though with some work the bounds can be formulated, they remain fairly loose. To get a feel of this in the general case, consider the following nearest neighbor approximation bound, given by Stone, [72]:

Lemma A.2.1 (Stone, 1977). *For any integrable function f, and n and any $k \leq n$,*

$$\sum_{i=1}^{k} E\{|f(X_{(i)}(X))|\} \leq k\gamma_d E\{|f(X)|\}$$

where $\gamma_d \leq \left(1 + 2/\sqrt{2-\sqrt{3}}\right)^d - 1$ depends upon the dimension only.

Excitingly, this lemma essentially states that the difference between expected value of a function and that estimated empirically from a set of nearest neighbors differs at most by a factor of γ_d, depending on dimensionality only. Unfortunately, the numerical value of $\gamma_1 = 3.8637, \quad \gamma_2 = 22.6556, \quad \gamma_3 = 114.0539, ...$ leaves a lot to be desired.

A.3 VC Bound on a Condensed Classifier

This section outlines the argument about the VC bound on the performance of a condensed nearest neighbor classifier based on the presentation given in Devroye et al., [18].

One can attempt to find a bound on a simple condensed nearest neighbor classifier from the point of view of VC theory. This is accomplished in several steps:

Theorem A.3.1 (Vapnik and Chervonenkis, 1971). *For any probability measure ν and class of sets $\mathcal{A}$, and for any n and $\varepsilon > 0$,*

$$P\left\{\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| > \varepsilon\right\} \leqslant 8s\left(\mathcal{A}, n\right) e^{-n\varepsilon^2/32}$$

where $s\left(\mathcal{A}, n\right)$ is the shatter coefficient of a family of classifiers $\mathcal{A}$[1]. To compute the shatter coefficient on which the VC bound of the theorem A.3.1 depends, a Voronoi partitioning of a space induced by a set of m points needs to be considered. The shatter coefficient of a set of a family consisting of up to m Voronoi cells $\mathcal{A}_m$ can be bounded by a set union bound:

$$s(\mathcal{A}_m, n) \leq s(\mathcal{A}_1, n)^m$$

Each cell is an intersection of $m - 1$ n-dimensional half-spaces. The shattering power of such family can again be bounded by the set intersection bound:

$$s(\mathcal{A}_1, n) \leq s(\mathcal{P}, n)^{(m-1)}$$

where $\mathcal{P}$ is a family of half-spaces. With the help of the Sauer's lemma the shatter coefficient of a half space can be bounded by an expression which only includes the VC dimension and a number of the data points, n:

$$s(\mathcal{P}, n) \leq \sum_{i=0}^{V_\mathcal{P}} \binom{n}{i} \leq \left(\frac{ne}{V_\mathcal{P}}\right)^{V_\mathcal{P}}$$

by Sauer's lemma, where $V_\mathcal{P}$ is the VC dimension of $\mathcal{P}$;

$$V_\mathcal{P} = d + 1$$

after Vapnik [80], where d is the dimensionality of the feature space. Collecting the above we get:

[1]A shatter coefficient is the maximum number of different subsets of n points that can be picket out by the family **A**

$$s(\mathcal{A}_m, n) \leq \left(\frac{ne}{(d+1)}\right)^{(m-1)m(d+1)}$$

Using $L_n(A)$ to denote an error probability of a classifier $A \in \mathcal{A}$ and $\widehat{L_n}(A)$ for its empirical estimate, the error can be bounded by the "worst case": $|L_n(A) - \widehat{L_n}(A)| \leq \sup_{A \in \mathcal{A}} |\nu_n(A) - \nu(A)|$, where $\nu(A)$ is the error rate of A and $\nu_n(A)$ is its empirical estimate on n data points. Substituting the result from above into the theorem A.3.1 proves the following:

Theorem A.3.2 (Devroye and Wagner, 1979). *For all $\varepsilon > 0$ and all distributions,*

$$P\left\{|L_n(A) - \widehat{L_n}(A)| \geqslant \varepsilon\right\} \leqslant 8\left(\frac{ne}{d+1}\right)^{(m-1)m(d+1)} e^{-n\varepsilon^2/32}$$

which is currently the best one can assert about the performance of a condensed nearest neighbor classifier. This bound is not particularly tight, allowing only to say that the performance improves in probability as m grows.

Appendix B

Divergence for Gaussian Densities

This section derives a closed form expression for the divergence between two gaussian distributions given by their sufficient statistics, $\{\mu_i, \Sigma_i\}$. This expression is useful for evaluation of divergence between two Gaussian mixture distributions.

$$
\begin{aligned}
&D(p||q) = \int p(\mathbf{x}) \log \left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x} = \\
&\int p(\mathbf{x}) \left(\log p(\mathbf{x}) - \log q(\mathbf{x})\right) d\mathbf{x} = \\
&\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} - \int p(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} = \\
&-H(p(\mathbf{x})) + C(p(\mathbf{x}), q(\mathbf{x}))
\end{aligned}
\tag{B.1}
$$

where $H(p(\mathbf{x}))$ is the entropy and $C(p(\mathbf{x}), q(\mathbf{x}))$ is the cross-entropy. Substituting explicit expressions for Gaussian densities for $p(\mathbf{x})$ and $q(\mathbf{x})$ into the expression for the entropy, $H(p(x))$:

$$
\begin{aligned}
&H(p(\mathbf{x})) = \int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} = \\
&\int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} \\
&\quad \times \log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)}\right] d\mathbf{x} = \\
&\int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} \\
&\quad \times \left\{ \log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - -\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p) \right\} d\mathbf{x} = \\
&\int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} \log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] d\mathbf{x} - \\
&\quad -\frac{1}{2} \int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} (\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p) d\mathbf{x} = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] \int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} d\mathbf{x} - \\
&\quad -\frac{1}{2} \int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} (\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p) d\mathbf{x} = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2} E_p \left[(\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right] = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2} E_p \left[tr\left((\mathbf{x}-\mu_p)^{\mathbf{T}} \mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)\right] = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2} E_p \left[tr\left(\mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)(\mathbf{x}-\mu_p)^{\mathbf{T}}\right)\right] = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2} E_p \left[tr\left(\mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_p\right)\right] = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2} E_p \left[tr\left(\mathbf{I}\right)\right] = \\
&\log \left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{d}{2} = \\
&-\log \left[(2\pi e)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}\right]
\end{aligned}
\tag{B.2}
$$

Similarly, for the cross-entropy term:

$$
\begin{aligned}
&C(p(\mathbf{x}), q(\mathbf{x})) = \int p(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} = \\
&\int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}}\mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} \\
&\quad \times \log\left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_q)^{\mathbf{T}}\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_q)\right)}\right] d\mathbf{x} = \\
&\int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}}\mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} \\
&\quad \times \left\{\log\left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}}\right] - -\frac{1}{2}(\mathbf{x}-\mu_q)^{\mathbf{T}}\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_q)\right\} d\mathbf{x} = \\
&\log\left[\frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}}\right] \int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}}\mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} d\mathbf{x} - \\
&\quad -\frac{1}{2} \int \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_p\right|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_p)^{\mathbf{T}}\mathbf{\Sigma}_p^{-1}(\mathbf{x}-\mu_p)\right)} (\mathbf{x}-\mu_q)^{\mathbf{T}}\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_q) d\mathbf{x} = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[(\mathbf{x}-\mu_q)^{\mathbf{T}}\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_q)\right] = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] \\
&\quad -\frac{1}{2} E_p\left[((\mathbf{x}-\mu_p)-(\mu_q-\mu_p))^T\mathbf{\Sigma}_q^{-1}((\mathbf{x}-\mu_p)-(\mu_q-\mu_p))\right] = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[(\mathbf{x}-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_p)\right. \\
&\quad \left. -2(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_p) + (\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p)\right] = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[tr\left((\mathbf{x}-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_p)\right)\right] \\
&\quad -E_p\left[(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_p)\right] + \frac{1}{2} E_p\left[(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p)\right] = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}(\mathbf{x}-\mu_p)(\mathbf{x}-\mu_p)^T\right)\right] \\
&\quad -(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(E_p[\mathbf{x}]-\mu_p) + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right)\right] - (\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_p-\mu_p) + \\
&\quad +\frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) = \\
&-\log\left[(2\pi)^{d/2} \left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \frac{1}{2} E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right)\right] + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p)
\end{aligned}
\tag{B.3}
$$

Substitution of the above results into the equation B.1 gives:

$$
\begin{aligned}
&D(p||q) = \log\left[(2\pi e)^{d/2}\left|\mathbf{\Sigma}_p\right|^{1/2}\right] - \log\left[(2\pi)^{d/2}\left|\mathbf{\Sigma}_q\right|^{1/2}\right] - \\
&-\frac{1}{2}E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right)\right] + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) = \\
&\log\left[\frac{(2\pi e)^{d/2}\left|\mathbf{\Sigma}_p\right|^{1/2}}{(2\pi)^{d/2}\left|\mathbf{\Sigma}_q\right|^{1/2}}\right] - \frac{1}{2}E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right)\right] + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) = \\
&\log\left[\frac{e^{d/2}\left|\mathbf{\Sigma}_p\right|^{1/2}}{\left|\mathbf{\Sigma}_q\right|^{1/2}}\right] - \frac{1}{2}E_p\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right)\right] + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) =
\end{aligned}
\tag{B.4}
$$

To avoid assymetry in the measure of divergence, we use the following:

$$
\begin{aligned}
&S(p||q) = \frac{1}{2}\left[D(p||q) + D(q||p)\right] = \\
&\frac{1}{2}\left[\log\left[\frac{e^{d/2}\left|\mathbf{\Sigma}_p\right|^{1/2}}{\left|\mathbf{\Sigma}_q\right|^{1/2}}\right] - \frac{1}{2}tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right) + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p) + \right] + \\
&+\frac{1}{2}\left[\log\left[\frac{e^{d/2}\left|\mathbf{\Sigma}_q\right|^{1/2}}{\left|\mathbf{\Sigma}_p\right|^{1/2}}\right] - \frac{1}{2}tr\left(\mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q\right) + \frac{1}{2}(\mu_p-\mu_q)^T\mathbf{\Sigma}_p^{-1}(\mu_p-\mu_q)\right] = \\
&\frac{1}{2}\left[\log\left[\frac{e^{d/2}\left|\mathbf{\Sigma}_p\right|^{1/2}}{\left|\mathbf{\Sigma}_q\right|^{1/2}}\frac{e^{d/2}\left|\mathbf{\Sigma}_q\right|^{1/2}}{\left|\mathbf{\Sigma}_p\right|^{1/2}}\right]\right] \\
&\qquad +\frac{1}{2}\left[-\frac{1}{2}tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right) + \frac{1}{2}(\mu_q-\mu_p)^T\mathbf{\Sigma}_q^{-1}(\mu_q-\mu_p)\right. \\
&\qquad\qquad \left.-\frac{1}{2}tr\left(\mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q\right) + \frac{1}{2}(\mu_p-\mu_q)^T\mathbf{\Sigma}_p^{-1}(\mu_p-\mu_q)\right] \\
&\frac{1}{2}\left[d - \frac{1}{2}\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p\right) + tr\left(\mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q\right)\right]\right. \\
&\qquad \left.+\frac{1}{2}(\mu_q-\mu_p)^T(\mathbf{\Sigma}_q^{-1}+\mathbf{\Sigma}_p^{-1})(\mu_q-\mu_p)\right] = \\
&\frac{1}{2}\left[d - \frac{1}{2}\left[tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p + \mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q\right)\right] + \frac{1}{2}(\mu_q-\mu_p)^T(\mathbf{\Sigma}_q^{-1}+\mathbf{\Sigma}_p^{-1})(\mu_q-\mu_p)\right] \\
&\frac{1}{4}\left[d - tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p + \mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q\right) + (\mu_q-\mu_p)^T(\mathbf{\Sigma}_q^{-1}+\mathbf{\Sigma}_p^{-1})(\mu_q-\mu_p)\right] = \\
&\frac{1}{4}\left[(\mu_q-\mu_p)^T(\mathbf{\Sigma}_q^{-1}+\mathbf{\Sigma}_p^{-1})(\mu_q-\mu_p) - tr\left(\mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p + \mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q - 2\mathbf{I}\right)\right]
\end{aligned}
\tag{B.5}
$$

For a lack of a better analytical method, the latter expression is used to compute the divergence between all pairs of components of two Gaussian mixture density. The mixture-to-mixture mapping that achieves the minimum divergence is chosen.

Bibliography

[1] F. Bedford. Perceptual learning. In D. Medin, editor, *The Psychology of Learning and Motivation*, volume 30, pages 1–60. Academic Press, 1993.

[2] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. A support vector method for clustering. In Todd K. Leen, Thomas G. Dietterich, and Voler Tresp, editors, *Advances in Neural Information Processing Systems*, pages 367–373. MIT Press, 2000.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] Christopher M. Bishop. Latent variable models. In Michael I. Jordan, editor, *Learning in graphical models*. MIT Press, Cambridge, MA, 1998.

[5] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[6] A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *Proc. Comp. Vis. and Pattern Rec.*, pages 196–202, Santa Barbara, CA, 1998.

[7] Aaron F. Bobick. *Natural Object Characterization*. PhD thesis, MIT, 1987.

[8] Denis K. Burnham, Lynda J. Earnshaw, and John E. Clark. Development of categorical identification of native and non-native bilabial stops: Infants, children and adults. *Journal of Child Language*, 18 (2):231–260, 1991.

[9] L. W. Campbell, D. A. Becker, A. J. Azarbayejani, A. F. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. In *Second International*

Conference on Face and Gesture Recognition, pages 157–162, Killington, VT, 1996.

[10] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, pages 409–415. MIT Press, 2000.

[11] C Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 26:1179–1184, 1974.

[12] Vladimir Cherakassky and Filip Mulier. *Learning from Data. Concepts, Theory and Methods.* Wiley, New York, 1998.

[13] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[14] T. J. Darrell and A. P. Pentland. Space-time gestures. *Proc. Comp. Vis. and Pattern Rec.*, pages 335–340, 1993.

[15] P. Dayan and T. Long. Statistical models of conditioning. In *NIPS 10*, Denver, CO, 1998.

[16] N.M. Dempster, A.P. Laird and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:185–197, 1977.

[17] P. Devijver and J. Kittler. On the edited nearest neighbor rule. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pages 72–80, Los Alamitos, CA, 1980. Pattern Recognition Society.

[18] Luc Devroye, Lázsló Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.

[19] Marco Dorigo and Marco Colombetti. *Robot Shaping. An Experiment in Behavior Engineering.* MIT Press, Cambridge, MA, 1998.

[20] B.A. Dosher and Z. L. Lu. Mechanisms of perceptual learning. *Vision Research*, 39(19):3197–3221, 1999.

[21] F. Dretske. Seeing, believing, and knowing. In D. N. Osherson, S. M. Kosslyn, and J. M. Hollerbach, editors, *Visual Cognition and Action, V.2*, pages 129–148. MIT Press, Cambridge, MA, 1990.

[22] Itiel E. Dror, Stephen M. Kosslyn, and Wayne L. Waag. Visual-spatial abilities of pilots. *Journal of Applied Psychology*, 78 (5):763–773, 1993.

[23] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2000.

[24] S. Edelman and H. H. Bülthoff. Orientation dependence in the recognition of familiar and novel views of 3d objects. *Vision Research*, 32:2385–2400, 1992.

[25] Shimon Edelman. Visual perception. In S. Shapiro, editor, *Encyclopedia of Artificial Inelligence*, pages 1655–1663. Wiley, 1991.

[26] I. Fine and Robert A. Jacobs. Perceptual learning for pattern discrimination task. *Vision Research*, 40:3209–3230, 2000.

[27] Sally Floyd and Manfred Warmuth. Sample compression, learnability, and Vapnik-Chervonenkis dimension. Technical Report UCSC-CRL-93-13, University of California, Santa Cruz, 1993.

[28] Basic Behavioral Science Research for Mental Health. A national investment. A report of the national advisory mental health council. Technical Report 96-3682, NIH, 1995.

[29] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[30] C.R. Gallistel and John Gibbon. Time, rate and conditioning. *Psychological Review*, 107:289–344, 2000.

[31] S. Geva and J. Sitte. Adaptive nearest neighbor pattern clasification. *IEEE Transactions onNeural Networks*, 2:318–322, 1991.

[32] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities. In *Comp. Vis. and Pattern Rec.*, pages 22–29, Santa Barabara, CA, 1998. IEEE.

[33] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[34] Pengyu Hong, Matthew Turk, and Thomas S. Huang. Constructing finite state machines for fast gesture recognition. In *15th International Conference on Pattern Recognition*, Barcelona, Spain, 2000.

[35] David M. Howard, Stuart Rosen, and Victoria Broad. Major/minor triad identification and discrimination by musically trained and untrained listeners. *Music Perception*, 10 (2):205–220, 1992.

[36] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 22(8), August 2000.

[37] Y. A. Ivanov, C. Stauffer, A. F. Bobick, and W. E. L. Grimson. Video surveillance of interactions. In *CVPR'99 workshop on Video Surveillance*, Ft. Collins, CO, 1999.

[38] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*. MIT Press, 1998.

[39] Frederick Jelinek. *Statistical Methods for Speech Recognition*. Language, Speech, an Communication series. MIT Press, Cambridge, Massachusetts, 1999.

[40] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer, New York, NY, 1996.

[41] Michael I. Jordan. *Learning in Graphical Models.* MIT Press, Cambridge, MA, 1998.

[42] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[43] L.P. Kaelbling, L.M. Littman, and A.W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[44] M. Kudo, J. Toyama, and M. Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20:(11–13):1103–1111, 1999.

[45] A. Likas. Reinforcement learning approach to online clustering. *Neural Computation*, 11:1915–1932, 1999.

[46] S. R. Lindsay. *Applied Dog Training.* Iowa State University Press, Ames, Iowa, 2000.

[47] Z. Liu and D. Weinstall. Mechanisms of generalization in perceptual learning. In *Advances in Neural Information Processing Systems*, 1999.

[48] Gábor Lugosi. On concentration-of-measure inequalities. Seminar notes, 1998.

[49] Michael P. Lynch, Rebecca E. Eilers, and Marc H. Bornstein. Speech, vision, and music perception: Windows on the ontogeny of mind. *Psychology of Music, Special Issue: Child development and music*, 20 (1):3–14, 1992.

[50] A.K. McCallum. *Reinforcement learning with selective perception and hidden state.* Ph.d., University of Rochester, Rochester, NY, 1995.

[51] Herzog M.H. and M. Fahle. A recurrent model for perceptual learning. *Journal of Optical Technology*, 66:836–841, 1999.

[52] A. Mignault and A. A. J. Marley. A real-time neuronal model of classical conditioning. *Adaptive Behavior*, 6(1):3–61, 1997.

[53] Thomas P. Minka. Expectation-maximization as lower bound maximization. Tutorial note, 1999.

[54] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, Cambridge, MA, 1998.

[55] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[56] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: An application to face detection. In *CVPR*, Puerto Rico, 1997.

[57] I. P. Pavlov. *Conditioned Reflexes*. Oxford University Press, Oxford, UK, 1927.

[58] Vladimir Pavlović and James M. Rehg. Impact of dynamic model learning on classification of human motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 788–795, Hilton Head, SC, 2000. IEEE.

[59] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[60] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, Cambridge, UK, 2000.

[61] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice Hall, Englewood Cliffs, 1993.

[62] R. Redner and H. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[63] R. A. Rescorla and A. R. Wagner. A theory of pavlovian conditioning. variations in the effectiveness of reinforcement and non-reinforcement. In A. H. Black and W. F. Proskay, editors, *Classical Conditioning*, volume 2, pages 54–99. Appleton-Century-Crofts, New York, 1972.

[64] Deb Roy and Alex Pentland. Multimodal adaptive interfaces. Technical Report 438, MIT Media Laboratory, Perceptual computing Section, 1997.

[65] Max Rudolf. *The Grammar of Conducting. A Comprehensive Guide to Baton Techniques and Interpretation*. Schimmer Books, New York, 1994.

[66] Philip N. Sabes and Michael I. Jordan. Reinforcement learning by probability matching. In *NIPS 8*, 1996.

[67] Nestor A. Schmajuk. *Animal Learning and Cognition: A Neural Network Approach*. Cambridge University Press, New York, 1997.

[68] E. S. Spelke. Origins of visual knowledge. In D. N. Osherson, S. M. Kosslyn, and J. M. Hollerbach, editors, *Visual Cognition and Action, V.2*, pages 99–128. MIT Press, Cambridge, MA, 1990.

[69] A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California at Berkeley, 1994.

[70] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.

[71] A. Stolcke and S. Omohundro. *Inducing Probabilistic Grammars by Bayesian Model Merging. In Grammatical Inference and Applications*, pages 106–118. Springer, 1994.

[72] C. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5:595–645, 1977.

[73] R. S. Sutton. Learning to predict by the method of temporal difference. *Machine Learning*, 3:9–44, 1988.

[74] R.S. Sutton and A.G Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA, 1998.

[75] M. Tarr and S. Pinker. Mental rotation and orientation-dependence in shape recognition. *Cognitive Psychology*, 21:233–282, 1989.

[76] M. A. L. Thathachar and P. S. Sastry. A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, 15:168–175, 1985.

[77] E. Thorndike. *Animal Intelligence*. Hafner, Darien, CT, 1911.

[78] Keiji Uchikawa and Robert M. Boynton. Categorical color perception of japanese observers: Comparison with that of americans. *Vision Research*, 27 (10):1825–1833, 1987.

[79] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[80] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[81] Andrew J. Viterbi. Error bounds for convolutionalcodes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[82] C. J. C. H Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[83] Chris Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Department of Computer Science, Royal Halloway, University of London, 1999.

[84] J. Weng, C. Evans, W. S. Hwang, and Y. B. Lee. The developmental approach to artificial intelligence: Concepts, developmental algorithms and experimental results. In *NSF Design and Manufacturing Grantees Conference*, Queen Mary, Long Beach, CA, January 5-8, 1999.

[85] Ronald J. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. In *IEEE First International Conference on Neural Networks*, San Diego, CA, 1987.

[86] A. D. Wilson. *Adaptive Models for the Recognition of Human Gesture*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2000.

[87] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI–TR–95–02–010, Santa Fe Institute, 1995.

[88] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[89] Lei Xu and Michael I. Jordan. On convergence properties of the EM algorithm for gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[90] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695. MIT Press, 2000.

[91] Heinrich Zollinger. Categorical color perception: Influence of cultural factors on the differentiation of primary and derived basic color terms in color naming by japanese children. *Vision Research*, 28 (12):1379–1382, 1988.