Action-Reaction Learning: Analysis and Synthesis of Human Behaviour

by

Tony Jebara

B.Eng., Electrical Engineering
McGill University, Montreal, Canada
June 1996

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE IN MEDIA ARTS AND SCIENCES
at the
Massachusetts Institute of Technology
May 1998

Signature of Author ______________________________

Program in Media Arts and Sciences
15 May 1998

Certified by ______________________________

Alex P. Pentland
Academic Head and Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by ______________________________

Stephen A. Benton
Chair
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Action-Reaction Learning: Analysis and Synthesis of Human Behaviour

by
Tony Jebara

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

Master of Science in Media Arts and Sciences

Abstract

I propose Action-Reaction Learning as an approach for analyzing and synthesizing human behaviour. This paradigm uncovers causal mappings between past and future events or between an action and its reaction by observing time sequences. I apply this method to analyze human interaction and to subsequently synthesize human behaviour. Using a time series of perceptual measurements, a system automatically uncovers a mapping between past gestures from one human participant (an action) and a subsequent gesture (a reaction) from another participant. A probabilistic model is trained from data of the human interaction using a novel estimation technique, Conditional Expectation Maximization (CEM). The estimation uses general bounding and maximization to find the maximum conditional likelihood solution. The learning system drives a graphical interactive character which probabilistically predicts the most likely response to a user's behaviour and performs it interactively. Thus, after analyzing human interaction in a pair of participants, the system is able to replace one of them and interact with a single remaining user.

Thesis Supervisor: Alex Pentland
Title: Academic Head and Toshiba Professor of Media Arts and Sciences, MIT Media Lab

This work was supported in part by British Telecom and Texas Instruments.

Action-Reaction Learning:
Analysis and Synthesis of Human Behaviour

by
Tony Jebara

The following people served as readers for this thesis:

Reader: ______________________________

Bruce M. Blumberg
Asahi Broadcasting Corporation Career Development
Assistant Professor of Media Arts and Sciences
MIT Media Laboratory

Reader: ______________________________

Aaron Bobick
Assistant Professor of Computational Vision
MIT Media Laboratory

Acknowledgments

I extend warm thanks to my advisor, Professor Alex Pentland for having given me to opportunity to come to MIT and for being a great source of inspirational ideas, insight and enthusiasm. Thanks, Sandy, for your support, knowledge, wisdom and patience, and for your faith in me.

I thank Professors Aaron Bobick and Bruce Blumberg for having been readers for this thesis. Thanks, Aaron, for your wit, intuition and in-your-face honesty. Thanks, Bruce, for your inspiring ideas and knowledge on animation and behaviour.

I thank Professor Michael Jordan for having read and commented on parts of the thesis. Thanks, Mike, for sharing your passion and knowledge of machine learning and statistics.

I also wish to thank my friends at the Media Laboratory for their support during the thesis. Thanks to Kris Popat who originally motivated me to think about conditional densities and for his inspirational work on decision-tree conditional density estimation. Thanks, Nuria Oliver, for helping edit this thesis and for your cherished support. Thanks, Nitin Sawhney, for working late-night and being there when everybody else was asleep. Thanks, Deb Roy, for showing me the ropes and reminding me to relax. Thanks to Brian Clarkson for hearing me whine about everything. Thanks to Bernt Schiele for reading the thesis and being the best German post-doc in VisMod. Thanks to Sumit Basu for letting me steal his cookies. Thanks, Tom Minka, for reading the thesis and for great conversations about statistics. Thanks, Ken Russell, for help with face modeling, excellent hacking and, of course, blitzing to FedEx. Thanks as well to Chris Wren and Barbara Rosario, office mates who had to deal with me taking so much space and having such a messy desk. Thanks to the VizModGirls: Tanzeem Choudry, Karen Navarro, Professor Rosalind Picard, Flavia Sparacino and Jen Healey. I'm running out of space so I'll speed up... Thanks to Thad Starner, Baback Moghaddam, Crazy Lee Campbell, Martin Zoomy Szummer, Andy Wilson, Claudio Pinhanez, Yuri Ivanov, Raul Fernandez, Chris Bentzel, Ifung Lu, Eric Trimble, Joey Berzowska, Jonathan Klein, Claudia Urrea, Marina Umaschi, Avinash Sankhla, Dimitri Kountourogiannis, Laxmi Perori and everybody else who I'm forgetting who made this place really fun.

Also, a heart-felt thanks to all my friends from back home in Montreal.

I would most like to thank my family: my father, mother and sister, Carine. They are the best part of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Behavior is what a man does, not what he thinks, feels, or believes.

Source Unknown

The Action Reaction Learning framework is an automatic perceptually based machine learning system. It autonomously studies the natural interactions of two humans to learn their behaviours and later engage a single human in a real-time interaction. The model is fundamentally empirical and is derived from what the humans do externally, not from their underlying behavioural architectures or hard wired cognitive knowledge and models.

Earlier models of human behaviour proposed by cognitive scientists analyzed humans as an input-output or stimulus-response system [68] [62]. The models were based on observation and empirical studies. These *behaviourists* came under criticism as cognitive science evolved beyond their over-simplified model and struggled with higher order issues (i.e. language, creativity, and attention) [35]. Nevertheless, much of the lower-order reactionary behaviour was still well modeled by the stimulus-response paradigm. To a casual observer, these simple models seem fine and it is only after closer examination that one realizes that far more complex underlying processes must be taking place.

The tendency to find simplistic interpretations for underlying complex phenomena has been almost omnipresent in history. Mankind understood the changing of the seasons and could predict and adapt to them long before realizing the world was a sphere which rotated around the sun on a tilted axis. The laws of physics are precise and elegant, however the underlying seasonal change mechanism is complex even though the resulting observed output is simple and regular. In this sense, the old heuristic models still serve their purpose. Similarly, the expression of certain behaviours can be understood at an external observational level without transcendental understanding of the true generative models that cause them.

We propose Action-Reaction Learning (ARL) for the recovery of human behaviour by making an appeal to the behaviourists' stimulus response or input-output model. We also emphasize the recovery of simple, externally observable behaviour as opposed to internal underlying cognitive machinery. For constrained applications, interaction and synthesis, it matters more what a human is doing on the outside than on the inside. The ARL system can be seen as an interesting way to explore how far a perceptual model of behavioural phenomena can be used. When can it explain correlations between gestures and to imitate the simple behaviours humans engage in? The emphasis on real-time perception and synthesis of real-time interactive and expressive output behaviour will therefore be central to the ARL paradigm.

The system begins by observing some interaction between humans to perform a type of imitation learning. The objective of the learning is specifically to later recreate the patterns of behaviour in a synthetic interaction with a single human participant. During the initial training session, the interaction (actions and reactions) being produced by two individuals are first analyzed using perceptual measurements and machine learning to uncover an observational model of the process. The perceptual system consists of a vision algorithm that tracks head and hand motion. Such computer vision analysis and automatic learning of dynamics and simple behaviour from perceptual measurements has recently developed very rapidly [24] [75] [46] [47] [71] [8] [7] [58]. An important transition is beginning to take place as the vision and other perceptual modalities become intimately with machine learning for human observation. These models begin to acquire the ability to make reliable predictions about regularities in human behaviour beyond simple dynamics and direct measurements [47].

Without explicitly discovering the particular underlying generative models that produced the interaction, the system learns a probabilistic mapping between past gestural actions and the future reactions that should follow. The learning is specifically an input-output type of learning which uncovers a predictive model as opposed to a full generative model. These two types of models are cast into statistical terms as conditional (predictive) models and joint (generative) models. In order to solve the desired conditional or predictive aspects of the model, we present a novel algorithm, Conditional Expectation Maximization (CEM), specifically for computing predictive or conditional probabilities from training data. This algorithm employs some novel mathematical machinery, General Bound Maximization (GBM), to optimize a model of the observed human interactions. Of particular relevance is the close similarity of the stimulus-response behaviourist model to input-output machine learning algorithms that have become workhorses in data-driven modeling. The Action-Reaction learning system's probabilistic learning algorithm uncovers such an input-output mapping. Here, the input is a stimulus and the output is the response from interaction data. The goal of the model is not to classify behaviour into categories or arrange it into some prespecified structure. Its objective is to learn how to predict and simulate appropriate behaviour in an interactive setting.

Having formed a predictive model by examining multiple humans interacting, the system is then used to synthesize one of the humans and simulate an interaction with graphical animation output. With advances in computation, the simulation and the analysis of behaviour has become a feasible proposition. In simulation domains, dynamics, kinematics, ethological models, rule based systems and reinforcement learning have been proposed to synthesize compelling interaction with artificial characters [6] [59] [53] [57] [3].

Thus, we propose the combination of the properties of both behaviour simulation and behaviour analysis into a common automatic framework. The Action-Reaction Learning approach acquires models of human behaviour from video and controls synthetic characters. Driven by these models and perceptual measurements, these characters are capable of interacting with humans in real-time. Ultimately, the user need not specify behaviour directly (and tediously) but teaches the system merely by interacting with another individual. The model results from an unsupervised analysis of human interaction and its ultimate goal is the synthesis of such human behaviour. This is attempted with minimal predetermined structures, hand-wired knowledge and user intervention. The behaviours this thesis will address and discuss will be limited to physical activities that can be externally measured by the system. [1]

In this thesis, we will be referring to a more generic definition of behaviour as in the Merriam-Webster dictionary: 'anything that an organism does involving action and response to stimulation'. Alternatively, a hierarchy of behavioural levels can be described as in Figure 1.1 starting with gestures, actions and ending with higher order behaviour. In this more formal sense, behaviour specifically involves higher order concepts such as complex planning, goals, context, and understanding. The Action-Reaction Learning framework will only target the lower-order reactionary gestures and actions in this hierarchy and will not address these higher order cognitive issues.

[1] It is not essential that the input be perceptual. However, perceptual modalities are rich, expressive, intuitive and non-obtrusive. One could take other measurements if they help infer behaviour, internal state or intentionality.

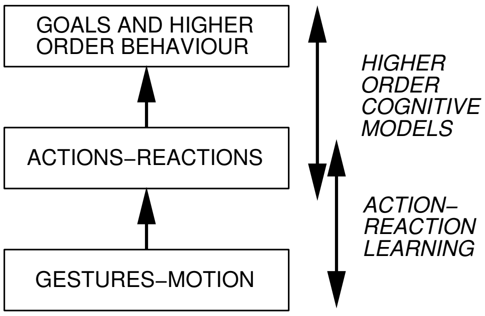


Figure 1.1: Behaviour Hierarchy

1.1 Objectives and Features

The following enumerates some of the objectives of this work. As will be discuss in the related work section, many other efforts in the area of behaviour acquisition and synthesis are underway. We will use this list to specifically distinguish the goals of this project and motivate its contribution. These are key ingredients that we will strive and argue for in the development and use of synthetic behavioural characters.

- Behaviour Learning from Autonomous Observation

  A system is desired that will learn behaviours autonomously without any explicit models or instructions. The system will use observations of natural interactions between humans as training data to acquire such models in a passive way. The natural interactions will take place in a constrained scenario to maintain tractability.

- Imitation Based Behaviour Learning

  The type of learning that will be acquired will be imitation learning to simulate the kind of behaviour patterns that are observed in others. The system is to learn how to interact with humans by imitating prototypical reactions to their actions and learning mappings of appropriate interactive behaviour.

- Fully Perceptual Grounding

  The system will use perception and generate perceptual output which will be the channel through which it performs its acquisition of behaviour. By sensing external stimuli, it should acquire behaviours at a perceptual level and ground them in physical and expressive manifestations (i.e. vision sensing and graphics synthesis).

- Unsupervised Automatic Learning

  Automatic learning is desired without significant data engineering or manual specification of models. There should not be external intervention or direct supervision beyond the specification of inputs and output features. Rather, the system should learn only from constrained yet natural human to human interactions without explicit teaching effort.

- No Discrete Alphabet or Set of Actions

  There will not be a discrete set of actions and reactions to facilitate behaviour selection and associative learning. Input will be in a continuous noisy space where event will seldom be observed exactly the same way twice. Thus, exact deterministic models or if-then types of associations which assist the learning process can not work.

- Probability over Reactions Given Actions

  A fully probabilistic model of the conditional probability over possible continuous reactions given continuous actions is desired. Thus, the system can have some natural stochastic behaviour as opposed to behaving in a fully predictable manner.

- Real-Time, Unencumbered Interaction

  The system is to use the acquired behaviours to interact with other humans in real-time without excessively encumbering the participants. In other words, the humans will engage the system in a natural way without extra paraphernalia and without excessively constraining their behaviour to accommodate the system. In addition, the system must compute the desired action to take and manifest it in some output form in an interactive manner.

- Behaviour Synthesis and Generalization

  The behaviour synthesized has to agree with the acquired behaviours from observations of human participants. An action on the user's behave should induce a sensible reaction on the behalf of the system. The behaviour being synthesized should generalize to interaction with different users and show some robustness to changes in operating conditions.

- Minimal Structural Specifications

  There will be a minimal amount of structure on the system's learning models. The structures being used for learning should be generic enough to uncover temporal interaction in other phenomena that are not only limited to the particular training scenario.

- Computational Efficiency

  The system must learn quickly with limited time and computational resources. The system does not have the luxury of exploring a huge space by trial and error or exhaustive search. In addition, it must learn from few examples without requiring large redundancies and many instances of training data.

1.2 Related Work, Background and Motivation

Several, almost parallel, developments have taken place in a variety of related fields which have significant impact on the proposed task. The acquisition and synthesis of interactive human behaviour is motivated by multiple developments in the fields of artificial perception, machine learning, artificial life, and other areas. In fact, Action-Reaction Learning involves an interesting marriage of multi-disciplinary approaches from perception, learning, synthesis and cognitive science. Some particular examples of related work are discussed here but will also be referred to again throughout this document when they need to be called upon. It is of course impossible to cite all the relevant work in the related disciplines so the following will be a sampling of some important ideas.

1.2.1 Perception

A strong motivation for this work arises from one area of perception: computer vision. An evolution has been proceeding in the field as vision techniques transition from low level static image analysis to dynamic and high level video interpretation. The availability of real-time tracking of bodies, faces, and hands [2] [73] [22] has spurred the development of dynamical analysis of human motion [71] [24] [75] [58] [9]. Isaard [24], Pentland [47], and Bregler [9] discuss the use of multiple linear dynamic models to account for the variability in the motion induced by an underlying behavioural control mechanism. These represent a transition from physical dynamic constraints to behavioural dynamic constraints and can actually infer higher order discrete descriptive models of the actions. In contrast to behaviour modeling vision systems, interactive applications such as Pfinder [73] have been also used for real-time interaction with synthetic characters.

In addition, the development of sophisticated 2D recovery, 3D rigid and non-rigid shape estimation and 3D pose recovery have permitted the use of vision to synthesize and manipulate interesting virtual rendering of real objects [10] [28] [60]. For example, Terzopolous [60] discusses facial tracking and deformation of 3D models to create visually compelling facial animations.

Visual sensing creates an intermediate channel tying together real and virtual worlds. However, it is not the only channel and multi-modal sensing is also becoming an important source of behavioural models and interaction. For instance, auditory perception is used in conjunction with hand tracking in synthetic animals [53] and in synthetic humanoids [61].

1.2.2 Cognitive Science and Ethology

We also look to the extensive knowledge of human behaviour and animal behaviour that has been compiled by cognitive science and ethology. A number of experimental observations and models are worth investigating for both implementation and inspiration purposes.

Cognitive models, for instance, raise the level of abstraction at which synthetic characters can be animated just as physical models permit characters to be animated realistically without pixel-level control [19]. These cognitive models often form a hierarchy with various levels of behavioural interpretation. A set of emotions built on cognitive ideas drives animated characters such as Lyotard, a synthetic cat implemented by Bates [4]. Bates' architecture, Tok, and the emotion unit, Em, are based on the OCC (Ortney, Clore and Collins [45]) cognitive structure of emotions. This yields a more tractable hierarchical approach to emotion synthesis and permits the high level development of behaviourally convincing characters.

Blumberg [6] discusses ethological models of competing behaviours as an effective way of generating autonomous behaviour in synthetic characters. These ethological analogies, permit a good visualization and a hierarchical explanation of what is actually a complex set of rules and functions. Much like the emotions discussed above, goals and other factors can compete in the hierarchy as well.

Additionally, mechanisms that have been used by cognitive scientists from experimental studies could be used as priors in artificial behaviours. The effect of short term memory and the observed decay rates over time have been investigated by [34]. Very similar techniques are used in the short term memory of Blumberg's virtual dog, Silas, where memory decays more rapidly as stimuli increase and accumulate in memory.

Other less graphical examples of cognitive-based behaviour include Weizenbaum's Eliza [70] program. The system uses a series of natural language rules as well as pattern matching to emulate a therapy session. Since the introduction of Eliza, similar dialogue systems are becoming more common and are routinely evaluated with Turing tests to determine how convincing the interactive behaviour is to human users. However, the criticism of these cognitive/computational models is often their dissociation from the physical reality and their lack of grounding in perception and data. Often, these employ a top-down Von-Neumann design (without mentioning bottom-up concepts) and lack grounded or embodied intelligence [11]. This puts into question the applicability of such rule-based mechanisms to domains beyond their particular task.

A more direct motivation for this thesis are the recent cognitive investigations of the primate pre motor cortex that suggest that the development of imitation skills and higher order intellectual activity is related to imitation-specific neurons [51]. Mataric uses such concepts to argue for imitation based learning and discusses the improved acquisition of complex visual gestures in human subjects [14]. In addition, Mataric's robotics research has also been inspired from this cognitive insight as well as ethological models for multi-agent interaction [40].

1.2.3 Synthesis: Interactive Environments, Graphics and Robotics

Interactive environments and the interaction with artificial characters can be traced back to the first days of computing with the introduction of computer games. However, interaction, virtual worlds and animation have gone far beyond these simplistic programs to develop sophisticated understanding and models of synthetic behaviour, dynamics and high-level non-deterministic control. ALIVE [37], a graphical virtual reality environment integrates complex behavioural mechanisms and vision input to synthesize a large space of emergent behaviours. Additionally, computer graphics research in kinematics, physical simulations and dynamics [72] have produced interesting and compelling synthesis and interactions with virtual humans and animals. This area includes work by Badler [3], Thalmann [17] and Terzopolous [59].

The synthesis and acquisition of behaviour has emerged in robotics as an important design paradigm and deserves mention. Evolving beyond the first order design of dynamics and control systems, robotics also utilizes artificial intelligence techniques including adaptivity, path planning, potential functions and navigation models. However, building on more biological inspiration, more complex behaviours including human-like and life-like idiosyncrasies are being implemented [12] [39]. As well, multi-agent interaction and cooperation models have addressed higher order issues and involve more sophisticated behavioural modeling [41] [40].

Of course, the more physically compelling the synthesis, the more believable the more believable the synthetic characters. However, there is a caveat: good graphics and good robotics do not necessarily imply good behaviours, believable interactions or true behavioural learning.

1.2.4 Automatic Machine Learning

The availability of large data sets and computational resources have encouraged the development of machine learning and data-driven models which pose an interesting alternative to explicit and fully structured models of behaviour. A battery of tools are now available which can automatically learn interesting mappings and simulate complex phenomena. These include techniques such as Hidden Markov Models, Neural Networks, Support Vector Machines, Mixture Models, Decision Trees and Bayesian Network Inference [5] [30]. In general, these tools have fallen into two classes: discriminative and generative models. The first attempts to optimize the learning for a particular task while the second models a phenomenon in its entirety. This difference between the two approaches will be addressed in this thesis in particular detail and the above probabilistic formalisms will be employed in deriving a machine learning system for our purposes.

More related learning applications include the analysis of temporal phenomena (after all, behaviour is a time varying process). The Santa Fe competition [20] [18] [43] was a landmark project in this subarea of learning. The methods investigated therein employed a variety of learning techniques to specifically model and simulate the behaviours and dynamics of time series data. The phenomena considered ranged from physiological data to J.S. Bach's last unfinished fugue. The emphasis in many techniques was fully automatic models with flexibility and applicability to a variety of domains. The results from these techniques give a powerful message about modeling complex phenomena: an underlying complex and unobservable hidden phenomenon can be predicted from simple observational cues. For example, the rhythm of the changing seasons and the oscillations of a pendulum can be predicted into the future using observations of their oscillations without a true understanding of the underlying mechanisms [20]. We will call upon the spirit of these temporal modeling techniques and the machine learning principles to acquire behaviours in a data-driven approach.

Of particular relevance to this thesis is the ability to train behavioural models without any direct user intervention. This concept is essential to this thesis and imposes strong constraints and requirements on behavioural design. This is similar in spirit to the work of Sims [57] who relied on genetic algorithms to explore a large space of behavioural models for animating simple life-forms. There is no explicit intervention of the user or the programmer. The designer ultimately engineers a virtual world with physics laws and then provides a cost function (i.e. efficient creature mobility) to favor the evolution of certain artificial life-forms. This concept is also explored in physical domains (as opposed to virtual) by Uchibe [65] where robots acquire behaviour using reinforcement learning. The objective is to play a soccer game and behaviours which accomplish good offensive and defensive plays are favored. Unlike the virtual world of Sims, the robots here have a real-world environment which reduces the manual specification of an artificial reality which can potential be designed by the programmer to induce quite different phenomena. These types of learning are reminiscent of learning by doing or trial-and-error.

An important insight is gained by these approaches. Complex behaviour can emerge using simple feedback and reinforcement type learning. This approach contrasts more involved work which requires explicit identification of behaviours, states, decisions and actions. These come about through the wisdom and artistic efforts of a programmer. Although this process can result in very compelling synthetic behaviours, the task is by no means automatic or easy and requires significant skill. This is often impractical and such techniques are beyond the reach of many users.

Unfortunately, trial-and-error and learning by doing can be an exhaustive process. Without a few clues along the way, a learner can get lost exploring a huge space of poor solutions dominated by negative reinforcement [33]. Unlike reinforcement learning or genetic algorithms which search a space of behaviour solutions, *imitation* learning discovers behaviour from a set of real examples. Behaviour can be learned by observing other agents (i.e. teachers) behaving or interacting with each other. These techniques reduce the search space and provide a set of templates from which further learning (such as reinforcement) can proceed. This type of learning will be the dominant one in this thesis. It involves minimal overhead since behaviour is taught to the system by humans interacting naturally instead of by a programmer identifying rules or cost functions.

Of course, once some initial learning has been performed, meta-learning and higher order introspective learning can begin and might yield better understanding of behaviour. This aspect of learning is still in its infancy in the machine learning field and has much potential [63]. Put simply, once a few learning sessions have been completed, it is possible to reflect on these individual instances and perform deduction and induction. In other words, learn how to learn, discover generalities from the aggregate and recover some higher order structures and properties of each of the individual lessons. This is an area of interesting future research for behaviour acquisition.

1.3 Disclaimer

The following is a disclaimer. In discussing behaviour and learning, we are referring to simple concepts, gestures, patterns and associations which are obtained by a constrained perceptual system in a constrained setting. Thus, higher order issues and the strong AI problem will not be addressed. These will be treated as unsolved and almost philosophical issues at this point. It will be evident from this document what the scope of the problem we are addressing is limited to. It would be imprudent to claim that the machinery discussed mirrors human intelligence or attains transcendental aspects of cognition. We will limit ourselves to the practically realizable systems herein, their abilities and limitations while cautiously avoiding excessive extrapolation.

1.4 Organization

The following list of chapters outlines this thesis. Several concepts are introduced at various levels ranging from the practical to the theoretical with new contributions at multiple levels.

- Chapter 2

  The Action-Reaction Learning framework is initially discussed. The approach treats present activity as an input and future activity as an output and attempts to uncover a probabilistic mapping between them (i.e. a prediction). In particular, by learning from a series of human interactions, one can treat the past interaction of two individuals as an input and try to predict the most likely reaction of the participants. The subsystems are coarsely described including the perceptual engine, the learning engine and the synthesis engine.

- Chapter 3

  The perceptual system (a computer vision based tracker) is presented for acquiring real-time visual data from human users. A technique for the reliable recovery of three blobs corresponding to the head and hands is shown as well as a simple graphical output (the synthesis engine) that generates a corresponding stick figure.

- Chapter 4

  The output of the vision system is discussed as a time series and a variety of pre-processing steps are motivated. This forms a pre-processing engine for the machine learning stage. The representation of the time series data is discussed as well as the feature space used to describe gestures (the actions and the reactions of the human users).

- Chapter 5

  The machine learning engine is investigated and some learning issues are brought into consideration. These include mainly the advantages and disadvantages between generative and discriminative types of learning. The chapter argues for the use of a conditional density (versus a joint density) on the actions and reactions between two human participants and also proposes a Bayesian formalism for it.

- Chapter 6

  The Generalized Bound Maximization framework is introduced as a general tool for optimization. The purpose of the optimization is to resolve some of the learning and estimation issues central to the Action-Reaction Learning paradigm. A set of techniques are presented and tested and form a toolbox which will be utilized in subsequent derivations.

- Chapter 7

  The previous chapter's machinery is applied to the particular problem of learning a conditional density. This results in the derivation of the Conditional Expectation Maximization algorithm. This algorithm is shown to be optimal in the sense of maximum conditional likelihood and will be used as the learning system in the ARL. The machine learning system's model of the relation between input and output (a conditioned mixture of Gaussians) is presented and implementation issues are addressed.

- Chapter 8

  The details of the integrated system as a whole are shown, including the interaction and data-flow between the sub-systems. This forms a high-level summary of the perceptual unit's functionality, the temporal features, the probabilistic inference and the graphical realization of the synthetic character and its animation.

- Chapter 9

  The usage of the system is presented including a particular application domain. The task is put forth and the training and the testing conditions are described. The system is demonstrated to learn some simple gestural behaviours purely from training observations of two humans and then interacts appropriately with a single human participant. Effectively, the system learns to play not by being explicitly programmed or supervised but simply by observing other human participants. Quantitative and qualitative results are presented.

- Chapter 10

  Important extensions to the work are described including more sophisticated perceptual systems, temporal modeling, probabilistic learning and synthesis systems. In addition, other modes of learning are also presented including continuous online learning. Conclusions and contributions are summarized.

- Chapter 11 - Appendix

  This appendix gives a non-intuitive proof-by-example that motivates the the distinction between direct conditional estimation and conditioned joint density estimation. The advantages of conditional probability densities are carried out formally and argue favorably for the input-output learning that is utilized by the ARL system.

Chapter 2

Action-Reaction Learning: An Overview of the Paradigm

The following is an introduction to the Action-Reaction Learning architecture. The system's function is to observe multiple-human interaction passively, learn from it and then utilize this knowledge to interact with a single human.

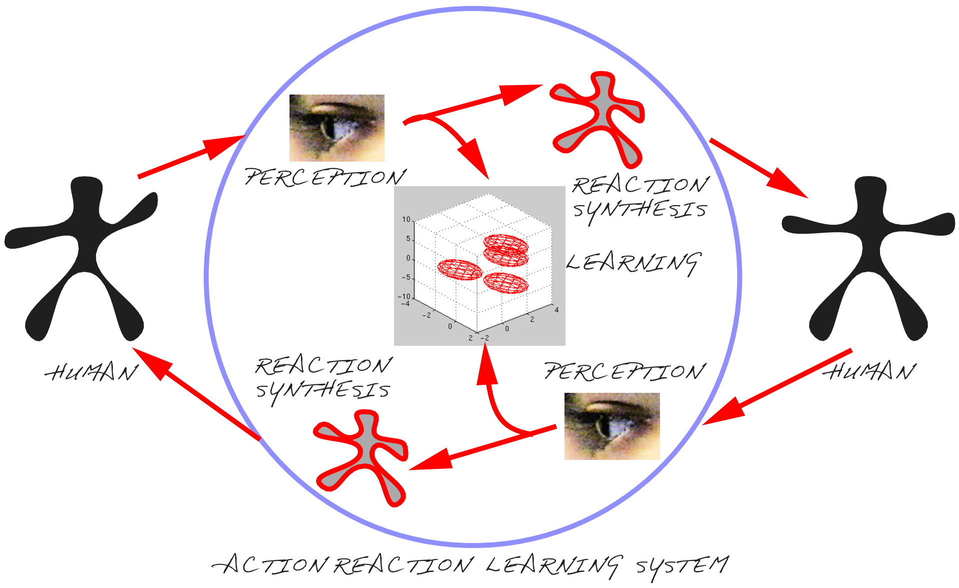


Figure 2.1: Offline: Learning from Human Interaction

The system is depicted in Figure 2.1. Three different types of processes exist: perceptual, synthesis and learning engines interlinked in real-time with asynchronous RPC data paths. Figure 2.1 shows the system being presented with a series of interactions between two individuals in a constrained context (i.e. a simple children's game) [1]. The system collects live perceptual measurements using a vision subsystem for each of the humans. The temporal sequences obtained are then analyzed by a machine learning subsystem to determine predictive mappings and associations between pieces of the sequences and their

[1] Of course, the individuals need not be in the same physical space and could be interacting through a virtual environment.

consequences.

On the left of the figure, a human user (represented as a black figure) is being monitored using a perceptual system. The perceptual system feeds a learning system with measurements which are stored as a time series within. Simultaneously, these measurements also drive a virtual character in a one-to-one sense (the gray figure) which mirrors the left human's actions as a graphical output for the human user on the right. A similar input and output is generated in parallel from the activity of the human on the right. Thus, the users interact with each other through the vision-to-graphics interface and use this virtual channel to visualize and constrain their interaction. Meanwhile, the learning system is 'spying' on the interaction and forming a time series of the measurements. This time series is training data for the system which is attempting to learn about this ongoing interaction in hopes of modeling and synthesizing similar behaviour itself.

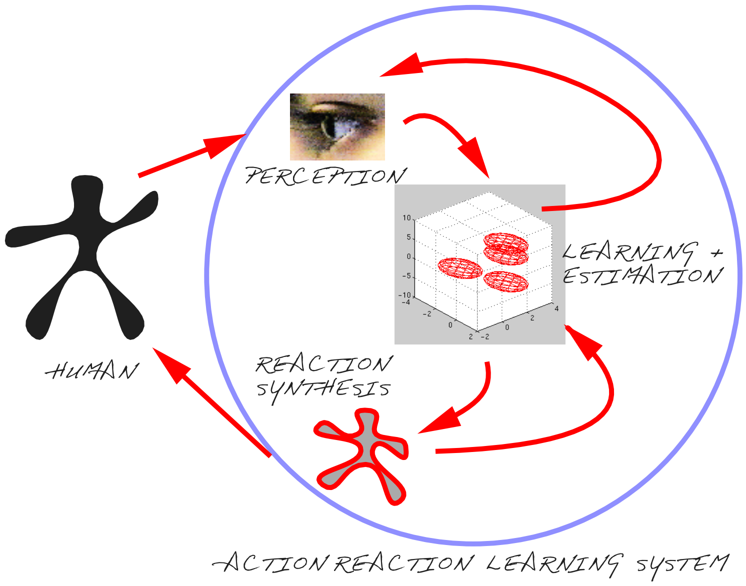


Figure 2.2: Online: Interaction with Single User

In Figure 2.2, the system has collected and assimilated the data. At this point it can computationally infer appropriate responses to the single remaining human user. Here, the perceptual system only needs to track the activity of the one human (black figure on the left) to stimulate the learning or estimation system for real-time interaction purposes (as opposed to interaction learning as before). The learning system performs an estimation and generates the most likely response to the user's behaviour. This is manifested by animating a computer graphics character (gray figure) in the synthesis subsystem. This is the main output of the ARL engine. It is fed back recursively into the learning subsystem so that it can remember its own actions and generate self-consistent behaviour. This is indicated by the arrow depicting flow from the reaction synthesis to the learning + estimation stage. Thus, there is a continuous feedback of self-observation in the learning system which can recall its own actions. In addition, the system determines the most likely action of the remaining user and transmits it as a prior to assist tracking in the vision subsystem. This flow from the learning system to the perception system (the eye) contains behavioural and dynamic predictions of the single user that is being observed and should help improve perception

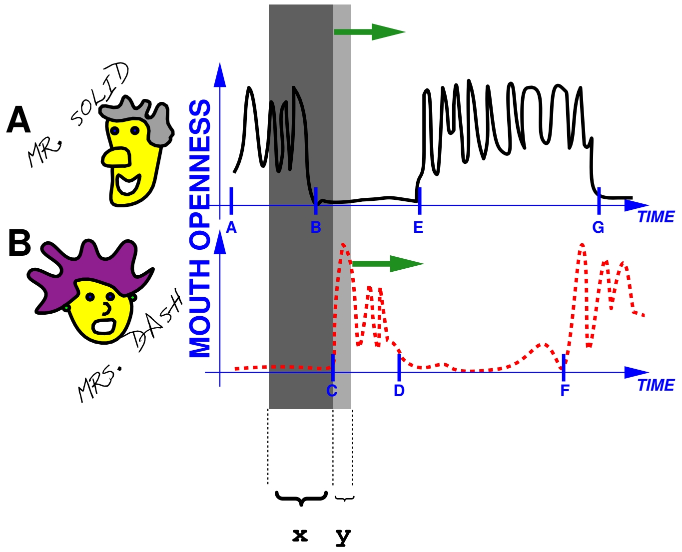


Figure 2.3: Dialog Interaction and Analysis Window

2.1 A Typical Scenario

Action-Reaction Learning (ARL) involves temporal analysis of a (usually multi-dimensional) data stream. Figure 2.3 displays such a stream (or time series). Let us assume that the stream is being generated by a vision algorithm which measures the openness of the mouth [44]. Two such algorithms are being run simultaneously on two different people. One person generates the dashed line and the other generates the solid line.

Now, imagine that these two individuals are engaged in a conversation. Let us also name them Mr. Solid (the fellow generating the solid line) and Mrs. Dash (the lady generating the dashed line). Initially (interval A-B on the time axis), Mr. Solid is talking while Mrs. Dash remains silent. He has an oscillatory mouth signal while she has a very low value on the openness of the mouth. Then, Mr. Solid says something shocking and pauses (B-C). Mrs. Dash then responds with a discrete 'oh, I see' (C-D). She too then pauses (D-E) and waits to see if Mr. Solid has more to say. He takes the initiative and continues to speak (E). However, Mr. Solid continues talking non-stop for just too long (E-G). So, Mrs. Dash feels the need to interrupt (F) with a counter-argument and simply starts talking. Mr. Solid notes that she has taken the floor and stops to hear her out.

What Action-Reaction Learning seeks to do is discover the coupling between the past interaction and the next immediate reaction of the participants. For example, the system may learn a model of the behaviour of Mrs. Dash so that it can predict and imitate her idiosyncrasies. The process begins by sliding a window over the temporal interaction as in Figure 2.3. The window looks at a small piece of the interaction and the immediate reaction of Mrs. Dash. This window over the time series forms the short term or iconic memory of the interaction and it is highlighted with a dark rectangular patch. The consequent reaction of Mrs. Dash and Mr. Solid is highlighted with the lighter and smaller rectangular strip. The first strip will be treated as an input $\mathbf{x}$ and the second strip will be the subsequent behavioural output of both Mr. Solid and Mrs. Dash ($\mathbf{y}$). To predict and imitate what either Mr. Solid or Mrs. Dash will do next, a system system must estimate the future mouth parameters of both (stored in $\mathbf{y}$). As the windows slide across a training interaction between the humans, many such $(\mathbf{x}, \mathbf{y})$ pairs are generated and presented as training data to the system. The task of the learning algorithm is to learn from these pairs and form a model relating $\mathbf{x}$ and $\mathbf{y}$. It can then generate a predicted $\mathbf{y}^*$ sequence whenever it observes a past $\mathbf{x}$ sequence. This allows it to compute and play out the future actions of one of the users (i.e. Mrs. Dash) when only the past interaction of the participants is visible.

Thus, the learning algorithm should discover some mouth openness behavioural properties. For example, Mrs. Dash usually remains quiet (closed mouth) while Mr. Solid is talking. However, after Solid has talked and then stopped briefly, Mrs. Dash should respond with some oscillatory signal. In addition, if Mr. Solid has been talking continuously for a significant amount of time, it is more likely that Mrs. Dash will interrupt assertively. A simple learning algorithm could be used to detect similar $\mathbf{x}$ data in another situation and then predict the appropriate $\mathbf{y}$ response that seems to agree with the system's past learning experiences.

Note now that we are dealing with a somewhat supervised learning system because the data has been split into input $\mathbf{x}$ and output $\mathbf{y}$. The system is given a target goal: to predict $\mathbf{y}$ from $\mathbf{x}$. However, this process is done automatically without any manual data engineering. One only specifies a-priori a constant width for the sliding window that forms $\mathbf{x}$ and the width of the window of $\mathbf{y}$ (usually, the width will be 1 frame for $\mathbf{y}$ to conservatively forecast only a small step into the future). The system then operates in an unsupervised manner as it slides these windows across the data stream. Essentially, the learning uncovers a mapping between *past and future* to later generate its best possible prediction.

2.2 Discussion and Properties

At this point, we outline some of the properties and features of this ARL behavioural learning paradigm and find some loose analogies to other cognitive and learning domains. Notions of automatic, imitation-type learning, perception oriented behaviour, short term memory, self-consistency, state and stochastic behaviour are addressed.

2.2.1 Perception and Expression

We begin with emphasis on observation and perception in this behavioural model. On one hand, a simple phenomenon can produce extremely complex chaotic behaviour, such as the famous Lorenz equations. On the other hand, however, a very complex system can also produce (for the most part) consistently recurring simple patterns and regularities, and trigger simplistic observations. For example, it is very easy to predict the location of the sun in the sky from the periodic nature of night and day, however, the underlying mechanism (gravitation and orbits) are significantly harder to understand. Mankind understood the first concept far before mastering the second. The ARL paradigm is predominantly concerned with perceptual observations taken from a system (i.e. a human) and focusses on regenerating the perceived behaviour. Thus, in this emphasis, an observational system is more likely to capture what people notice in their perception as opposed to the true underlying causes and processes. Some complex components of human behaviour (i.e. watching TV) might require extremely complex mental modeling but ultimately, might be expressed in a minimal way (i.e. staring blankly). The behaviour we are trying to recover is the perceptually energetic and observable component that is manifested in physical actions. It should be easily measureable using human-like sensors (vision). Therefore, behavioural modeling resources will not be squandered recovering modes of operation that will not have strong expressivity. The focus on perceptual and expressive modes would, consequently, provide an exterior observer of the ARL system's interaction synthesis a more convincing illusion of behaviour and life.

2.2.2 Imitation Learning

In cognitive science work, imitation style learning has been investigated as a source of higher order intelligence and fast acquisition of knowledge [51]. The ARL model is also intimately coupled with the notions of imitation learning. There is no exhaustive search in the acquisition of behaviour but rather a direct relationship between teachers and student through the interactions that were demonstrated. Here, the teachers are humans interacting with each other and the student is the ARL system. There is no truly laborious teaching on the behalf of the humans since they merely interact naturally. In essence, the learning that is taking place involves no real programming or data processing effort for the humans. There is no manual classification, labeling, segmentation, reinforcement or structure identification in this scenario. Observations are acquired and learned autonomously by the system. The model is fully estimated using standard, computationally tractable probabilistic approaches and exploits tried and true Bayesian statistical formalisms.

2.2.3 Probabilistic and Bayesian Machine Learning Techniques

The machine learning approach that will be implemented will be discussed subsequently however it is important to note that it is a rather generic one. There is no explicit notion of dynamics, physical constraints, physical laws, behavioural structures, etc. This allows the model to be flexible and applicable to different learning environments. Sometimes, perceptual and expressive data will not fall into cognitive models that can be implemented computationally. Thus, for generalization purposes, we will not take unnecessary advantage of constraints about the domain, the tasks and the perceptual data provided to the ARL system by using hard wired cognitive models or a priori user defined models.

In addition, a probabilistic model provides a methodological framework for operating on and flexibly learning from standard data samples. It allows conditioning, sampling, computing expected behaviours and computing maximum likelihood estimates in a formal and consistent sense [5]. In addition, the underlying probabilistic mechanisms allow 'soft' associations, stochastic behaviour and continuous, unlabeled and unconstrained representations. Even though it may not be as structured or as specific as cognitive or alternative models of behaviour, it is computationally more flexible. Finally, machine learning is an automatic process with minimal effort on the behalf of the user. The desired behaviours are enacted in front of the system rather than explicitly identified by the user who must incorporate specific knowledge and skill from various domains. In other words, many variables can be estimated from data rather than manually tweaked by users.

2.2.4 Cognitive Concepts

The simplicity of Action-Reaction Learning is reminiscent of the behaviourists' model. However, the model also addresses some of the limitations and hurdles associated with the behaviourists' viewpoint. ARL also employs cognitive concepts that are more recent than behaviourism. Some cognitive concepts such as short term memory and attentional mechanisms are contained in the model. One example is an explicit attentional window with memory decay [16]. This is implemented to analyze past interactions in a short-term memory sense. The window also forms a coarse state model which allows the system to recall its own previous actions and maintain some self-consistent behaviour. Eventually, the approach could be made adaptive and more complete notions of internal state could be included (i.e. spanning variable length memory). The subsequent learning system would thus exhibit superior generalization and more realistic behaviour without extending the problem beyond the capacity of contemporary machine learning techniques.

Chapter 3

Visual Inputs and Outputs

A wide array of possible perceptual inputs and possible output modalities can be considered for a system that interacts with and learns from human users. A primary concern in the perceptual system is the recovery of action parameters which are particularly expressive and interactive. In addition, to maintain real-time interactiveness and fast training, the input/output parameters should be compact (low dimensional). Another equally important attribute of the perceptual features is linearity and smooth behaviour in their parametric space. This property is more difficult to characterize and the representation of data is an ongoing research issue. It is sufficient to say that the learning system already has a significant task in learning mappings between actions and reactions to acquire behaviour. The difficulties in the learning task can be severely compounded by the use of spurious, complex or strangely non-linear perceptual inputs. Thus, in designing a perceptual system, we begin cautiously and use only the simplest features.

3.1 Head and Hand Tracking

We will begin with a relatively compact perceptual system that will be used for gesture behaviour learning. A tracking system is used to follow head and hand as three objects (head, left and right hand). These are represented as 2D ellipsoidal blobs with 5 parameters each. With these features alone, it is possible to engage in simple gestural games and interactions.

The vision algorithm begins by forming a probabilistic model of skin colored regions [1] [56] [53]. During an offline process, a variety of skin-colored pixels are selected manually, forming a distribution in rgb space. This distribution can be described by a probability density function (pdf) which is used to estimate the likelihood of any subsequent pixel ($\mathbf{x}_{rgb}$) being a skin colored pixel. The pdf used is a 3D Gaussian mixture model as shown in Equation 3.1 (with $M = 3$ individual Gaussians typically).

$$p(\mathbf{x}_{rgb}) = \sum_{i=1}^{M} \frac{p(i)}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma_i|}} \; e^{-\frac{1}{2}(\mathbf{x}_{rgb}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}_{rgb}-\mu_i)} \tag{3.1}$$

The parameters of the pdf ($p(i)$,μ_i and Σ_i) are estimated using the Expectation Maximization [15] algorithm to maximize the likelihood of the training rgb skin samples. This pdf forms a classifier and every pixel in an image is filtered through it. If the probability is above a threshold, the pixel belongs to the skin class, otherwise, it is considered non-skin. Figures 3.1(a) and (d) depict the classification process.

To clean up some of the spurious pixels misclassified as skin, a connected components algorithm is performed on the region to find the top 4 regions in the image, see Figure 3.1(b). This increases the robustness of the EM based blob tracking. We choose to process the top 4 regions since sometimes the

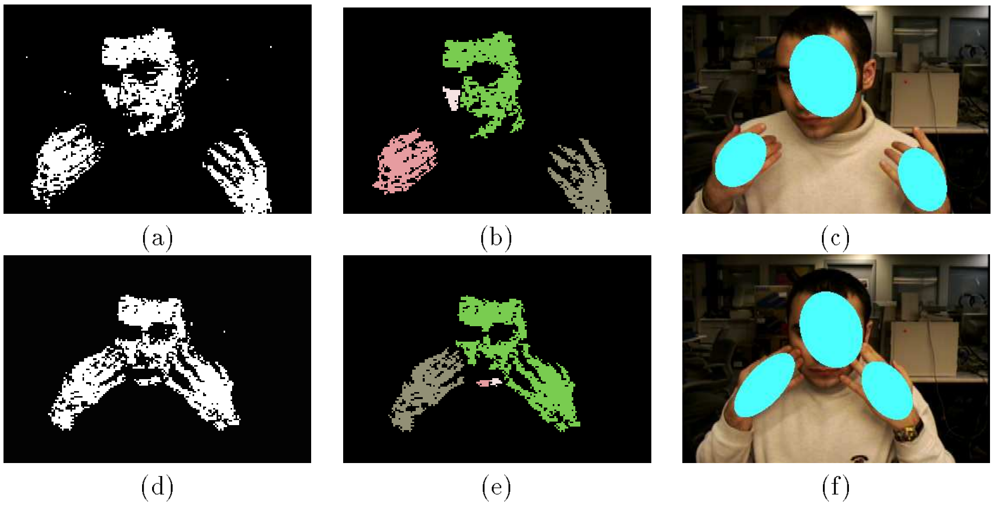


Figure 3.1: Head and Hand Blob Tracking

face is accidentally split into two regions by the connected components algorithm. In addition, if the head and hands are touching, there may only be one non-spurious connected region as in Figure 3.1(e).

Since we are always interested in tracking three objects (head and hands) even if they touch and form a single connected region, it is necessary to invoke a more sophisticated pixel grouping technique. Once again, we use the EM algorithm to find 3 Gaussians that this time maximize the likelihood of the *spatially* distributed (in xy) skin pixels. Note that the implementation of the EM algorithm here has been heavily optimized to require less than 50ms to perform each iteration for an image of size 320 by 240 pixels. This Gaussian mixture model is shown in Equation 3.2.

$$p(\mathbf{x}_{xy}) = \sum_{j=1}^{3} \frac{p(j)}{2\pi\sqrt{|\Sigma_j|}} \; e^{-\frac{1}{2}(\mathbf{x}_{xy}-\mu_j)^T \Sigma_j^{-1}(\mathbf{x}_{xy}-\mu_j)} \tag{3.2}$$

The update or estimation of the parameters is done in real-time by iteratively maximizing the likelihood over each image. The resulting 3 Gaussians have 5 parameters each (from the 2D mean and the 2D symmetric covariance matrix) and are shown rendered on the image in Figures 3.1(c) and (f). The covariance (Σ) is actually represented in terms of its square root matrix, Γ where $\Gamma \times \Gamma = \Sigma$. Like Σ, the Γ matrix has 3 free parameters ($\Gamma_{xx}, \Gamma_{xy}, \Gamma_{yy}$) however these latter variables are closer to the dynamic range of the 2D blob means and are therefore preferred for representation. The 5 parameters describing the head and hands are based on first and second order statistics which can be reliably estimated from the data in real-time. In addition, they are well behaved and do not exhibit wild non-linearities. Consequently they are adequate for temporal modeling. More complex measurements could be added in the future but these would typically be more unstable to estimate and might have non-linear phenomena associated with them. The 15 recovered parameters from a single person are shown as a well behaved, smooth time series in Figure 3.2. These define the 3 Gaussian blobs (head, left hand and right hand).

The parameters of the blobs are also processed in real-time via a Kalman Filter (KF) which smoothes and predicts their values for the next frame. The KF model assumes constant velocity to predict the next observation and maintain tracking.

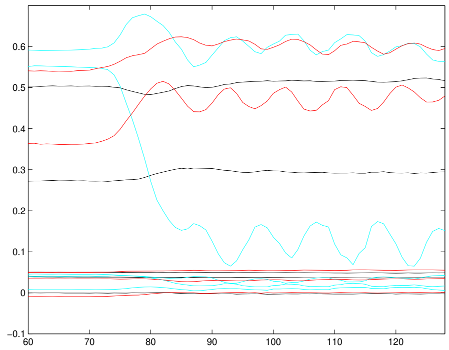


Figure 3.2: Time Series Data of 3 Blobs (1 User)

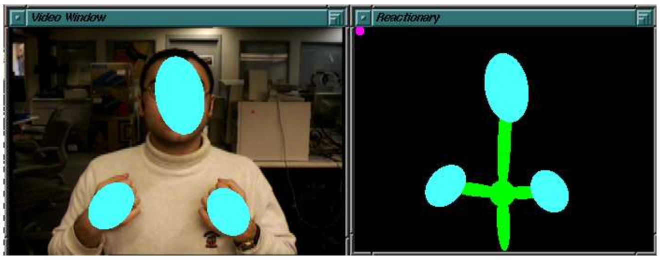


Figure 3.3: Graphical Rendering of Perceptual Measurements

3.1.1 Glove Tracking

It is also possible to avoid some of the problems in tracking multiple skin blobs by using colored gloves. Some of these problems included confusion when blobs would severely occlude or pass by each other and confuse the correspondence. Thus, colored gloves allows the system to be more robust to such occlusions as well as other various environments. Therefore, the vision system can be switched into this optional operation mode whenever skin tracking is unreliable. Instead of using only a probabilistic density model of skin color, each user wears two differently colored gloves and thus a unique probabilistic color model is used for each object. For instance, the face is tracked using the default skin color class, the left hand employs a blue glove color class and the right hand uses a red glove color class. This multi-color object tracking is a more reliable mode of operation and is used to initially train the system. It provides cleaner data and should result in much faster convergence of the behavioural learning. Subsequently, when the system has started learning, and has *some* limited behavioural model, it is possible to remove the gloves on the assumption that the behavioural learning has acquired some understanding of the role of the three blobs as distinct entities (left hand, right hand and head). Otherwise, the correspondence between blobs might not be resolved properly.

3.2 Stick Figure Graphical System

At each time frame, the 15 estimated parameters for the Gaussians can be rendered for viewing as the stick figure in Figure 3.3. This is also the display provided to each user so that he may view the gestures of other human (or computer) players through his personal computer screen.

The output is kept simple to avoid confusing users into believing that more sophisticated perception is taking place. This is critical since users will immediately realize how to use this limited channel expressively. If a sophisticated rendering of a face and hands was used on top of the stick figure, humans might mistakingly believe that these features contain information that is being relayed to the learning system. Thus, we avoid misleading the users and hope that they will emphasize large head and hand gestures.

3.3 The Training and Testing Environment

Figure 3.4 depicts the training environment. Here, two users are being observed by the system via two video cameras. Additionally, both users can see virtual representations of each other as blob and stick figures. Since this is a distributed RPC system, any number of users can be engaged in this type of interaction. Each user opens a display and a vision system and can select and view any of the other participants connected to the Action-Reaction Learning module. We will not discuss the multi-user case and will focus on single or 2-person cases exclusively. However, the perceptual framework (and the learning framework) is generalizable to large group interactions as well. In addition, in a third nearby display, the Action Reaction Learning system is running and plotting trajectories describing the gestures of both users in real-time.

In the testing environment, one of the two users will leave and disable his perceptual system. This will trigger the ARL system to start simulating his behaviour and interpolating the missing component of the perceptual space that he was previously triggering. This process should be transparent to the remaining user who should still feel (to a certain degree) that she is virtually interacting with the other human. Of course, only one video camera and one display is necessary at this point.

3.4 Other Sensing and Graphics

Certainly, it is possible to extend the perceptual measurements above and a variety of techniques have been developed within the framework of this thesis. However, we will defer discussion of these input and output modalities until later. Without loss of generality, we focus on this constrained case of head and hand blob tracking for the rest of the system's description. Subsequently, future work with higher order visual perception will be described and will fit almost directly into the framework derived for the head and hand modeling case.

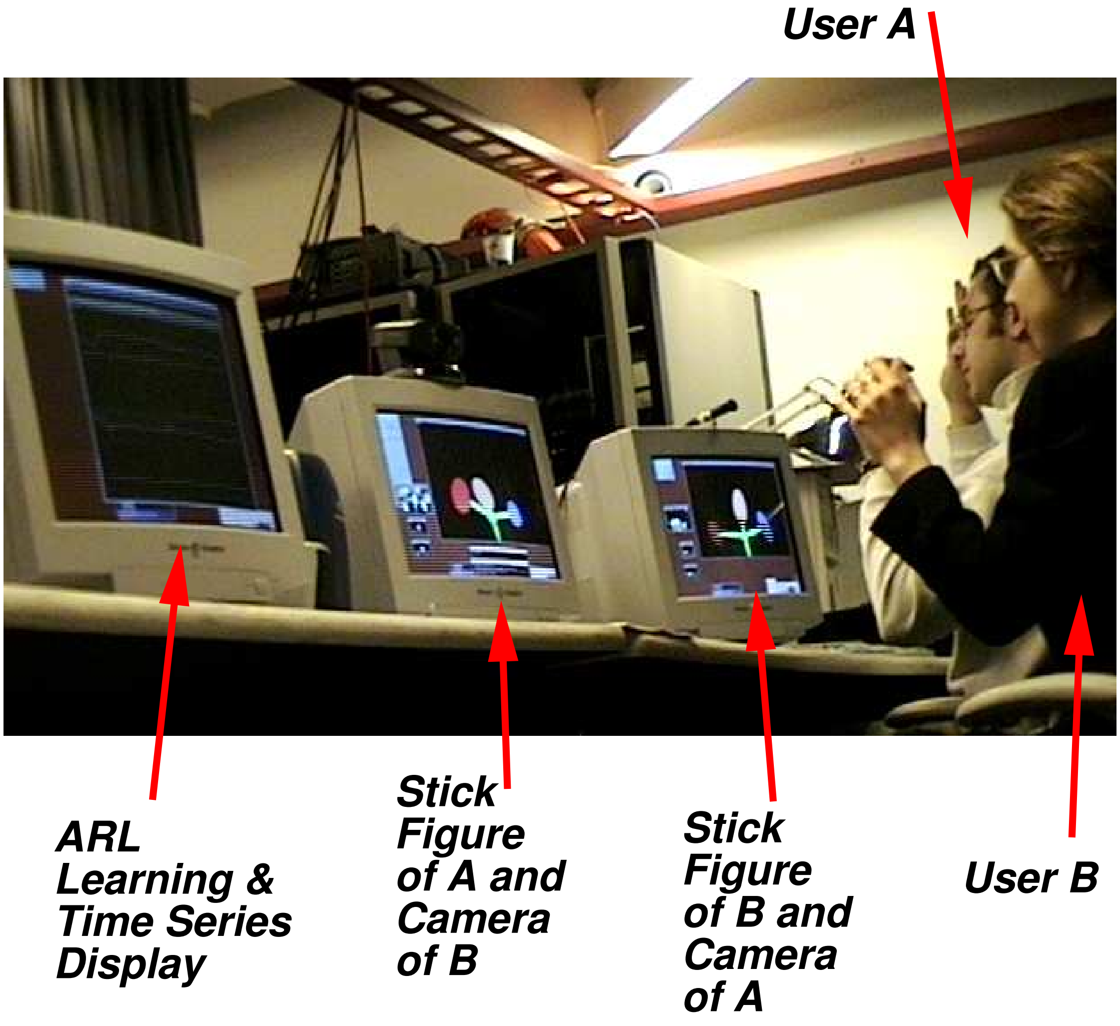


Figure 3.4: Training the System Perceptually

Chapter 4

Temporal Modeling

The Action-Reaction Learning system functions as a server which receives real-time multi-dimensional data from the vision systems and re-distributes it to the graphical systems for rendering. Typically during training, two vision systems and two graphics systems are connected to the ARL server. Thus, it is natural to consider the signals and their propagation as multiple temporal data streams. Within the ARL server, perceptual data or tracked motions from the vision systems are accumulated and stored explicitly into a finite length time series of measurements.

For the head and hand tracking case, two triples of Gaussian blobs are generated (one triple for each human) by the vision systems and form 30 continuous scalar parameters that evolve as a multi-dimensional time series. Each set of 30 scalar parameters can be considered as a 30 dimensional vector $\mathbf{y}(t)$ arriving into the ARL engine from the vision systems at a given time t. The ARL system preprocesses then trains from this temporal series of vectors. However, certain issues arise when processing time series data and dealing with temporal evolution of multidimensional parameters. The representation of this data is critical to reliably predict and forecast the evolution of the time series or, equivalently, estimate the parameters of the 6 blobs in near future. The future value of $\mathbf{y}$ will be referred to as $\mathbf{y}(t)$ and it must be forecasted from several measurements of the previous vectors $\mathbf{y}(t-1), \mathbf{y}(t-2), ...$ which will form a window of perceptual history.

4.1 Time Series

A detailed account of the Santa Fe competition is presented in [20]. The document covers many of the issues involved in time series modeling and prediction that are relevant to the objectives posed above. In this collected work [20], multiple signal types are considered with rather general tools as well as domain-specific approaches. An important insight results: the representation of the time-series signal critically depends on the machine learning engines that it will feed into. In that sense, the temporal representation of data can be treated as a pre-processing engine that feeds a specific learning system. The learning stage obtains data and forms a model that is optimized to predict future time series values from present ones. This task falls into the general category of regression which learns a mapping between inputs and outputs, effectively approximating a function and computing a new output given a new input. In the case of time series, the input is a past sequence and the output is the subsequent sequence (i.e. the forecast or prediction).

While regression has traditionally been associated with linear models and ARMA (autoregressive moving average) processing, the Santa Fe results note the weakness of linear models and the over-simplifying assumptions they imply. These classical techniques are contrasted with the far more succesful neural and connectionist approaches. This more recent modeling paradigm uses fundamentally non-linear techniques for signal prediction. This non-linear nature is necessary since the data employed in the com-

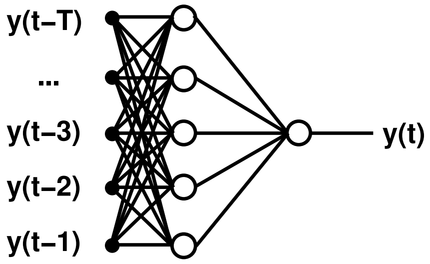


Figure 4.1: Static Network Representation of Temporal Modeling

petition ranged over complex non-linear phenomena such as physiological signals and Bach's unfinished compositions. Similarly, we argue that we have no good reason to assume that the perceptually recovered gestures and behaviours in the ARL system should be linear as well. At this point we examine the neural or connectionist approaches for time series forecasting and for representation. These ideas will be used to motivate the design of the ARL learning system and the representations it will employ. We focus on the connectionist approach due to its explicit non-linear optimization of prediction accuracy and its superior performance in the Santa Fe competition against systems like hidden Markov models, dynamic models, and so on.

In his implementation of a Time Delay Neural Network (TDNN), Wan [67] contrasts this time series based connectionist architecture with the standard static multi-layer neural network. In theory, the nonlinear autoregression being computed is a mapping between an output vector $\mathbf{y}(t)$ and T previous instances of the vector $\mathbf{y}(t-1), \mathbf{y}(t-2), ..., \mathbf{y}(t-T)$ as in Equation 4.1. The latest observation here is $\mathbf{y}(t-1)$ and the oldest is $\mathbf{y}(t-T)$. The neural network has to find an approximation to the function $g()$ which is often denoted $\hat{g}()$. In our case, $\mathbf{y}$ is a 30 dimensional vector containing the perceptual parameters recovered from the two human teachers in the training mode. Each user generates 3 Gaussian blobs (head and hands) and these 30 parameters are merely concatenated into a single vector $\mathbf{y}$.

$$\mathbf{y}(t) = g\left(\mathbf{y}(t-1), \mathbf{y}(t-2), ..., \mathbf{y}(t-T)\right) \tag{4.1}$$

Thus, the function $g()$ produces an output $\mathbf{y}(t)$ which is 30 dimensional for the head and hand tracking case but requires T previous vector observations of $\mathbf{y}$. Thus, the input space of $g()$ is $D \times T = 30 \times T$. If implemented as a regular multi-layer neural network (static) as in Figure 4.1, there is a large increase in complexity with increasing past observation vectors T that can be stored simultaneously.

The T value represents the number of observation vectors in the system's memory and these will determine how much past data it may use to make forecasts about the immediate future. For head and hand tracking data (which generates vectors at roughly 15Hz), values of $T \approx 120$ are required to form a short term memory of a few seconds ($\approx$ 6 seconds). Thus, the input domain of the neural function $g()$ would grow to over 3600 dimensions.

Wan discusses an algorithm which treats the time series using finite impulse response (FIR) neurons and symmetries on a static network topology to reduce the dimensionality problem and take advantage of some symmetries and redundancies. We shall now divert from his methodology and address the large dimensionality of the input space using an alternative approach.

4.2 Principal Components Analysis

Instead of directly dealing with the redundancies in a large network topology, we will factor out complexity in the input space using a common dimensionality reduction procedure, *Principal Components Analysis*

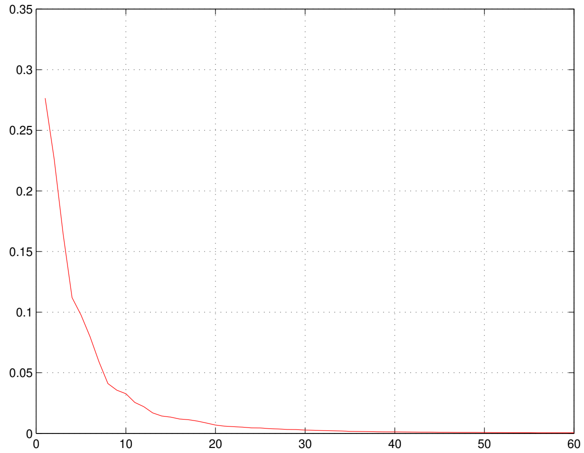


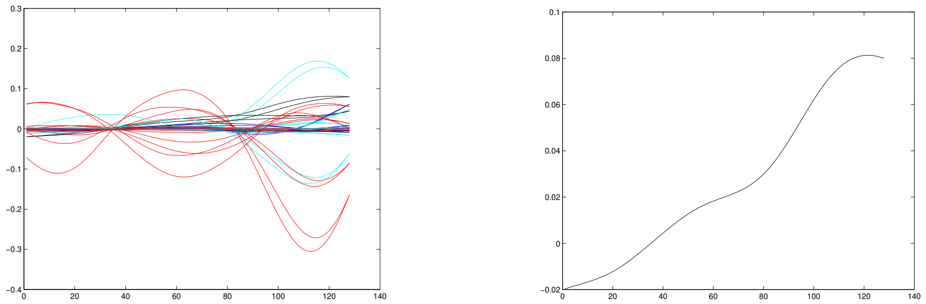
Figure 4.2: Top 60 Eigenvalues in Y-space

(PCA) [5]. Consider the a large vector $Y(t)$ composed of the concatenation of all the T vectors that were just observed $\mathbf{y}(t-1), \mathbf{y}(t-2), \ldots\mathbf{y}(t-T)$. Again, as time evolves and we consider the training set as a whole, many large vectors Y (each representing 6 second chunks of interaction) are produced as well as their corresponding subsequent value $\mathbf{y}(t)$. Therefore the vectors Y contain short term memory of the past and are not merely snapshots of perceptual data at a given time. Thus, many instances of short term memories Y are collected and form a distribution.

PCA forms a partial Gaussian model of the distribution of Y in a large dimensional space by estimating a mean and covariance. In the ARL system, thousands of these Y vectors are formed by tracking a handful of minutes of interaction between two humans. The PCA analysis then computes the eigenvectors and the eigenvalues of the covariance. The eigenvectors are ranked according to their eigenvalues with higher eigenvalues indicating more energetic dimensions in the data. Figure 4.2 depicts the top few eigenvalues.

From simple calculations, we see that over 95% of the energy of the Y vectors (i.e. the short term memory) can be represented using only the components of Y that lie in the subspace spanned by the first 40 eigenvectors. Thus, by considering Y in the eigenspace as opposed to in the original feature space, one can approximate it quite well using less than 40 coefficients. In fact, from the sharp decay in eigenvalue energy, it seems that the distribution of Y occupies only a small submanifold of the original 3600 dimensional embedding space. Thus we can effectively reduce the dimensionality of the large input space by almost two orders of magnitude. We shall call the low-dimensional subspace representation of $Y(t)$ the immediate past short term memory of interactions and denote it with $\mathbf{x}(t)$.

In Figure 4.3 the first mode (the most dominant eigenvector) of the short term memory is rendered as a 6 second evolution of the 30 head and hand parameters of two interacting humans. In addition, it is shown as a 6 second evolution of one of these parameters alone, the x (i.e. horizontal) coordinate of the head of one of the humans. Interestingly, the shape of the eigenvector is not exactly sinusoidal nor is it a wavelet or other typical basis function since it is specialized to the training data. The oscillatory nature of this vector indicates that gestures involved significant periodic motions (waving, nodding, etc.) at a certain frequency. Thus, we can ultimately describe the gestures the participants are engaging in using a linear combination of several such prototypical basis vectors. These basis vectors span the short term memory space containing the Y vectors.



(a) For all 30 Features (head and hands of two users) (b) For Feature 1

Figure 4.3: First Eigenvector of Past Interaction

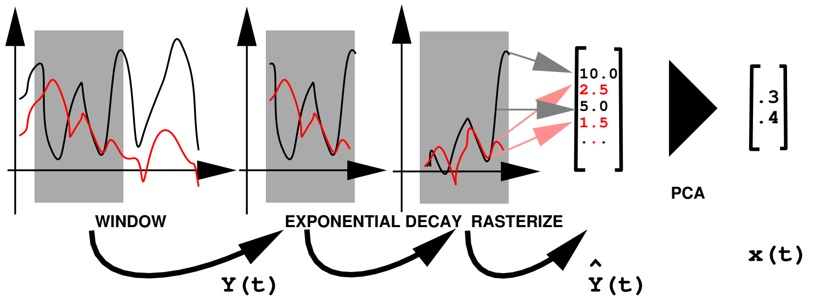
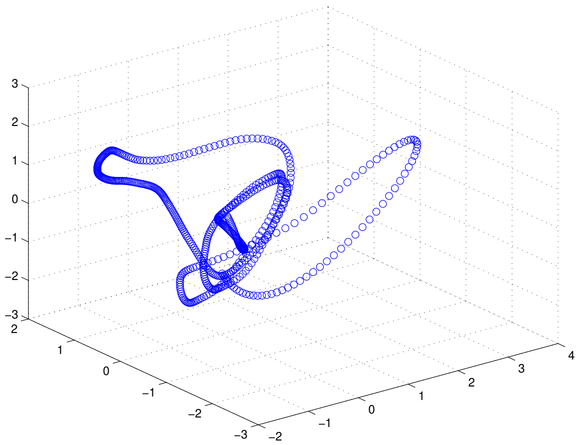


Figure 4.4: Exponential Decay and Pre-Processing

4.3 Pre-Processing Implementation and Exponential Memory Decay

It should be noted that the above analysis actually used weighted versions of the Y vectors to include a soft memory decay process. Figure 4.4 depicts the pre-processing, exponential weighting and dimensionality reduction. Instead of abruptly cutting off the Y at the T'th sample of $\mathbf{y}$, a smooth decay is applied to more distant observations.

Recall that the large vectors $Y(t)$ represent a full window of past interaction (short term memory). This window effectively covers $120 \times 50ms = 6.5$ seconds of temporal data. This data is weighted with an exponential decay which scales down $\mathbf{y}$ vectors that constitute the big $Y(t)$ vector. The further back in time a $\mathbf{y}$ component is, the more its amplitude is attenuated. Thus, an exponential ramp function is multiplied with each Y window (i.e. a few seconds of each of the 30 time series). This reflects our intuition that the more temporally distant the elements in the time series, the less relevant they are for prediction. This decay agrees with some aspects of cognitive models obtained from psychological studies [16]. Once the vectors have been attenuated, they form a new 'exponentially decayed' short term memory window $\hat{Y}(t)$. The process is shown in Figure 4.4 where a window is placed over the time series, generating a short term memory Y. An exponential decay function is used to decay it and generates the $\hat{Y}$ version. The eigenspace previously discussed is really formed over the $\hat{Y}$ distribution and representing $\hat{Y}$ in only this subspace (i.e. the top eigenvectors) generates the compact vector $\mathbf{x}(t)$. This is the final, low dimensional representation of the gestural interaction between the two humans over the past few seconds.



(b) Projection onto top 3 Eigenvectors

Figure 4.5: Eigenspace Properties for $\mathbf{x}$

4.4 Probabilistic Time Series Modeling

Of course, immediately after the time window over the past, another observation $\mathbf{y}(t)$ (of the near future) is also obtained from the training data. One may again simply vectorize the parameters of the perceptual system (the Gaussian tracking blobs) into yet another $\mathbf{y}$ vector (of dimensionality $\mathcal{R}^{30}$). The $\mathbf{x}(t)$ vector represents the past action and the $\mathbf{y}(t)$ represents the consequent reaction exactly at time t. For a few minutes of data, we can obtain thousands of pairs of $\mathbf{x}$ and $\mathbf{y}$ vectors (i.e. action-reaction pairs) by sliding the attentional window over the time series (i.e. considering all values of t) and processing as explained above. Figure 4.5 shows the evolution of the dominant 3 dimensions of the $\mathbf{x}(t)$ vectors as we consider an involved interaction between two participants over time t of roughly half a minute. This represents the evolution of the short term memory of the learning system during half a minute.

Given sufficient pairs of the vectors ($\mathbf{x}$(t),$\mathbf{y}$(t)) from training data, it is possible to start seeing patterns between a short term memory of the past interaction of two humans and the immediate subsequent future reaction. A system which can forecast this behaviour could predict what to do next and engage with a single human. However, instead of learning an exact deterministic mapping between $\mathbf{x}$ and $\mathbf{y}$, as is done in a predictive neural network, we will discuss a more probabilistic approach. This involves estimating a probability density denoted as $p(\mathbf{y}|\mathbf{x})$ which yields the probability of a reaction *given* a short history of past action. We will always be observing the past ($\mathbf{x}$) but the future ($\mathbf{y}$) is what we are trying to predict. We are not, for instance, interested in the conditional pdf $p(\mathbf{x}|\mathbf{y})$, which computes the probability of the past ($\mathbf{x}$) given the future. In other words, we will seldom use the learning system in the almost 'philosophical' task of describing the actions that *could* have led to some future result, $\mathbf{y}$. Mostly, we will query the system about what future result should follow the actions it just saw. The use of probabilistic techniques here allows the notion of randomness and stochasticity which are more appropriate for synthesizing compelling behaviour. In essence, they make the system generate behaviour that is interesting and correlated with the past and the user's stimulating actions but is *also* not entirely predictable and contains some pseudo random choices in its space of valid responses.

Chapter 5

Learning: Conditional vs. Joint

Beyond EM: Generalized Bound Maximization, Conditional Densities and the CEM Algorithm

In the following chapters, we present a section dedicated to the machine learning aspect of this work. The concept of learning and probabilistic estimation is central to the Action-Reaction Learning paradigm and framework. The objective is to learn a mapping between the past interaction ($\mathbf{x}$) and the most likely reaction ($\mathbf{y}$) of humans from video data. This machine learning problem requires special attention and we describe a novel approach to resolving it. The contribution here is to stress that Action-Reaction Learning is best considered as a discriminatory learning problem and we show a formal Bayesian description of it. Essentially, we would like to estimate and use a conditional probability, $p(\mathbf{y}|\mathbf{x})$. However, traditional tools such as the EM (Expectation Maximization) [15] algorithm do not apply. Therefore, we instead propose an optimization framework and derive the CEM (Conditional Expectation Maximization) algorithm to resolve the problem.

We discuss important differences between conditional density estimation and conditioned joint density estimation in practical machine learning systems and at a Bayesian level. This motivates the use of direct conditional estimation and maximum conditional likelihood. To this end, we introduce and argue favorably for the generalized bound maximization (GBM) framework. The approach encompasses a variety of bounding methods that are useful for many optimization and learning problems. In particular, we apply these techniques to develop the Conditional Expectation Maximization (CEM) algorithm which maximizes conditional likelihood. The GBM techniques extend the bounds computed by the EM algorithm and offer reliable optimization and convergence. In addition, the bounding techniques are more general and can be applied to functions which can not be posed as maximum-likelihood incomplete-data problems. Essentially, we reformulate gradient ascent steps as optimization of a bound and avoid ad hoc estimation of step size. The approach is guaranteed to maximize the given function at each step and local convergence is proven. Unlike gradient techniques, the GBM framework is amenable to deterministic annealing and can be used to search for global optima. Results are shown for conditional density estimation as well as nonlinear function optimization.

5.1 Machine Learning: Practical Considerations

There are many different issues that need to be addressed before implementing a machine learning system. A large number of approaches exist and a reasonable one must be selected depending on factors such as the type of problem at hand and the practical complexity constraints of its implementation. In addition, sometimes an approach is provably inferior or a mere subset of a more general framework. Many pitfalls have been avoided, for example, when reliable statistical approaches were used instead of adhoc methods. Of course, a full outline of the issues in machine learning is beyond the scope of this document. However, we shall carefully discuss one critical decision: the use of model-based versus model-free learning. In statistical terms, this refers to the estimation of joint densities versus conditional densities. The list below outlines this and other important machine learning concepts that will be referred to in this document.

- Problem: Conditional vs. Conditioned Joint Densities

  Before using learning techniques, it is critical to identify the exact problem to be solved: are we trying to build a description of some phenomenon or are we observing a phenomenon to subsequently learn how to perform a particular task? In probabilistic terms, are we trying to estimate a joint density of all variables or is a conditional density ultimately desired? The former is called a generative model ('model-based') since it can simulate the whole phenomenon while the latter is a discriminative model (or 'model-free') since it models the phenomenon only enough to accomplish a particular task. For discriminative models, we refer to a system with the 'task' of computing an output $\mathbf{y}$ given an input $\mathbf{x}$. Generative models do not explicitly partition output and input but simply describe $(\mathbf{x}, \mathbf{y})$ as a joint phenomenon.

- Features

  Learning requires data and the learning system is typically fed variables that have been pre-selected and pre-processed. Feature selection (and re-parameterization) is often critical to ensure that the machine learning algorithm does not waste resources, get distracted by spurious features or tackle needlessly high dimensionalities. However, it is advantageous if a learning sytem does not depend excessively on feature parametrization.

- Model

  It is always necessary to identify the type of model that will be used to learn from data. Many statistical and deterministic approaches exist varying from Decision Trees, Multivariate Densities, Hidden Markov Models, Mixture Models, Neural Networks, etc. Ideally, a general estimation framework (such as statistical techniques) is desired which can cope with such a variety of models.

- Structure

  Certain parameters or attributes of the model will be predetermined by the user who can select them before training on data. Often, these include complexity, constraints, topology and so on. It is desirable that these decisions are not too critical and a learning system should solve its task despite having been given an incorrect structure or inappropriate complexity.

- Inference

  The remaining model parameters the user does not specify can then be estimated by observing data. Typically, the criteria for estimating these parameters include (in increasing degree of sophistication) Least-Square Error, Maximum Likelihood, Maximum A Posteriori, Bayesian Integration and the variants thereof. More sophisticated inference techniques will often converge to more global and more generalizable learning solutions.

- Algorithms

  Typically, a variety of algorithms can be called upon to perform the desired inference. These include searching, sampling, integration, optimization, EM, gradient ascent and approximation methods. Often, these algorithms and their effectiveness will greatly influence the performance of the resulting learning system.

5.2 Joint versus Conditional Densities - Pros and Cons

Recently, Hinton [23] and others proposed strong arguments for using model-based approaches in classification problems. However, in certain situations, the advantages of model-based approaches dim in comparison with performance of discriminative models optimized for a given task. The following list summarizes some advantages of generative models and joint density estimation for the purposes of both classification *and* regression problems.

- Better Inference Algorithms

  Generative models and joint densities can be computed using reliable techniques for maximum likelihood and maximum a posteriori estimation. These joint density estimation techniques include the popular EM algorithm and typically outperform gradient ascent algorithms which are the workhorses of conditional density problems (i.e. in many Neural Networks). The EM algorithm provably converges monotonically to a local maximum likelihood solution and often is more efficient than gradient descent.

- Modular Learning

  In a generative model, each class is learned individually and only considers the data whose labels correspond to it. The model does not focus upon inter-model discrimination and avoids considering the data as whole. Thus the learning is simplified and the algorithms proceeds faster.

- New Classes

  It is possible to learn a new class (or retrain an old class) without updating the models of previous learned classes in a generative model since each model is trained in isolation. In discriminative models, the whole system must be retrained since inter-model dynamics are significant.

- Missing Data

  Unlike conditional densities (discriminative models), a joint density or generative model is optimized over the whole dimensionality and thus models all the relationships between the variables in a more equal manner. Thus, if some of the data that was expected to be observed for a given task is missing, a joint model's performance will degrade gracefully. Conditional models (or discriminative models) are trained for a particular task and thus a different model must be trained for missing data tasks. Gharamani et al [21] point out the exponential complexity growth of the number of models needed if one needs an optimal discriminative system for each possible task.

- Rejection of Poor or Corrupt Data

  Sometimes, very poor data could be fed into the learning system and a generative model has the ability to detect this corrupt input and possibly signal the user to take some alternate measure.

It is important to note that the last two advantages occur infrequently in many applications and is the expected situation for the ARL framework. Typically, the system is called upon to perform the task it was trained for. Thus, the benefits of its superior performance over occasional missing data and poor data might rarely be noticed. In fact, on most standardized databases, the performance on the desired task will often be orders of magnitude more critical due to the infrequency of missing data or corrupt data.

The second and third advantages involve computational efficiency since discriminative or conditional models need to observe all data to be optimally trained. However, the need to observe all the data is not a disadvantage but truly an advantage. It allows a discriminative model to better learn the interactions between classes and their relative distributions for discrimination. Thus, as long as the discriminative model is not too computationally intensive and the volume of data is tractable, training on all the data is not a problem.

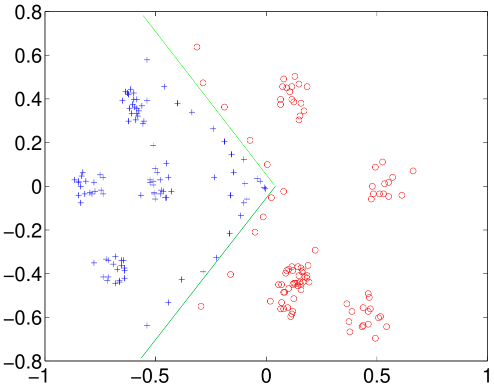


Figure 5.1: Simple Discriminative Model - Complex Generative Model

The most critical motivation for generative models in regression problems is actually the first advantage: the availability of superior inference algorithms (such as EM [15]). Typically, the training process for discriminative models (i.e. conditional densities) is cumbersome (i.e. neural network backpropagation and gradient ascent) and somewhat adhoc, requiring many re-initializations to converge to a good solution. However, tried and true algorithms for generative models (joint density estimation) avoid this and consistenly yield good joint models.

In fact, *nothing* prevents us from using both a generative model and a discriminative model. Whenever the regular task is required and data is complete and not corrupt, one uses a superior discriminative model. Whenever missing data is observed, a joint model can be used to 'fill it in' for the discriminative model. In addition, whenever corrupt data is possible, a *marginal* model should be used to filter it (which is better than a joint and a conditional model for this task). However, the bulk of the work the learning system will end up doing in this case will be performed by the conditional model.

We now outline specific advantages of conditional models and discuss our approach to correct one of their major disadvantages: poor inference algorithms.

- Management of Limited Resources

  Conditional or discriminative models utilize resources exclusively for accomplishing the task of estimating output from input observations. Thus, the limited resources (complexity, structures, etc.) will be used exclusively for this purpose and not squandered on irrelevant features that give no discrimination power.

- Simple Discrimination of Complex Generative Models

  It is often the case that complex joint (generative) models can be easily separated by simple decision boundaries as in Figure 5.1. There is no need here to model the intrinsically complex phenomena themselves when it is so simple to discriminate the two different classes with two linear boundaries.

- Feature Selection

  Conditional models by default do not need features that are as well chosen as joint models. Since spurious features will not help the discriminant model compute its output, they will be effectively ignored by it. A generative model might waste modeling power on these features even though they ultimately offer no discrimination power.

- Better Conditional Likelihood on Test Data

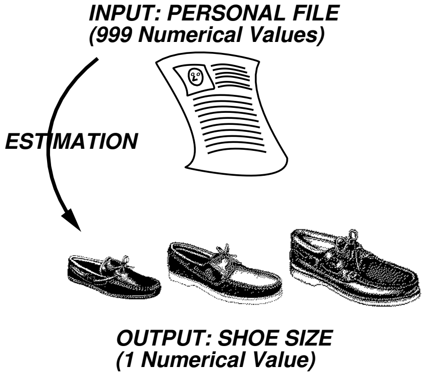


Figure 5.2: Large Input Data, Small Output Estimation

Typically, a learning system is trained on training data and tested on test data. Since in testing (either for joint or conditional models) we are always evaluating conditional likelihood (i.e. probability of guessing the correct class or guessing the right output) it is only natural that a model which optimizes this ability on training data will do better when tested (unless overfitting occurs).

5.2.1 Squandered Resources: The Shoe Salesman Problem

We shall now describe an entertaining estimation problem and see how it argues convincingly for the conditional (discriminant) models instead of joint (generative) models. Let us consider the case of a shoe salesman who has to learn how to fit his clients with shoes of the right size. Each client has a file containing all their personal information *except shoe size*. The shoe salesman has access to this data and knows a variety of things about his clients. Some of these features include: hair color, eye color, height, weight, sex, age, etc. In fact, let us say that each personal file contains 999 numerical values (features) describing one client. Of course, shoe size is not in the file and the first few hundred clients that visit the salesman get their shoe size measured directly using a ruler. However, the salesman (for whatever reason) finds this a tedious process and wishes to automate the shoe size estimation. He decides to do this by learning what shoe size to expect for a client exclusively from the personal file. Assume that each client mails the shoe salesman his personal file before coming in to get new shoes.

Figure 5.2 depicts this learning problem. Given the 999 measurements (called $\mathbf{x}$) (which will always be sent to the salesman), what is the correct shoe size ($\mathbf{y}$)? The problem with fitting a generative model to this data is that it will make no distinction between $\mathbf{x}$ and $\mathbf{y}$. These variables are simply clumped together as measurements in a big joint space and the model tries to describe the overall phenomenon governing the features. As a result, the system will be swamped and learn useless interactions between the 999 variables in $\mathbf{x}$ (the file). It will not focus on learning how to use the $\mathbf{x}$ to compute $\mathbf{y}$ but instead equally worry about estimating any variable in the common soup from any possible observation. Thus, the system will waste resources and be distracted by spurious features that have nothing to do with shoe size estimation (like hair and eye color) instead of optimizing the ones that do (height, age, etc.).

A misallocation of resources occurs when a generative model is used as in Figure 5.3. Note how the model would place different linear sub-models to describe the interaction between hair darkness and eye darkness. This is wasteful for the application since these attributes have nothing to do with shoe size as can be seen by the lack of correlation between hair color and shoe size. In fact, only observing the height of the client seems to be good to determine the shoe size. The misallocation of resources is a symptom of the joint density estimation in $p(\mathbf{x}, \mathbf{y})$ which treats all variables equally. A discriminatory or conditional model that estimates $p(\mathbf{y}|\mathbf{x})$ does not use resources for estimation of hair darkness or some useless feature.

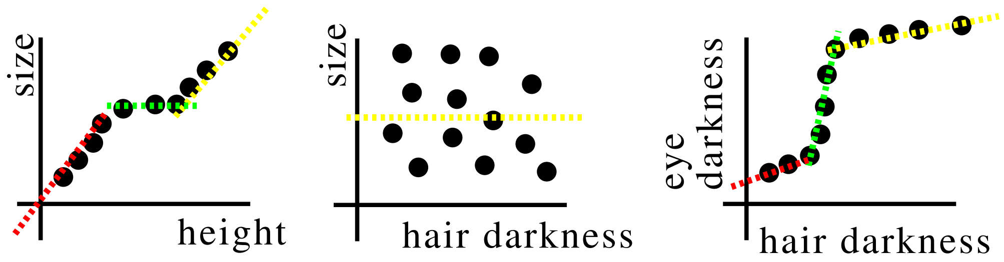


Figure 5.3: Distracting Features

Instead, only the estimation of $\mathbf{y}$, the shoe size, is of concern. Thus, as the dimensionality of $\mathbf{x}$ increases (i.e. 999), more resources are wasted modeling other features and performance on $\mathbf{y}$ usually degrades even though information is being *added* to the system. However, if it is made explicit that only the $\mathbf{y}$ output is of value, additional features in $\mathbf{x}$ should be more helpful than harmful.

We will now show that conditional or discriminative models fit nicely into a fully probabilistic framework. The following chapters will outline a monotonically convergent training and inference algorithm (CEM, a variation on EM) that will be derived for conditional density estimation. This overcomes some of the adhoc inference aspects of conditional or discriminative models and yields a formal, efficient and reliable way to train them. In addition, a probabilistic model allows us to manipulate the model after training using principled Bayesian techniques.

5.3 Conditional Densities: A Bayesian Framework

In Bayesian inference, the probability density function of a vector $\mathbf{z}$ is typically estimated from a training set of such vectors $\mathcal{Z}$ as shown in Equation 5.1 [5].

$$p(\mathbf{z}|\mathcal{Z}) = \int p(\mathbf{z}, \Theta|\mathcal{Z})d\Theta = \int p(\mathbf{z}|\Theta, \mathcal{Z})p(\Theta|\mathcal{Z})d\Theta \tag{5.1}$$

By integrating over Θ, we are essentially integrating over all the pdf models possible. This involves varying the families of pdfs *and* all their parameters. However, often, this is impossible and instead a family is selected and only its parametrization Θ is varied. Each Θ is a parametrization of the pdf of $\mathbf{z}$ and is weighted by its likelihood given the training set. However, computing the integral [1] is not always straightforward and Bayesian inference is approximated via maximum a posteriori (MAP) or maximum likelihood (ML) estimation as in Equation 5.2. The EM algorithm is frequently utilized to perform these maximizations.

$$p(\mathbf{z}|\mathcal{Z}) \approx p(\mathbf{z}|\Theta^*, \mathcal{Z}) \text{ where } \Theta^* = \begin{cases} \arg\max p(\Theta|\mathcal{Z}) = \arg\max p(\mathcal{Z}|\Theta)p(\Theta) & \text{MAP} \\ \arg\max p(\mathcal{Z}|\Theta) & \text{ML} \end{cases} \tag{5.2}$$

5.3.1 Conditional Density Estimation

Having obtained a $p(\mathbf{z}|\mathcal{Z})$ or, more compactly a $p(\mathbf{z})$, we can compute the probability of any point $\mathbf{z}$ in the vector space[2]. However, evaluating the pdf in such a manner is not necessarily the ultimate objective. Often, some components of the vector are given as input ($\mathbf{x}$) and the learning system is required the estimate the missing components as output[3] ($\mathbf{y}$). In other words, $\mathbf{z}$ can be broken up into two sub-

[1] Either analytically or by sampling techniques
[2] This is the typical task of unsupervised learning.
[3] This is the typical task of supervised learning.

vectors $\mathbf{x}$ and $\mathbf{y}$ and a conditional pdf is computed from the original joint pdf over the whole vector as in Equation 5.3. This conditional pdf is $p(\mathbf{y}|\mathbf{x})^j$ with the j superscript to indicate that it is obtained from the previous estimate of the joint density. When an input $\mathbf{x}'$ is specified, this conditional density becomes a density over $\mathbf{y}$, the desired output of the system. This density is the required function of the learning system and if a final output estimate $\hat{\mathbf{y}}$ is need, the expectation or arg max can be found via Equation 5.4.

$$p(\mathbf{y}|\mathbf{x})^j = \frac{p(\mathbf{z})}{\int p(\mathbf{z})d\mathbf{y}} = \frac{p(\mathbf{x},\mathbf{y})}{\int p(\mathbf{x},\mathbf{y})d\mathbf{y}} = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})} = \frac{\int p(\mathbf{x},\mathbf{y}|\Theta)p(\Theta|\mathcal{X},\mathcal{Y})d\Theta}{\int p(\mathbf{x}|\Theta)p(\Theta|\mathcal{X},\mathcal{Y})d\Theta} \tag{5.3}$$

$$\hat{\mathbf{y}} = \left\{ \begin{array}{l} \arg\max p(\mathbf{y}|\mathbf{x}') \\ \int \mathbf{y}p(\mathbf{y}|\mathbf{x}')d\mathbf{y} \end{array} \right. \tag{5.4}$$

Obtaining a conditional density from the unconditional (i.e. joint) probability density function in such a roundabout way can be shown to be suboptimal. However, it has remained popular and is convenient partly because of the availability of powerful techniques for joint density estimation (such as EM).

If we know a priori that we will need the conditional density, it is evident that it should be estimated *directly* from the training data. Direct Bayesian conditional density estimation is defined in Equation 5.5. The vector $\mathbf{x}$ (the input or *covariate*) is always given and the $\mathbf{y}$ (the output or *response*) is to be estimated. The training data is of course also explicitly split into the corresponding $\mathcal{X}$ and $\mathcal{Y}$ vector sets. Note here that the conditional density is referred to as $p(\mathbf{y}|\mathbf{x})^c$ to distinguish it from the expression in Equation 5.3.

$$\begin{array}{ll} p(\mathbf{y}|\mathbf{x})^c & = p(\mathbf{y}|\mathbf{x},\mathcal{X},\mathcal{Y}) \\ & = \int p(\mathbf{y},\Theta^c|\mathbf{x},\mathcal{X},\mathcal{Y})d\Theta^c \\ & = \int p(\mathbf{y}|\mathbf{x},\Theta^c,\mathcal{X},\mathcal{Y})p(\Theta^c|\mathbf{x},\mathcal{X},\mathcal{Y})d\Theta^c \\ & = \int p(\mathbf{y}|\mathbf{x},\Theta^c)p(\Theta^c|\mathcal{X},\mathcal{Y})d\Theta^c \end{array} \tag{5.5}$$

Here, Θ^c parametrizes a conditional density $p(\mathbf{y}|\mathbf{x})$. Θ^c is exactly the parametrization of the conditional density $p(\mathbf{y}|\mathbf{x})$ that results from the joint density $p(\mathbf{x},\mathbf{y})$ parametrized by Θ. Initially, it seems intuitive that the above expression should yield exactly the same conditional density as before. It seems natural that $p(y|x)^c$ should equal $p(y|x)^j$ since the Θ^c is just the conditioned version of Θ. In other words, if the expression in Equation 5.1 is conditioned as in Equation 5.3, then the result in Equation 5.5 should be identical. This conjecture is wrong.

Upon closer examination, we note an important difference. The Θ^c we are integrating over in Equation 5.5 is not the same Θ as in Equation 5.1. In the direct conditional density estimate (Equation 5.5), the Θ^c only parametrizes a conditional density $p(\mathbf{y}|\mathbf{x})$ and therefore provides no information about the density of $\mathbf{x}$ or $\mathcal{X}$. In fact, we can *assume* that the conditional density parametrized by Θ^c is just a function over $\mathbf{x}$ with some parameters. Therefore, we can essentially ignore any relationship it could have to some underlying joint density paramtrized by Θ. Since this is only a conditional model, the term $p(\Theta^c|\mathcal{X},\mathcal{Y})$ in Equation 5.5 behaves differently than the similar term $p(\Theta|\mathcal{Z}) = p(\Theta|\mathcal{X},\mathcal{Y})$ in Equation 5.1. This is illustrated in the manipulation involving Bayes rule shown in Equation 5.6.

$$\begin{array}{ll} p(\Theta^c|\mathcal{X},\mathcal{Y}) & = \frac{p(\mathcal{Y}|\Theta^c,\mathcal{X})p(\Theta^c,\mathcal{X})}{p(\mathcal{X},\mathcal{Y})} \\ & = \frac{p(\mathcal{Y}|\Theta^c,\mathcal{X})\mathbf{p}(\mathcal{X}|\boldsymbol{\Theta}^{\mathbf{c}})p(\Theta^c)}{p(\mathcal{X},\mathcal{Y})} \\ & = \frac{p(\mathcal{Y}|\Theta^c,\mathcal{X})\mathbf{p}(\mathcal{X})|p(\Theta^c)}{p(\mathcal{X},\mathcal{Y})} \end{array} \tag{5.6}$$

In the final line of Equation 5.6, an important manipulation is noted: $p(\mathcal{X}|\Theta^c)$ is replaced with $p(\mathcal{X})$. This implies that observing Θ^c does not affect the probability of $\mathcal{X}$. This operation is invalid in the joint density estimation case since Θ has parameters that determine a density in the $\mathcal{X}$ domain. However, in

(a) Joint Density Estimation (b) Conditional Density Estimation

Figure 5.4: The Graphical Models

conditional density estimation, if $\mathcal{Y}$ is not also observed, Θ^c is independent from $\mathcal{X}$. It in no way constrains or provides information about the density of $\mathcal{X}$ since it is merely a conditional density over $p(\mathbf{y}|\mathbf{x})$. The graphical models in Figure 5.4 depict the difference between joint density models and conditional density models using a directed acyclic graph [36] [29]. Note that the Θ^c model and the $\mathcal{X}$ are independent if $\mathcal{Y}$ is not observed in the conditional density estimation scenario. In graphical terms, the Θ joint parametrization is a parent of the children nodes $\mathcal{X}$ and $\mathcal{Y}$. Meanwhile, the conditional parametrization Θ^c *and* the $\mathcal{X}$ data are co-parents of the child $\mathcal{Y}$ (they are marginally independent). Equation 5.7 then finally illustrates directly estimated conditional density solution $p(\mathbf{y}|\mathbf{x})^c$.

$$\begin{array}{ll} p(\mathbf{y}|\mathbf{x})^c & = \int p(\mathbf{y}|\mathbf{x},\Theta^c) p(\Theta^c|\mathcal{X},\mathcal{Y}) d\Theta^c \\ & = \int p(\mathbf{y}|\mathbf{x},\Theta^c) \frac{p(\mathcal{Y}|\Theta^c,\mathcal{X}) p(\mathcal{X}) p(\Theta^c)}{p(\mathcal{X},\mathcal{Y})} d\Theta^c \\ & = \int p(\mathbf{y}|\mathbf{x},\Theta^c) p(\mathcal{Y}|\Theta^c,\mathcal{X}) p(\Theta^c) d\Theta^c \quad / \, p(\mathcal{Y}|\mathcal{X}) \end{array} \tag{5.7}$$

The Bayesian integration estimate of the conditional density appears to be different and inferior from the conditional Bayesian integration estimate of the unconditional density. [4] The integral (typically) is difficult to evaluate. The corresponding conditional MAP and conditional ML solutions are given in Equation 5.8.

$$p(\mathbf{y}|\mathbf{x})^c \approx p(\mathbf{y}|\mathbf{x},\Theta^*) \text{ where } \Theta^* = \left\{ \begin{array}{ll} \arg\max p(\mathcal{Y}|\Theta^c,\mathcal{X}) p(\Theta^c) & \mathrm{MAP}^c \\ \arg\max p(\mathcal{Y}|\Theta^c,\mathcal{X}) & \mathrm{ML}^c \end{array} \right. \tag{5.8}$$

At this point, the reader is encouraged to read the Appendix for an example of conditional Bayesian inference ($p(\mathbf{y}|\mathbf{x})^c$) and how it differs from conditioned joint Bayesian inference ($p(\mathbf{y}|\mathbf{x})^j$). From this example we note that (regardless of the degree of sophistication of the inference) direct conditional density estimation is different and superior to conditioned joint density estimation. Since in many applications, full Bayesian integration is computationally too intensive, the ML^c and the MAP^c cases derived above will be emphasized. In the following, we shall specifically attend to the conditional maximum likelihood case (which can be extended to the MAP^c) and see how General Bound Maximization (GBM) techniques can be applied to it. The GBM framework is a set of operations and approaches that can be used to optimize a wide variety of functions. Subsequently, the framework is applied to ML^c and MAP^c expressions that were advocated above to find their maximum. The result of this derivation is the Conditional Expectation Maximization (CEM) algorithm which will be the workhorse learning system we will be using for the ARL training data.

[4] A proof by example can be found in the appendix.

Chapter 6

Optimization - General Bound Maximization

6.1 Bounding for Optimization

An important tool in optimization is bounding [55]. Bounding can be used to compute upper or lower bounds on functions, approximation errors, etc. to tackle intractable maximization or minimization problems. The principles of bounding are central in variational methods which utilize bounds as approximation and optimization tools [54] [52]. Of particular relevance are applications in statistics and graphical model learning as discussed by Rustagi [54] and Jordan [30]. An interesting application is shown by Jaakkola [25] where bounds on Bayesian Networks are introduced to reduce intractable computations in a graphical model. We will discuss bounding techniques in a slightly different way, emphasizing quadratic bounds as opposed to the dual bounds typical of variational techniques and we occasionally refer back to some ideas used by classical variational methods in the literature.

Let us consider the usefulness of a bound for arbitrary function optimization. [1] Take a function, f, which maps a scalar, x, to a scalar, i.e. $f : \mathcal{R}^1 \rightarrow \mathcal{R}^1$. For bounding purposes, the function's domain need not be restricted to this space and can be directly extended to $\mathcal{R}^n$ and some non-Euclidean spaces as well. Now consider consider p which we shall call our contacting lower bound of f because of the properties in Equation 6.1.

$$\begin{array}{l} p(x) \leq f(x) \quad \forall x \\ p(x^*) = f(x^*) \end{array} \tag{6.1}$$

It can be shown that increasing p will also increase f as in Equation 6.2. This is a trivial proof however it guarantees that iteratively optimizing a function's bound is a useful alternative to directly optimizing the function. In the following, we can see that if we are currently operating at a point x^* on function f, maximizing the bound p underneath f will make us move to another locus, x^{**} where f is increased.

$$\begin{array}{ll} \text{if} & p(x^*) \leq p(x^{**}) \\ \text{then} & f(x^*) \leq f(x^{**}) \\ \text{since} & f(x^*) = p(x^*) \leq p(x^{**}) \leq f(x^{**}) \end{array} \tag{6.2}$$

[1] We shall focus on maximization in this document but most arguments should apply to minimization problems as well.

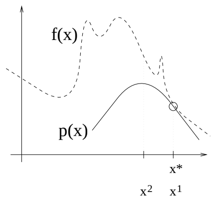


Figure 6.1: A Parabolic Bound on an Arbitrary Function

6.1.1 Quadratic Bounds

Let us now consider a simple form for the function p, a parabola (or paraboloid in higher dimensions) as in Equation 6.3.

$$\begin{array}{ll} p(x) = k - \frac{(x-y)^2}{\sigma} & x \epsilon \mathcal{R}^1 \\ p(\mathbf{x}) = k - (\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y}) & \mathbf{x} \epsilon \mathcal{R}^n \end{array} \tag{6.3}$$

In the above, the parabola contains a maximum which is the scalar value k. It also contains a locus for the maximum which is $\mathbf{y}$ (an element in n-dimensional Euclidean space). Finally, it contains a shape parameter, Σ (or σ in 1 dimension) which is a symmetric positive semi-definite matrix. There are simpler forms for the bound p (such as a constant or linear function) however the above bound is flexible because it has a single arbitrary maximum, k, at an arbitrary locus, $\mathbf{y}$. In addition, the maximization of a quadratic bound is straightforward and the maximization of a sum of quadratic bounds involves only linear computation. In addition, unlike variational bounds, the bounding principles used here do not necessarily come from the dual functions (i.e. linear tangents for logarithms, etc.) and are always concave paraboloids which are easy to manipulate and differentiate [54] [52]. Higher order functions and other forms which may be more flexible than quadratics are also usually more difficult to maximize.

6.2 Maximizing with Bounds: Beyond Gradient Ascent

Now consider the use of a bound such as the one proposed above. We have an analytically specified function, f, which we are trying to optimize starting from an initial point $x^* = x_1$. Figure 6.1 depicts the bound and the function to be optimized. Note the contact point where the function meets the bound is at x^*.

We can now find the maximum of that bound and move the contact point x^* to this new locus on the function. This point is now x_2 and is used to find another parabola which lies below the function f and touches it at our current operating point, x_2. As this process is iterated, we gradually converge to a maximum of the function. This process is illustrated in Figure 6.2. At the function's maximum, however, the only parabola we can draw which is a lower bound is one whose peak is at the function's maximum (i.e. a 'fixed' point). The algorithm basically stops there. Thus, there is a natural notion of a fixed point approach. Such an algorithm will always converge to a local maximum and stay there once locked on.

In the example, the algorithm skipped out of a local maximum on its way to its destination. This is not due to the fact that the local maximum is lower but due to the some peculiarity of the bounding approach and its robustness to very local attractor. The bound is shaped by the overall structure function which has a wider and higher local maximum at x^{final}. We will discuss this trade-off between the amplitude of a local maximum and its robustness (width of the basin) later on and their relationship to annealing and global optimization.

Note how these bounding techniques differ from gradient ascent which is a first order Taylor approximation. A second order Taylor approximation is reminiscent of Newton and Hessian methods which

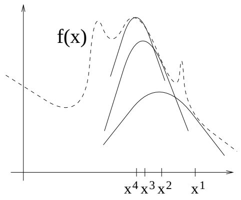


Figure 6.2: Bound Maximization Convergence

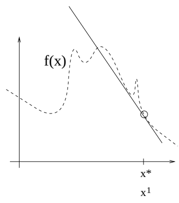


Figure 6.3: Gradient Ascent

contain higher order derivative data. However, both gradient ascent and higher order approximations to the functions are *not* lower bounds and hence using these to maximize the function is not always guaranteed to converge. Basically gradient ascent is a degenerate case of the parabolic bound approach where the width of the parabola is set to infinity (i.e. a straight line) and the step size chosen in an ad hoc manner by the user (i.e. infinitesimal) instead of in a more principled way. A parabolic bound's peak can be related to its width and if this value can be properly estimated, it will never fail to converge. However, selection of an arbitrary step size in gradient ascent can not be guaranteed to converge. Observe Figure 6.3 which demonstrates how an invalid step size can lead gradient approaches astray. In addition Figure 6.4 demonstrates how higher order methods can also diverge away from maxima due to adhoc step size. The parabola estimated using Taylor approximations is neither an upper nor a lower bound (rather it is a good approximation which is a different quality altogether). Typically, these techniques are only locally valid and may become invalid for significantly large step sizes. Picking the maximum of this parabola is not provably correct and again an ad hoc step size constraint is needed.

However, gradient and Taylor approximation approaches have remained popular because of their remarkable ease of use and general applicability. Bound techniques (such as the Expectation Maximization or EM algorithm [15]) have not been as widely applicable. This is because almost any analytic function can be differentiated to obtain a gradient or higher order approximation. However, we shall show some important properties of bounds which should illustrate their wide applicability and usefulness. In addition, some examples of nonlinear optimization and conditional density estimations will be given as

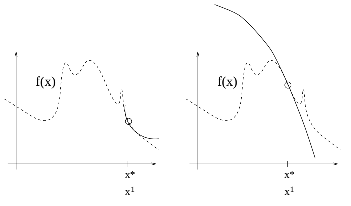


Figure 6.4: 2nd Order Approximations

demonstrations.

6.3 Placing and Tightening the Bound

In general, a lower bound has many parameters which allow it to support the function being optimized in different ways. We outline three desiderata that should be followed for placing the lower bound on the function. Typically, when we shall use bounding techniques in this document and when we discuss their further properties, we shall assume that these requirements are being met whenever possible.

- Requirement 1 $\longrightarrow$ Contact

  The bound must make contact with the function (Equation 6.4) at the current operating point for the maximization process to behave as described earlier.

$$\begin{array}{l} p(\mathbf{x}) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \\ p(\mathbf{x}^*) = f(\mathbf{x}^*) \end{array} \tag{6.4}$$

- Bound Requirement 2 $\longrightarrow$ Tangentiality

  The bound's tangent plane should be parallel to the function's tangent plane at the current operating point as in Equation 6.5. Otherwise, there will be an intersection between the function and the lower bound and the lower bound will rise above it, violating the lower bound assumption.

$$\frac{\partial p(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \text{ at } \mathbf{x} = \mathbf{x}^* \epsilon \mathcal{R}^{\mathrm{n}} \tag{6.5}$$

- Bound Requirement 3 $\longrightarrow$ Tightness

  Finally, the bound should be as 'close' to the function as possible to approximate the function properly. Typically, a lower bound which falls away from the function too quickly is not very likely to cause rapid optimization. In other words, what is desired is that the maximum of the bound is close to the maximum of the function. Let us focus on the quadratic bound described earlier (Equation 6.6).

$$\begin{array}{ll} p(x) = k - \frac{(x-y)^2}{\sigma} & x \epsilon \mathcal{R}^1 \\ p(\mathbf{x}) = k - (\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y}) & \mathbf{x} \epsilon \mathcal{R}^n \end{array} \tag{6.6}$$

  We use $\mathbf{y}$ to denote the location parameter which moves the locus of the parabola's maximum around. The k parameter varies the maximum's amplitude. Finally, note the Σ (or σ in 1 dimension). This is the scale (or shape) parameter which changes the shape and orientation of the parabola. We can also define the inverse of the scale parameter as $W = \Sigma^{-1}$ (or $w = \frac{1}{\sigma}$).

  Typically, the scalar parameter k and the location parameter $\mathbf{y}$ are determined directly from the constraints in Equation 6.4 and Equation 6.5. However, the scale parameter remains to be determined. Evidently, it must be chosen so that the lower bound property of the quadratic is not violated. However, we also would like to maximize the determinant of the scale parameter as given by Equation 6.7.

$$\max |\Sigma| \text{ s.t. } f(x) \geq k - (\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y}) \;\; \forall \mathbf{x} \epsilon \mathcal{R}^n \tag{6.7}$$

This statement is not as obvious as the previous two requirements and we justifiy it subsequently.

$$\begin{array}{l} p(\mathbf{x}) \leq f(\mathbf{x}) \quad \forall x \\ p(\mathbf{x}^*) = f(\mathbf{x}^*) = a \\ \frac{\partial p(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{b} \quad \text{at } \mathbf{x} = \mathbf{x}^* \end{array} \tag{6.8}$$

Suppose that Requirements 1 and 2 have been resolved and, specifically, we have the computed Equation 6.8 to obain real values (a and $\mathbf{b}$). These expressions can be manipulated further for the case of the quadratic as in Equation 6.9 yielding two constraint expressions (containing the constant terms a and $\mathbf{b}$).

$$\begin{array}{lcl} p(\mathbf{x}) & = & a \quad \text{at } \mathbf{x} = \mathbf{x}^* \\ p(\mathbf{x}) & = & k - (\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y}) \\ k & = & a + (\mathbf{x}^*-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}^*-\mathbf{y}) \\ & & \\ \frac{\partial p(x)}{\partial \mathbf{x}} & = & \mathbf{b} \quad \text{at } \mathbf{x} = \mathbf{x}^* \\ -2\Sigma^{-1}(\mathbf{x}^*-\mathbf{y}) & = & \mathbf{b} \\ \mathbf{y} & = & \frac{1}{2}\Sigma\mathbf{b} + \mathbf{x}^* \end{array} \tag{6.9}$$

The expressions in Equation 6.9 can then be used to isolate k and express it exclusively in terms of the constants $\mathbf{x}^*$, a and $\mathbf{b}$ as well as the still free shape parameter Σ. Note the isolated k in Equation 6.10 and its relationship to Σ (or W equivalently). As we increase $|\Sigma|$ (or decrease $|W|$), we are effectively increasing k. The larger $|\Sigma|$ yields high values of k. Since maximizing the quadratic bound will move our current locus from $\mathbf{x}^*$ to $\mathbf{y}$ the higher the $k = p(\mathbf{y}) \leq f(\mathbf{y})$ the higher the value of the function we *can* jump to. It is obvious that the widest parabola or maximal $|\Sigma|$ is desirable since it is more likely to allow higher jumps in the function being optimized. In reality, we would ideally like to maximize the expression $\mathbf{b}^T\Sigma\mathbf{b}$ over all the elements of the Σ. Thus the requirement for maximal Σ or maximal k can be interchanged.

$$\begin{array}{lcl} k & = & a + (\mathbf{x}^*-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}^*-\mathbf{y}) \\ k & = & a + \frac{1}{4}\mathbf{b}^T\Sigma\mathbf{b} \end{array} \tag{6.10}$$

In fact, we typically begin by finding the optimal Σ [2] and then isolating the corresponding k and $\mathbf{y}$. These three quantities are really functions of the operating point $\mathbf{x}^*$ (once the function to optimize, $f()$, is specified of course) and can be expressed as functions of each other as well. This concept is demonstrated in concrete examples that will follow.

6.4 General Bound Maximization and Properties

General bound maximization refers to the use of a bound along with some important manipulation techniques to reduce intractable optimization problems into more reasonable forms. We define a set of manipulations to facilitate bounding for a wide variety of functions f which may contain all sorts of arithmetic operations, power terms, and elementary functions. These manipulations can be applied iteratively and recursively to ultimately break down a complicated function into several simple bounds. These bounds can then be reassembled and used to perform a simple maximization as in the previous example, moving the current operating point x_1 to a better (higher) locus on the function, x_2 and iterating until convergence.

[2] In this sense, Σ acts like a variational parameter.

- Property 1 - Addition and Subtraction

  If a function involves a summation, the individual components of the function can be bounded separately and then the bounds can be summed to form a bound on the overall function (Equation 6.11).

$$\begin{array}{ll} f(x) = g(x) + h(x) & \\ & \\ \text{if} & p_g(x) \leq g(x) \\ \text{and} & p_h(x) \leq h(x) \\ \text{then} & f(x) \geq p(x) = p_g(x) + p_h(x) \end{array} \tag{6.11}$$

  For subtraction, we can just define $h(x)$ as $-h(x)$ and bound that quantity with $p_h(x)$.

- Property 2 - Multiplication and Division

  If a function involves a multiplication, the logarithm of the multiplied terms can be bounded and then the resulting bounds can again be summed to form a bound on the overall function (Equation 6.12). Since logarithms are being used, assume that f, g, h are positive functions. Here we specifically want to rearrange the bounding to be a sum of bounds for simplicity. Otherwise, it could be possible to just multiply two bounds and still have an overall bounding.

$$\begin{array}{ll} f(x) = g(x) \times h(x) & \\ f(x) \geq \log(f(x)) + 1 & \\ f(x) \geq \log(g(x) \times h(x)) + 1 & \\ f(x) \geq \log(g(x)) + \log(h(x)) + 1 & \\ & \\ \text{if} & p_g(x) \leq \log(g(x)) \\ \text{and} & p_h(x) \leq \log(h(x)) \\ \text{then} & f(x) \geq p(x) = p_g(x) + p_h(x) + 1 \end{array} \tag{6.12}$$

  For division, we can just define $log(h(x))$ as $-log(h(x))$ and bound that quantity with $p_h(x)$.

- Property 3 - Composition

  If an expression involves the composition of two functions, the expression as a whole can be bounded by the composing a bound on one function with the other sub-function. (Equation 6.13).

$$\begin{array}{ll} f(x) = g(h(x)) & \\ & \\ \text{if} & p_g(x) \leq g(x) \\ \text{then} & f(x) \geq p(x) = p_g(h(x)) \end{array} \tag{6.13}$$

  For example, consider $f(x) = \sin(\log(x))$ where $h(x) = \log(x)$ and $g(x) = \sin(x)$. Assume that we can bound g with a parabola $g(x) \geq k - w(x-y)^2$. Then function $f(x)$ can be bounded by $f(x) \geq k - w(log(x) - y)^2$. It is far simpler to take derivatives of the bounding function and set these to zero to iteratively optimize f than it is to directly solve for the maximum via $\frac{\partial f(x)}{\partial x} = 0$.

- Property 4 - Partial Linearity

  If a function involves a linear operation then its bounds can also be modified linearly as long as the scale factors are non-negative (Equation 6.14).

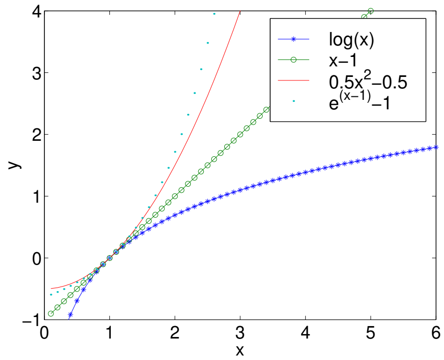


Figure 6.5: Bounding the logarithm

$$
\begin{array}{ll}
\multicolumn{2}{l}{f(x) = a \times g(x) + b} \\
\text{if} & a \geq 0 \\
\text{and} & p_g(x) \leq g(x) \\
\text{then} & f(x) \geq p(x) = a \times p_g(x) + b
\end{array}
\tag{6.14}
$$

In addition, a scaling operation on the domain can also be applied to the bound as well (Equation 6.15).

$$
\begin{array}{ll}
\multicolumn{2}{l}{f(x) = g(a \times x + b)} \\
\text{if} & p_g(x) \leq g(x) \\
\text{then} & f(x) \geq p(x) = p_g(a \times x + b)
\end{array}
\tag{6.15}
$$

- Property 5 - Jensen's Inequality

  A well known tool for bounding techniques is Jensen's inequality which is repeated below for convenience (Equation 6.16).

$$
\begin{array}{ll}
E[f(x)] \leq f(E[x]) & \text{if } f \text{ is concave} \\
E[f(x)] \geq f(E[x]) & \text{if } f \text{ is convex}
\end{array}
\tag{6.16}
$$

- Property 6 - Function Bounding

  Often, it is useful to replace a function in an expression with a lower bounding function which makes contact with it at the current operating point. Figure 6.5 depicts how the $\log(x)$ function can be replaced by upper bounds. In this figure, the current operating point is $x^* = 1$. The convex functions depicted are $p1(x) = x - 1$, $p2(x) = \frac{x^2}{2} - \frac{1}{2}$, and $p3(x) = e^{(x-1)} - 1$. This property will be very important later as we apply these techniques to conditional density estimation.

- Property 7 - Monotonic Transformations

Of course, there are many possible bounds to select for any optimization and it is difficult to find a single form for the bound which will always work. This is true even if we have a rather generic form such as a parabola. For instance, it is impossible to lower bound $f(x) = -x^4$ with a quadratic since the function will always be decreasing faster than any parabola. However, for many functions, it is possible to get around this by instead maximizing another function, say $h(x) = g(f(x))$ where g is a monotonically increasing function such as $g(x) = e^x$ which, when composed with f will have the same maxima and minima as f but will behave in a more controllable way. For this example, $h(x) = e^{f(x)} = e^{-x^4}$ results and this *can* be bounded with a parabola. Selecting the monotonic function to use in these situations is often intuitively obvious.

6.5 Optimization Examples

Naturally, the best way to visualize the above concepts is to apply the bound maximization technique to standard functions and see the results. Let us consider the trivial case when $f(x) = \sin(x)$ (a sinusoid). This function optimization is not interesting in itself but its power will be demonstrated later when we consider arbitrary summations of different sinusoids (i.e. as in a Fourier decomposition). We derive an algorithm for optimizing an arbitrary such decomposition to yield the maximum of any landscape described by sinusoids. The extension to multiple dimensions is also discussed.

We wish to bound the single sinusoid $f(x) = \sin(x)$ (i.e. a one dimensional function) with a parabola as in Equation 6.17. We shall satisfy the 3 requirements (contact, tangentiality and tightness) to derive the optimization algorithm.

$$\sin(x) \geq k - w(x-y)^2 \tag{6.17}$$

Next consider the contact point $x^* = z$ where the parabola and the sinusoid actually touch in Equation 6.18.

$$\begin{array}{rcl} \sin(z) & = & k - w(z-y)^2 \\ k & = & \sin(z) + w(z-y)^2 \end{array} \tag{6.18}$$

Subsequently, we also enforce the tangentiality requirement that the derivatives are equal at contact to avoid crossover (see Equation 6.19.

$$\begin{array}{rcl} \frac{\partial \sin(x)}{\partial x}|_{x=z} & = & \frac{\partial (k-w(x-y)^2)}{\partial x}|_{x=z} \\ y & = & z + \frac{\cos(z)}{2w} \end{array} \tag{6.19}$$

Inserting the results of Equation 6.19 and Equation 6.18 into the inequality in Equation 6.17 we get an expression for the shape parameter, $w = \frac{1}{\sigma}$ as in Equation 6.20. The equation is manipulated to isolate w onto one side of the inequality. *Any* w can be selected now, so long as it satisfies this inequality for the given z and for all x in the domain of the function. It is subsequently straightforward to algebraically compute y and k and get a valid parabolic bound on the sinusoid for the given contact point z.

Thus, the critical step still remains: how do we compute w? A few ad hoc ways of finding such a w directly exist although these do not give an optimal value of w. Recall, now, that we also have the tightness requirement in Equation 6.7 which seeks to compute the smallest $|w|$ or largest σ. Since z is given, we wish to compute the smallest possible w that will satisfy this inequality in Equation 6.20 for all x in the domain of the function. Ultimately, we wish to recover $w_{min}(z)$ which is the minimum value w can have for a given z before violating the inequality (crossing our function $f(x)$ in some locations, $x \epsilon \mathcal{R}^1$ and invalidating the bound).

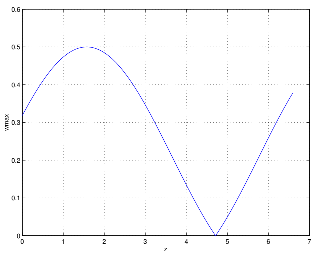


Figure 6.6: Parabolic width as a function of contact point z

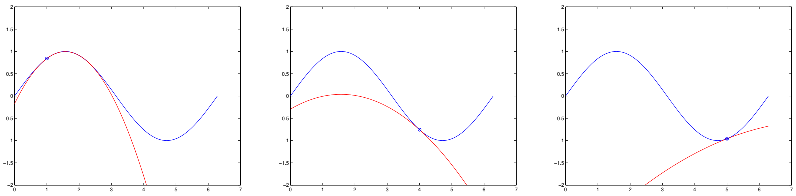


Figure 6.7: Examples of Parabolic Bounding of a Sinusoid

$$\begin{aligned} \sin(x) &\geq k - w(x-y)^2 \\ w &\geq \frac{\sin(z)-\sin(x)-z\cos(z)+x\cos(z)}{(x-z)(x-z)} \end{aligned} \tag{6.20}$$

$$w_{min}(z) = \max_x \frac{\sin(z)-\sin(x)-z\cos(z)+x\cos(z)}{(x-z)(x-z)} \tag{6.21}$$

It is difficult to find the function $w_{min}(z)$ analytically for a sinusoid. Thus, for each z, we solve for w_{min} numerically by searching for the maximum of the right hand side expression in Equation 6.21 over all x (an efficient 1D Brent's search method is used [49]). The function is then stored as a lookup table (LUT) and is plotted in Figure 6.6 for visualization.

Given a sinusoid $f(x) = sin(x)$ and a contact point z, we can immediately compute w_{min} from this LUT and then the corresponding y (which only depends on z and w) and finally the k. A few examples of the bounding are shown in Figure 6.7.

Evidently, maximizing these parabolas is trivial (the maximum is y) and if z is set to y, we can repeat the calculation for a new contact point and converge to the local maximum as shown in Figure 6.8 (in fact, in this case, we converge after only one iteration).

At this point, optimizing the sinusoid is by no means challenging and the answer could have been solved for analytically without any of the above. However, since we have solved for the bound as opposed to a direct solution, the properties outlined earlier allow us to reapply the above derivations and solve a broader class of problems.

6.5.1 Fourier Decomposition Optimization

Let us now consider an arbitrary function represented by its Fourier decomposition. This function, denoted $F(x)$, is a linear combination of sinusoids as in Equation 6.22.

$$\mathrm{F(x)} = \sum_{i=0}^{M} \alpha_i \sin(xi) + \beta_i \sin(xi + \tfrac{\pi}{2}) \tag{6.22}$$

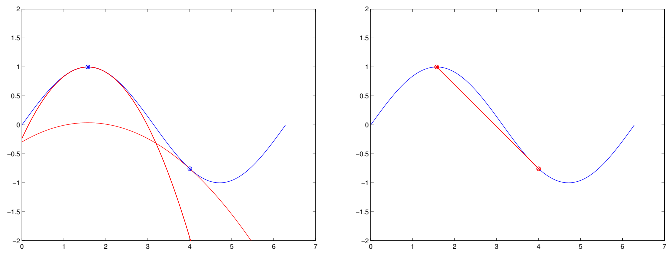
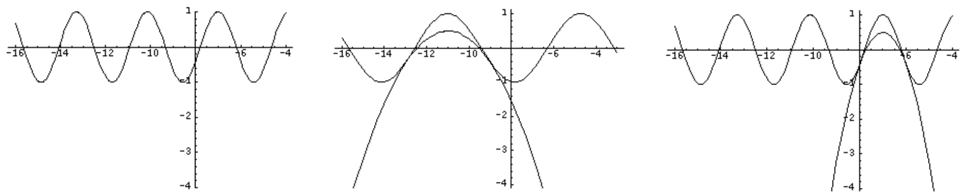


Figure 6.8: Convergence of Parabolic Bound for Sinusoid



(a) (b) (c)

Figure 6.9: Extending to arbitrary sinusoids

Since we have a way to solve for the quadratic bound for a canonical sinusoid $\sin(x)$ at a contact point z, we should be able to generalize this bound to all the individual terms in Equation 6.22 using some bound properties such as Equation 6.15. Each sinusoid in the linear combination in Equation 6.22 is of the form $\sin(ax+b)$ with $a > 0$ [3]. In addition, we need to modify the contact point z with which we will look up the values of w, k and y since these computations are particular to $\sin(x)$ and not $\sin(ax+b)$.

$$\begin{array}{lcll} \sin(x) & \geq & k - w(x-y)^2 & \text{with equality at } \; x = az+b \\ \sin(ax+b) & \geq & k - wa^2(x - \frac{y-b}{a})^2 & \text{with equality at } \; x = z \end{array} \tag{6.23}$$

Figure 6.9(a) depicts the function $f(x) = \sin(2x+3)$ $(a = 2, b = 3)$ which we wish to bound with a contact point at $x^* = z = -6$. We instead compute the parabolic bound for $\sin(x)$ at $x = (2*(-6)+3) = -9$ and obtain $w = 0.23$, $y = -10.98$ and $k = 0.49$. We can thus insert those quantities into the parabola expression and obtain the bound for the desired function $\sin(2x+3) > k - w4(x - \frac{y-3}{2})^2$ which is plotted in Figure 6.9 (c).

Using Equation 6.14 we can further manipulate each term by scaling it appropriately. Thus, we can get an expression for the lower bound on $F(x)$ in terms of a multitude of parabolic elements ($p_{2i}(x)$ and $p_{2i+1}(x)$).

$$\text{F(x)} \quad \geq \quad P(x) = \sum_{i=0}^{M} \alpha_i p_{2i}(x) + \beta_i p_{2i+1}(x) \tag{6.24}$$

Since it is necessary to now maximize the above sum of bounds, we can merely do so by taking the derivative of the sum with respect to x and setting that to 0. Maximizing the sum of quadratic terms thus merely reduces to the solution of a linear system. This is not quite as trivial as *choosing* the y locus

[3] Here, a is also only an integer traditionally but this is not necessary for the following derivation. Also note that a does not truly have to be positive since $\sin(-x) = sin(x+\pi)$ can be used to factor out the negativity.

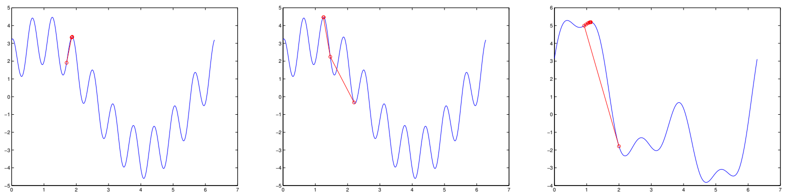


Figure 6.10: Convergence for Sum of Sinusoids

as the maximum for a single parabola but it is still extremely efficient. This operation is depicted in Equation 6.25. The x value computed is the locus of the maximum of the parabola. Thus, the equation implements one maximization step and yields the operating or contact point for the next iteration (for the bounding step). Repeating these two steps will converge to a local maximum of the function as in Figure 6.10.

$$\begin{aligned} \frac{\partial p(x)}{\partial x} &= \sum_{i=0}^{M} \alpha_i \frac{\partial p_{2i}(x)}{\partial x} + \beta_i \frac{\partial p_{2i+1}(x)}{\partial x} \\ \frac{\partial p(x)}{\partial x} &= 0 \\ x &= (\sum_{i=0}^{M} \alpha_i w_{2i} y_{2i} + \beta_i w_{2i+1} y_{2i+1})(\sum_{i=0}^{M} \alpha_i w_{2i} + \beta_i w_{2i+1})^{-1} \end{aligned} \tag{6.25}$$

Since we are able to optimize any combination of sinusoids with the above, and any arbitrary ($l2$-bounded) analytic function can be approximated by sinusoids (even in multiple dimensions), it is possible in principle to use the above derivation to optimize a wide variety of functions. These functions could also be non-analytic functions which are only sampled and approximated by sinusoids using least-squares. In addition, the above derivations for the canonical sinusoid $sin(x)$ function could have been done for another function such as radial basis functions, $exp(x)$, and so on and then generalized to linear combinations of this canonical functions. One small caveat exists when using linear combinations of arbitrary bounded functions: the negative of the function is bounded differently from the function itself. Thus, for arbitrary linear combinations, one needs to have a bound for canonical $f(x)$ and $-f(x)$ (since a bound flips from lower to upper bound due to the negative sign).

6.6 Beyond Local Optimization: Annealing the Bounds

A pivotal property of the above local maximization approach is the fact that it can be extended into a more global maximization algorithm. We shall derive explicit mechanisms for deterministically annealing the optimization such that it converges (asymptotically) to a more global maximum of a function from any local maximum attractor basin. Deterministic annealing has been applied in statistical approaches (i.e. for the EM algorithm [66]) very successfully and in general is better motivated than the ad hoc approaches associated with gradient ascent (such as momentum or weight decay). Although a detailed derivation and discussion of this mechanism and our usage of it is beyond the scope of this work, we shall discuss some of the implementation and motivation currently being considered.

An important insight can be gleaned from the deterministically annealed joint density estimation approaches and their recent success. Although the annealing approach hasn't quite become a standard mechanism in the learning community, it has been used more and more in very straightforward estimation problems (i.e. Gaussian mixture models mostly). Simply put, annealing in EM-type algorithms involves placing a distribution over the contact point (i.e. current operating point) x^*. Thus, the algorithm starts with a relatively flat distribution (corresponding to high temperature) indicating that it does not know where to initialize its local maximum search. Eventually, as the optimization iterates, this distribution gets 'tightened' and narrows towards a more localized x^* as the temperature is cooled and the algorithm behaves as a local optimizer. Thus, the optimization is less sensitive to the initialization point and the local maximum basin it happens to start in.

In the spirit of the above annealing process, we can imagine a different type of metaphor. Consider the function we are optimizing (i.e. $f(\mathbf{x})$) which is speckled with many local maxima. Now consider a heavily smoothed version of $f(\mathbf{x})$ which we shall call $f^t(\mathbf{x})$ (where t is proportional to the amount of smoothing). By smoothing, imagine flattening out the function and its many little local maxima by convolving it with, say, a Gaussian kernel. Evidently, after enough convolutions or smoothings, $f^t(\mathbf{x})$ will start looking like a single maximum function and more global maximization is possible. Thus, we can run a local optimization scheme on $f^t(\mathbf{x})$ and get to its single global maximum. As we reduce the degree of smoothness, $f^t(\mathbf{x})$ starts to look more and more like the original function $f(\mathbf{x})$ and we will begin optimizing the true function locally. However, the current operating point has now been placed in the basin of a more global maximum. It is important to note that here the term 'global maximum' is somewhat of a misnomer since a small 'peaky' global maximum might get ignored in favor of a wider maximum. This is due to the width of the effective basin around the sub-optimal maximum which has an effect on how dominant it remains after smoothing. The higher a maximum is and the more dominant and smooth the basin that surrounds it, the more likely it will remain after heavy smoothing.

Actually, it could be argued that this peculiarity of the 'smoothing based annealing' here is an advantage rather than a disadvantage. It is well known that robustness (of a maximum) is often a critical feature of optimization and estimation. By robustness we adopt the following definition: a robust maximum is a high maximum which is somewhat insensitive to perturbations in the locus or the parameters. This is similar to the Bayes' risk argument. Thus, the annealing process described here should move the optimization algorithm into a wider basin which is typically more robust. A sharp, peaky true global maximum might be the result of some artifact in the function being optimized (i.e. like overfitting in estimation problems). This kind of abnormality might not be typical of the landscape of the function and therefore should be ignored in favor of a wider more robust maximum.

Of course in practice, explicit smoothing of the function $f(\mathbf{x})$ is a cumbersome task either numerically or analytically. So, what we shall do is smooth the function *implicitly*. This is done by artificially widening the parabolas that are used to bound. In essence, we are *paying more attention to* wider parabolas in the optimization. To accomplish this, every w term or W matrix is taken to a power less than one (for higher temperatures, the power factor is closer to zero). This is shown more explicitly in Equation 6.26.

$$\begin{array}{rcl} w_{\text{annealed}} & = & w^{1/T} \\ W_{\text{annealed}} & = & W^{1/T} \end{array} \tag{6.26}$$

Of course, T is a temperature parameter which has to be varied using a pre-determined schedule. We will use the traditional exponential decay schedule which starts the temperature off near infinity and drops it towards $T = 1$. At infinity, the parabolas are effectively flat lines. Thus, the hypothetical function they bound is also flat and has in fact 0 local minima. At $T = 1$ the system reverts to the typical local maximization performance described earlier.

In Figure 6.11 some examples of the annealing are shown on combinations of sinusoids. Almost all the initialization points lead to convergence to the same global maximum. The exponential decay coefficient used was 0.4 and global convergence was obtained in about 5 iterations every time.

Effectively, the annealing technique considers local maxima as basins with a volume (in 1D this is just a height and width). The larger (wider and taller) basins will dominate as we smooth the function and force convergence towards a more global solution. Also note that deterministic annealing is quite different from simulated annealing. Only a few steps were required to achieve global convergence. In simulated annealing, random steps are taken to 'skip' out of local maxima in hopes of landing in a better basin. This is typically a less efficient search technique than a deterministic optimization approach.

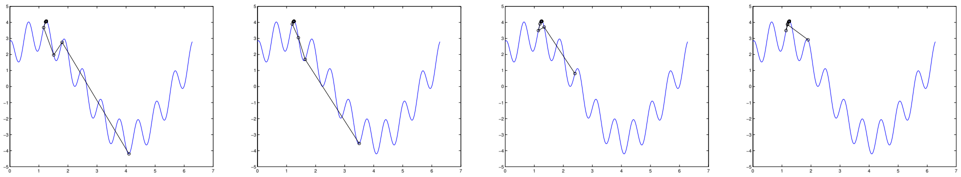


Figure 6.11: Annealed Convergence for Sum of Sinusoids

6.7 High Dimensions

At this point, it is important to note that the above derivations can be applied with little extra work to multi-dimensional optimization problems since they do not explicitly require 1D assumptions. Thus, they are inherently different from 1D search techniques such as bisection, Brent's method, etc. In the following chapter, we apply the above techniques to a specific multi-dimensional problem, conditional density estimation which is of particular interest. However, it is not the only place where the above work could be useful and many other possibilities remain to be investigated (some are now closer at hand than others).

Chapter 7

CEM - A Maximum Conditional Likelihood Learner

7.1 From EM to CEM

In this chapter, we discuss the extension of the EM (*Expectation Maximization*) algorithm for joint density estimation to conditional density estimation and derive the resulting CEM (*Conditional Expectation Maximization*) algorithm. The machine learning system that is central to the Action-Reaction Learning paradigm requires conditional densities to model interaction. Thus, the optimization techniques described in the preceding chapter will be called upon to formally derive the necessary algorithm. The use of CEM is demonstrated for a specific probabilistic model, a conditioned mixture of Gaussians, and implementation and machine learning issues are discussed.

In joint density estimation, the tried and true EM (expectation maximization) algorithm proceeds by maximizing likelihood over a training set. By introducing the concept of missing data, the algorithm breaks down what would be a complex density optimization into a two-step iteration. The missing unknown data components are estimated via the E-step and then a far simpler maximization over the complete data, the M-step, is performed. This is in fact identical to the iteration described earlier for General Bound Maximization. The E-step basically computes a bound (via Jensen's inequality) for the likelihood and the M-step maximizes that bound. Iteration of this process is guaranteed to converge to a local maximum of likelihood.

Recall the maximum likelihood joint density estimation (ML) case in Equation 5.2. Imagine we are faced with such a joint density optimization task and need to compute an optimal Θ over sets $\mathcal{X}$ and $\mathcal{Y}$ of N points as in Equation 7.1.

$$\begin{array}{rl} \arg\max p(\mathcal{X},\mathcal{Y}|\Theta) & = \arg\max \log(p(\mathcal{X},\mathcal{Y}|\Theta)) \\ & = \arg\max \sum_{i=1}^{N} \log(p(\mathbf{x}_i,\mathbf{y}_i|\Theta))) \end{array} \tag{7.1}$$

The EM algorithm is an iterative optimization which begins seeded at an initial starting point in the model space ($\Theta^{(t-1)}$). Instead of maximizing the expression in Equation 7.1, one can instead maximize its iterative form in Equation 7.2 where Θ^t is the algorithm's estimate for a better model from the previous estimate $\Theta^{(t-1)}$.

$$\begin{array}{l} \arg\max \log p(\mathcal{X},\mathcal{Y}|\Theta^t) - \log p(\mathcal{X},\mathcal{Y}|\Theta^{(t-1)}) \\ \quad = \arg\max \sum_{i=1}^{N} \log(p(\mathbf{x}_i,\mathbf{y}_i|\Theta)) - \log(p(\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})) \\ \quad = \arg\max \Delta l \end{array} \tag{7.2}$$

The density $p(\mathbf{x}_i, \mathbf{y}_i|\Theta)$ in general is not simple and is often better described by the discrete or continuous summation of simpler models. This is always true when data is missing. In other words, it is a mixture model as in Equation 7.3. The summation is performed over what is typically called the 'missing components' ($\mathbf{m}$ or m) which are individually more tractable. Thus, the original rather complex pdf can be considered to be a linear integration or summation of simpler pdf models.

$$p(\mathbf{x}_i, \mathbf{y}_i|\Theta) = \begin{cases} \int p(\mathbf{m}, \mathbf{x}_i, \mathbf{y}_i|\Theta)d\mathbf{m} & \text{Continuous } \mathbf{m} \\ \sum_{m=1}^{M} p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta) & \text{Discrete } m \end{cases} \tag{7.3}$$

7.1.1 Discrete Hidden Variables CEM

We shall now concentrate our attention on the case of discrete m (i.e. m takes on a discrete integer value from between $[1, M]$) which can be considered as an index into the pdf to select each of the individual simpler components. The continuous missing data case will be considered later.

The EM algorithm, in essence, uses Jensen's inequality to find a lower bound to the above quantity by taking advantage of the linear superposition qualities of the individual probability models. The log likelihood (the function being maximized) is manipulated using this simple bounding technique in Equation 7.4 [5] [74]. Jensen's inequality lower bounds the logarithm of the sum to make it easier to handle. The fundamental result is that the logarithm is applied to each simple model $p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta)$ individually instead of applying to the whole sum. It is then straightforward to compute the derivatives with respect to Θ and set those equal to zero to do the maximization.

$$\begin{aligned} \Delta l &= \sum_{i=1}^{N} \log(p(\mathbf{x}_i, \mathbf{y}_i|\Theta^t)) - \log(p(\mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})) \\ &= \sum_{i=1}^{N} \log \frac{\sum_{m=1}^{M} p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^t)}{\sum_{n=1}^{M} p(n, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})} \\ &\geq \sum_{i=1}^{N} \sum_{m=1}^{M} \frac{p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})}{\sum_{n=1}^{M} p(n, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})} \log \frac{p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^t)}{p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})} \end{aligned} \tag{7.4}$$

Let us now consider the case when we are estimating a conditional density [48]. Conditioning the discrete mixture model in Equation 7.3 results in Equation 7.5. This is a conditional density with missing data m which has been re-written as a joint density with missing data m over a marginal density with missing data m. We shall show that this missing data problem can not be solved using the standard EM approach.

$$p(\mathbf{y}_i|\mathbf{x}_i, \Theta) = \begin{cases} \int p(\mathbf{m}, \mathbf{y}_i|\mathbf{x}_i, \Theta)d\mathbf{m} = \frac{\int p(\mathbf{m}, \mathbf{x}_i, \mathbf{y}_i|\Theta)d\mathbf{m}}{\int p(\mathbf{m}, \mathbf{x}_i|\Theta)d\mathbf{m}} & \text{Continuous } \mathbf{m} \\ \sum_{m=1}^{M} p(m, \mathbf{y}_i|\mathbf{x}_i, \Theta) = \frac{\sum_{m=1}^{M} p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta)}{\sum_{m=1}^{M} p(m, \mathbf{x}_i|\Theta)} & \text{Discrete } m \end{cases} \tag{7.5}$$

The conditional log-likelihood utilizes this density model and is evaluated as in Equation 7.6. This is also an incremental expression (Δl^c) indicating that we are maximizing the improvement the model undergoes with each iteration (versus directly maximizing the model itself).

$$\begin{aligned} \Delta l^c &= \log p(\mathcal{Y}|\mathcal{X}, \Theta^t) - \log p(\mathcal{Y}|\mathcal{X}, \Theta^{(t-1)}) \\ &= \sum_{i=1}^{N} \log(p(\mathbf{y}_i|\mathbf{x}_i, \Theta^t)) - \log(p(\mathbf{y}_i|\mathbf{x}_i, \Theta^{(t-1)})) \\ &= \sum_{i=1}^{N} \log \frac{\sum_{m=1}^{M} p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^t)}{\sum_{m=1}^{M} p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})} - \log \frac{\sum_{n=1}^{M} p(n, \mathbf{x}_i|\Theta^t)}{\sum_{n=1}^{M} p(n, \mathbf{x}_i|\Theta^{(t-1)})} \\ &= \sum_{i=1}^{N} \log(R_j) - \log(R_m) \end{aligned} \tag{7.6}$$

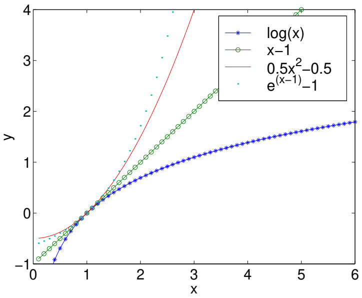


Figure 7.1: Bounding the Logarithm

Note that the above expression is the difference between the logarithms of two ratios (R_j and R_m). R_j is the ratio of the new joint over old joint and R_m is the ratio of the new marginal over the old marginal. In maximum likelihood however, only the first term is present (R_j). We could obtain a lower bound for it using Jensen's inequality. Since the log function is concave, Jensen will lower bound the log of a sum or expectation. However, in the maximum conditional likelihood situation, we subtract the second term (the ratio of marginals). Jensen can provide a lower bound for this second term however it is being subtracted. Since subtracting the second terms indicates that we are using the negative of the lower bound it instead acts as an upper bound. Thus, we will not be bounding conditional likelihood if we apply Jensen's inequality to the second term and can no longer guarantee convergence. Thus, let us refrain and only apply it to the first term ($\log R_j$) as in Equation 7.7.

$$\begin{array}{rcl} \Delta l^c & = & \sum_{i=1}^N \log \frac{\sum_{m=1}^M p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{\sum_{m=1}^M p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} - \log \frac{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^t)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^{(t-1)})} \\ & \geq & \sum_{i=1}^N \sum_{m=1}^M \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}{\sum_{n=1}^M p(n,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} \log \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} - \log \frac{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^t)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^{(t-1)})} \end{array} \tag{7.7}$$

As a side note we can see that the conditional density is tricky because it is the joint divided by the marginal (over x). For a mixture model, the numerator (the joint) is a linear combination of multiple models. However, the marginal is also a linear combination of models and it resides in the denominator. Thus, linear superposition occurring in joint mixture models does not hold when they are conditioned and things become more unmanageable. In fact, it seems that we simply can not pull the summation outside of the logarithm in the above expression to end up with an elegant maximization step.

At this point, if we were only considering EM approaches, we would typically resort to a Generalized Expectation Maximization approach (GEM) [15] which involves some gradient ascent techniques to make up for the inapplicability of EM. For instance, the Iteratively Re-Weighted Least Squares algorithm does such an operation in the Mixtures of Experts [32] [31] architecture. The estimation uses EM on a piece of the problem and then resorts to a gradient ascent approach in an inner loop which can not be posed as a typical EM iteration. This is still better than performing the whole conditional likelihood maximization completely with gradient ascent (as proposed by Bishop [5]). This is due to the extra convergence properties the EM-bound provides on top of the inner gradient ascent steps. However, it is not necessary to, dare we say, throw in the towel and resort to gradient ascent quite yet if more bounding techniques (i.e. other than EM and Jensen) can be first investigated. We shy away from the gradient ascent technique since it is often plagued with ad hoc step size estimation, slow non-monotonic convergence, and susceptibility to local maxima.

Recall earlier that we discussed the existence of numerous bounds for upper and lower bounding. It is important, in addition, that the bound chosen makes contact with the desired function at the current operating point. In the above case, the so-called current operating is $\Theta^{(t-1)}$, the previous state of the model we are optimizing. When we are at our current operating point (i.e. $\Theta^t = \Theta^{(t-1)}$), the ratio of the marginals between Θ^t and $\Theta^{(t-1)}$ must equal 1. Thus, the logarithm of the R_m ratio in the expression above *can* be upper bounded and thus the negative of the logarithm will be lower bounded. Many functions can used to upper bound the logarithm as shown again in Figure 7.1. The choices of the bounds include the following functions: $x-1$, $\frac{x^2}{2}-\frac{1}{2}$, and e^x-1. We choose the tightest convex upper bound on the log which is $x-1$. It also seems the simplest to manipulate. This manipulation removes the cumbersome logarithm of a summation and essentially decouples the sub-models that are added inside. We end up with Equation 7.8 which is much easier to manipulate. In fact, for our purposes, this expression will act as the analog of the $Q(\Theta^t, \Theta^{(t-1)})$ function for the EM algorithm.

$$\begin{aligned}
\Delta l^c &\geq \sum_{i=1}^N \sum_{m=1}^M \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}{\sum_{n=1}^M p(n,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} \log \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} - \log \frac{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^t)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^{(t-1)})} \\
\Delta l^c &\geq \sum_{i=1}^N \sum_{m=1}^M \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}{\sum_{n=1}^M p(n,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} \log \frac{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})} - \frac{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^t)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta^{(t-1)})} + 1 \\
\Delta l^c &\geq Q(\Theta^t, \Theta^{(t-1)})
\end{aligned} \tag{7.8}$$

At this point, we re-write a conditioned mixture model expression without *any* joint density models in Equation 7.9. This is similar to the Mixture of Experts notation [32] where many conditional models (called experts) are gated by marginal densities. This decomposition introduces an important flexibility in the conditional density: the marginal densities or gates need not integrate to one and can merely be arbitrary positive functions $f()$ instead of probability densities $p()$. This generalization does not violate the form of the conditional density in any way and only indicates that the gates need not be pdfs and need not integrate to 1 or any finite number. Thus, the gates can diverge, have discontinuities, grow towards infinity and so on (as long as they remain positive) and still maintain a legal conditional density. In effect, the conditional density parametrization can be manipulated more flexibly if it is not written with joint densities and marginals.

$$\begin{aligned}
p(\mathbf{y}_i|\mathbf{x}_i,\Theta) &= \frac{\sum_{m=1}^M p(m,\mathbf{x}_i,\mathbf{y}_i|\Theta)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta)} \\
&= \frac{\sum_{m=1}^M p(\mathbf{y}_i|m,\mathbf{x}_i,\Theta)p(m,\mathbf{x}_i|\Theta)}{\sum_{n=1}^M p(n,\mathbf{x}_i|\Theta)} \\
&= \frac{\sum_{m=1}^M p(\mathbf{y}_i|m,\mathbf{x}_i,\Theta)f(m,\mathbf{x}_i|\Theta)}{\sum_{n=1}^M f(n,\mathbf{x}_i|\Theta)}
\end{aligned} \tag{7.9}$$

Thus, Equation 7.8 can also be written as in Equation 7.10 in the experts-gates formalism. This formalism indicates that the parameters in the experts can often be re-written independently of the parameters of the gates. In other words, the conditional densities and the marginal densities (in appropriate coordinates) can be expressed with separate degrees of freedom. This allows us to decouple the conditional likelihood maximization further. The function Q here is a general bound for any conditional density. Thus, for the CEM algorithm, the computation of the Q function here is effectively the CE or Conditional Expectation step. The M-step, or Maximization, is specific to the density we are currently considering.

$$\begin{aligned}
Q(\Theta^t, \Theta^{(t-1)}) &= \sum_{i=1}^N \sum_{m=1}^M \frac{p(\mathbf{y}_i|m,\mathbf{x}_i,\Theta^{(t-1)})f(m,\mathbf{x}_i|\Theta^{(t-1)})}{\sum_{n=1}^M p(\mathbf{y}_i|n,\mathbf{x}_i,\Theta^{(t-1)})f(n,\mathbf{x}_i|\Theta^{(t-1)})} \\
&\quad \times \log \frac{p(\mathbf{y}_i|m,\mathbf{x}_i,\Theta^t)f(m,\mathbf{x}_i|\Theta^t)}{p(\mathbf{y}_i|m,\mathbf{x}_i,\Theta^{(t-1)})f(m,\mathbf{x}_i|\Theta^{(t-1)})} \\
&\quad - \frac{\sum_{n=1}^M f(n,\mathbf{x}_i|\Theta)}{\sum_{n=1}^M f(n,\mathbf{x}_i|\Theta^{(t-1)})} + 1
\end{aligned} \tag{7.10}$$

Although we are bounding conditional densities, note that we are also abiding by the convention that the marginal densities can be expressed as functions f since the integration to 1 over the $\mathbf{x}$ domain is not a requisite for conditional densities. Of course, we can, at any time, re-normalize the gates and express proper joint densities. Equation 7.11 depicts a more practical expression for the Q function where the computations of the CE step are completed and are used numerically. The M-step then involves taking derivatives of the Q function or applying further bounds to maximize the current Q. We consider a specific cases where the densities come from a particular family of models (i.e. conditioned Gaussian mixture models [64]) and see how the bounding can be applied further. This will be the M-step for the Gaussian mixture case. Similar derivations can be performed for other families.

$$\begin{array}{rcl} Q(\Theta^t, \Theta^{(t-1)}) & = & \sum_{i=1}^N \sum_{m=1}^M [\ \hat{h}_{im}(\log p(\mathbf{y}_i|m, \mathbf{x}_i, \Theta^t) + \log f(m, \mathbf{x}_i|\Theta^t) \\ & & - \log h_{im}) - r_i f(m, \mathbf{x}_i|\Theta)\ ] + 1 \\ \text{where} \quad h_{im} & = & p(\mathbf{y}_i|m, \mathbf{x}_i, \Theta^{(t-1)}) p(m, \mathbf{x}_i|\Theta^{(t-1)}) \\ \text{where} \quad \hat{h}_{im} & = & \frac{h_{im}}{\sum_{m=1}^M h_{im}} \\ \text{where} \quad r_i & = & \frac{1}{\sum_{n=1}^M p(n, \mathbf{x}_i|\Theta^{(t-1)})} \end{array} \tag{7.11}$$

7.1.2 Continuous Hidden Variables CEM

For completeness, we also derive the CEM algorithm for the continuous hidden variable case. In Equation 7.12 we present the analog to the above Q function bound on conditional likelihood for continuous hidden variables ($\mathbf{m}$). The derivation mirrors that of the discrete hidden variable case except that summations over the hidden data are replaced with integration. We apply Jensen's inequality and the linear upper bound on the logarithm to obtain a Q function as before.

$$\begin{array}{rcl} \Delta l^c & = & \log p(\mathcal{Y}|\mathcal{X}, \Theta^t) - \log p(\mathcal{Y}|\mathcal{X}, \Theta^{(t-1)}) \\ & = & \sum_{i=1}^N \log(p(\mathbf{y}_i|\mathbf{x}_i, \Theta^t)) - \log(p(\mathbf{y}_i|\mathbf{x}_i, \Theta^{(t-1)})) \\ & = & \sum_{i=1}^N \log \frac{\int p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^t)\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^t)\mathbf{dm}} - \log \frac{\int p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^{(t-1)})\mathbf{dm}} \\ & = & \sum_{i=1}^N \log \frac{\int p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^t)\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})\mathbf{dm}} - \log \frac{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^t)\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^{(t-1)})\mathbf{dm}} \\ & \geq & \sum_{i=1}^N \int \frac{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}{\int p(\mathbf{n},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})\mathbf{dn}} \log\left(\frac{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}\right)\mathbf{dm} - \frac{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^t)\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^{(t-1)})\mathbf{dm}} + 1 \\ Q(\Theta^t|\Theta^{(t-1)}) & = & \sum_{i=1}^N \int \frac{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}{\int p(\mathbf{n},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})\mathbf{dn}} \log\left(\frac{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^t)}{p(\mathbf{m},\mathbf{x}_i,\mathbf{y}_i|\Theta^{(t-1)})}\right)\mathbf{dm} - \frac{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^t)\mathbf{dm}}{\int p(\mathbf{m},\mathbf{x}_i,|\Theta^{(t-1)})\mathbf{dm}} + 1 \end{array} \tag{7.12}$$

7.2 CEM for Gaussian Mixture Models

Consider the Gaussian distribution in the mixture model in Figure 7.2. For the joint density case, Equation 7.13 depicts the mixture model. Here we are using the $\mathcal{N}$ definition to represent a multivariate normal (i.e. Gaussian) distribution. We can also consider an unnormalized Gaussian distribution $\hat{\mathcal{N}}$ shown in Equation 7.14. Equation 7.15 depicts the conditioned mixture model which is of particular interest for our estimation [64]. In Equation 7.15 we also write the conditioned mixture of Gaussians in an experts and gates notation and utilize unnormalized Gaussian gates.

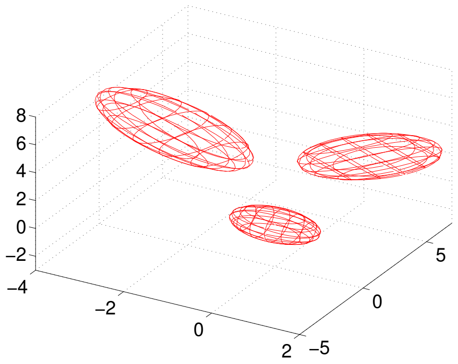


Figure 7.2: Mixture of 3 Gaussians

$$
\begin{aligned}
p(\mathbf{x},\mathbf{y}|\Theta) &= \sum_{m=1}^{M} \alpha_m \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} ; \begin{bmatrix} \mu_x^m \\ \mu_y^m \end{bmatrix} , \begin{bmatrix} \Sigma_{xx}^m & \Sigma_{xy}^m \\ \Sigma_{xy}^m & \Sigma_{yy}^m \end{bmatrix} \right) \\
&= \sum_{m=1}^{M} \alpha_m \frac{1}{2\pi^{D/2}\sqrt{\begin{vmatrix} \Sigma_{xx}^m & \Sigma_{xy}^m \\ \Sigma_{yx}^m & \Sigma_{yy}^m \end{vmatrix}}} e^{-\frac{1}{2} \begin{bmatrix} \mathbf{x}-\mu_x^m \\ \mathbf{y}-\mu_y^m \end{bmatrix}^T \begin{bmatrix} \Sigma_{xx}^m & \Sigma_{xy}^m \\ \Sigma_{yx}^m & \Sigma_{yy}^m \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}-\mu_x^m \\ \mathbf{y}-\mu_y^m \end{bmatrix}}
\end{aligned} \tag{7.13}
$$

$$
\hat{\mathcal{N}}(\mathbf{x};\mu_x,\Sigma_{xx}) \quad := \quad e^{-\frac{1}{2}(\mathbf{x}-\mu_x)^T \Sigma_{xx}^{-1} (\mathbf{x}-\mu_x)} \tag{7.14}
$$

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x},\Theta) &= \frac{\sum_{m=1}^{M} \alpha_m \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} ; \begin{bmatrix} \mu_x^m \\ \mu_y^m \end{bmatrix} , \begin{bmatrix} \Sigma_{xx}^m & \Sigma_{xy}^m \\ \Sigma_{xy}^m & \Sigma_{yy}^m \end{bmatrix} \right)}{\sum_{n=1}^{M} \alpha_n \mathcal{N}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n)} \\
&= \frac{\sum_{m=1}^{M} \alpha_n \mathcal{N}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n) \times \mathcal{N}(\mathbf{y};\mu_y^m + \Sigma_{yx}^m (\Sigma_{xx}^m)^{-1}(\mathbf{x}-\mu_x^m), \Sigma_{yy}^m - \Sigma_{yx}^m (\Sigma_{xx}^m)^{-1} \Sigma_{xy}^m)}{\sum_{n=1}^{M} \alpha_n \mathcal{N}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n)} \\
&= \frac{\sum_{m=1}^{M} \alpha_n \hat{\mathcal{N}}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n) \times \mathcal{N}(\mathbf{y};\nu^m + \Gamma^m \mathbf{x}, \Omega^m)}{\sum_{n=1}^{M} \alpha_n \hat{\mathcal{N}}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n)}
\end{aligned} \tag{7.15}
$$

The $Q(\Theta^t, \Theta^{(t-1)})$ function thus evolves into Equation 7.16. Note the use of a different parametrization for the experts: ν is the conditional mean, Γ is a regressor matrix and Ω is the conditional covariance. We immediately note that the experts and gates can be separated and treated independently since they are parametrized by independent variables ($\nu^m, \Gamma^m, \Omega^m$ versus $\alpha_m, \mu_x^m, \Sigma_{xx}^m$). Both the gates and the experts can be varied freely and have no variables in common. In fact, we shall optimize these independently to maximize conditional likelihood. An iteration is performed over the experts and then an iteration over the gates. If each of those manipulations causes an increase, we will converge to a local maximum of conditional log-likelihood. This is similar in spirit to the ECM (Expectation Conditional Maximization) algorithm proposed in [42] since some variables are held constant while others are maximized and then vice-versa.

$$
\begin{aligned}
Q(\Theta^t, \Theta^{(t-1)}) &= \textstyle\sum_{i=1}^N \sum_{m=1}^M \Big(\hat{h}_{im}(\log p(\mathbf{y}_i|m, \mathbf{x}_i, \Theta^t) \\
&\quad + \log f(m, \mathbf{x}_i|\Theta^t) - \log h_{im}) - r_i f(m, \mathbf{x}_i|\Theta)) + 1 \\
&= \textstyle\sum_{i=1}^N \sum_{m=1}^M \Big(\hat{h}_{im}(\log \mathcal{N}(\mathbf{y}_i; \nu^m + \Gamma^m \mathbf{x}_i, \Omega^m) \\
&\quad + \log \alpha_m \hat{\mathcal{N}}(\mathbf{x}_i; \mu_x^m, \Sigma_{xx}^m) - \log h_{im}) - r_i \alpha_m \hat{\mathcal{N}}(\mathbf{x}_i; \mu_x^m, \Sigma_{xx}^m)\Big) + 1
\end{aligned} \tag{7.16}
$$

7.2.1 Updating the Experts

To update the experts, we hold the gates fixed and merely take derivatives with respect to the expert parameters ($\nu^m, \Gamma^m, \Omega^m$ or call them $\Phi_1^m, \Phi_2^m, \Phi_3^m$). This derivative is simplified and set equal to zero as in Equation 7.17. This is our update equation for the experts. Note how each expert has been decoupled from the other experts, from the gates and from other distracting terms. The maximization reduces to the derivative of the logarithm of a single conditioned Gaussian. This can be done analytically with some matrix differential theory and, in fact, the computation resembles a conditioned Gaussian maximum-likelihood estimate in Equation 7.18 [38]. The responsibilities for the assignments are the standard normalized joint responsibilities (as in EM). The update equation is effectively the maximum conditional likelihood solution for a conditioned multivariate Gaussian distribution.

$$
\frac{\partial Q(\Theta^t, \Theta^{(t-1)})}{\partial \Phi^m} = \textstyle\sum_{i=1}^N \hat{h}_{im} \frac{\partial \log \mathcal{N}(\mathbf{y}_i; \nu^m + \Gamma^m \mathbf{x}_i, \Omega^m)}{\partial \Phi^m} := 0 \tag{7.17}
$$

$$
\begin{aligned}
\mu^m &:= \left(\textstyle\sum_{i=1}^N \hat{h}_{im} \begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix}\right)\left(\textstyle\sum_{i=1}^N \hat{h}_{im}\right)^{-1} \\
\Sigma^m &:= \left(\textstyle\sum_{i=1}^N \hat{h}_{im}\left(\begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix} - \mu^m\right)\left(\begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix} - \mu^m\right)^T\right)\left(\textstyle\sum_{i=1}^N \hat{h}_{im}\right)^{-1} \\
\nu^m &:= \mu_y^m - \Sigma_{yx}^m (\Sigma_{xx}^m)^{-1} \mu_x^m \\
\Gamma^m &:= \Sigma_{yx}^m (\Sigma_{xx}^m)^{-1} \\
\Omega^m &:= \Sigma_{yy}^m - \Sigma_{yx}^m (\Sigma_{xx}^m)^{-1} \Sigma_{xy}^m
\end{aligned} \tag{7.18}
$$

We observe monotonic increases when the above maximization is iterated with the bounding step (i.e. estimation of $\hat{h}_i m$). This result is not too surprising since we are effectively applying a version of EM to update the experts. At this point, we turn our attention to the still unoptimized gates and mixing proportions.

7.2.2 Updating the Gates

Unlike the experts which can be bounded with EM and reduce to logarithms of Gaussians, the gates can not be as easily differentiated and set to 0. They are bounded by Jensen *and* the $\log(x) \leq x - 1$ bounding and the Gaussians of the gates do not reside nicely within a logarithm. In fact, *additional* bounding operations are sometimes necessary and thus, it is important to break down the optimization of the gates. We shall separately consider the mixing proportions (α), the means (μ_x) and the covariances (Σ_{xx}). This separation facilitates our derivation but the trade-off is that each iteration involves 4 steps: 1) optimize the experts, 2) optimize the gate mixing proportions, 3) optimize the gate means and 4) optimize the gate covariances[1]. This ECM-type [42] approach may seem cumbersome and theoretically we would like to maximize all simultaneously. However, the above separation yields a surprisingly efficient implementation and in practice the numerical computations converge efficiently.

[1] In fact, step 1) and step 2) can be performed simultaneously since they only involves taking derivatives and setting to 0.

7.2.3 Bounding Scalars: The Gate Mixing Proportions

We will update the mixing proportions by holding the experts fixed as well as the gate means and covariances. By taking derivatives of the Q function over only α_m we get Equation 7.19.

$$\begin{array}{lll} \frac{\partial Q(\Theta^t,\Theta^{(t-1)})}{\partial \alpha_m} & = & \sum_{i=1}^N \hat{h}_{im}\frac{1}{\alpha_m} - r_i\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m) := 0 \\ \alpha_m & = & \frac{\sum_{i=1}^N r_i\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m)}{\sum_{i=1}^N \hat{h}_{im}} \end{array} \tag{7.19}$$

The update for α_m is a function of the μ_x^m and the Σ_{xx}^m. It is possible to keep this equality and use it later as we derive the update rules for the means and the covariances. However, it is quite cumbersome to manipulate analytically if it is maintained as shown above. Thus, we lock the values of μ_x and Σ_{xx} at their previous estimates (i.e. at $\Theta^{(t-1)}$) and numerically update the α mixing proportions.

$$\alpha_m \quad := \quad \frac{\sum_{i=1}^N r_i\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m)|_{\Theta^{(t-1)}}}{\sum_{i=1}^N \hat{h}_{im}} \tag{7.20}$$

Since we are maximizing a bound, the above update rule increases conditional likelihood monotonically. This was also verified with a numerical implementation.

7.2.4 Bounding Vectors: The Gate Means

Upon close observation, the above technique of taking derivatives of Q and setting them to zero seems to be intractable for the case of the means. The resulting derivative contains transcendental functions which can not be easily isolated to obtain an update equation. What we can do at this point is bound the Q function specifically for the means to get a nicer expression which *can* be easily maximized. We introduce the previously discussed parabolic bounds for the Q function as in Equation 7.21. Or, equivalently, each pair of gate and data point is individually bounded by a parabola as in Equation 7.22 (with height parameter k_{im}, width parameter w_{im} and location parameter γ_{im}).

$$\begin{array}{lll} Q(\Theta^t,\Theta^{(t-1)}) & = & \sum_{i=1}^N\sum_{m=1}^M Q(\Theta^t,\Theta^{(t-1)})_{im} \\ Q(\Theta^t,\Theta^{(t-1)}) & \geq & \sum_{i=1}^N\sum_{m=1}^M k_{im} - w_{im}\|\mu_x^m - \gamma_{im}\|^2 \end{array} \tag{7.21}$$

$$\begin{array}{lll} Q(\Theta^t,\Theta^{(t-1)})_{im} & \geq & k_{im} - w_{im}\|\mu_x^m - \gamma_{im}\|^2 \\ \left\{ \begin{array}{l} \hat{h}_{im}(\log\mathcal{N}(\mathbf{y}_i;\nu^m+\Gamma^m\mathbf{x}_i,\Omega^m) \\ \quad +\log\alpha_m\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m) - \log h_{im}) \\ -r_i\alpha_m\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m) + 1/M \end{array} \right\} & \geq & k_{im} - w_{im}\|\mu_x^m - \gamma_{im}\|^2 \\ & & \Longrightarrow \text{equality at } \mu_x^{m*} \end{array} \tag{7.22}$$

Once again, the contact point for this parabola (i.e. the μ_x^{m*}) is the value for the previous model $\Theta^{(t-1)}$ or the mean of the gate before any iteration. Let us denote the previous parameters of the gate as μ_x^{m*} and Σ_{xx}^{m*}. To facilitate the derivation, we can perform what is commonly called a *whitening* operation on the coordinate system for each gate. For each gate, we can consider that its previous parameters where zero-mean and identity covariance. This is done by applying an affine transformation to the $\mathbf{x}$-space (i.e. shifting the data points $\mathbf{x}_i$ appropriately). The data is translated by the mean and scaled by the inverse of the square root of the covariance matrix. In theory, nothing changes except that each gate observes affinely displaced data instead and it then seems to have a zero mean and identity covariance as its previous operating point. This whitening operation does not need to be done explicitly (i.e. numerically) but helps simplify the following derivation for the bound.

Thus, without loss of generality, we consider that each data point has been appropriately whitened for each gate. The result is that $\mu_x^{m*} := 0$ and $\Sigma_{xx}^m = \Sigma_{xx}^{m*} = I$. Using this assumption, we can compactly solve for the parameters of the bounding parabola $(k_{im}, w_{im}, \gamma_{im})$. This is done by meeting the three requirements (contact, tangentiality and tightness). The results is shown in Equation 7.23.

$$\begin{array}{rcl} k_{im} & = & \hat{h}_{im}(\log \mathcal{N}(\mathbf{y}_i; \nu^m + \Gamma^m \mathbf{x}_i, \Omega^m) + \log \alpha_m - \frac{1}{2}{\mathbf{x}_i}^T \mathbf{x}_i - \log h_{im}) \\ & & -r_i \alpha_m e^{-\frac{1}{2}{\mathbf{x}_i}^T \mathbf{x}_i} + 1/M + w_{im}{\gamma_{im}}^T \gamma_{im} \\ \gamma_{im} & = & \frac{1}{2w_{im}}(\hat{h}_{im} - r_i \alpha_m e^{-\frac{1}{2}\mathbf{x}_i^T \mathbf{x}_i})\mathbf{x} \\ w_{im} & \geq & r_i \alpha_m \frac{e^{-\frac{1}{2}(\mathbf{x}_i - \mu_x^m)^T(\mathbf{x}_i - \mu_x^m)} - e^{-\frac{1}{2}{\mathbf{x}_i}^T \mathbf{x}_i} - e^{-\frac{1}{2}{\mathbf{x}_i}^T \mathbf{x}_i} {\mathbf{x}_i}^T \mu_x^m}{{\mu_x^m}^T \mu_x^m} + \frac{\hat{h}_{im}}{2} \end{array} \tag{7.23}$$

As shown earlier, if a globally valid bound is desired, the value of w_{im} must always be greater than the expression in Equation 7.23 no matter what the value of μ_x^m. Thus we can find the minimum w_{im} (i.e. $wmin_{im}$) for which this is true and form a parabolic bound for the Q function that is as 'tight' as possible. We shall now determine an efficient way to compute $wmin_{im}$ without searching the high dimensional space of μ_x^m each iteration.

Note in the expression for w_{im} above that μ_x^m is used only in an inner product with itself (${\mu_x^m}^T \mu_x^m$) or in an inner product with the whitened data point ${\mathbf{x}_i}^T \mu_x^m$. Thus, we can represent these two quantities regardless of the dimensionality of the vector space using at most two degrees of freedom: the magnitude of μ_x^m and the cosine of the angle between the vectors μ_x^m and $\mathbf{x}_i$. Let us now denote the new variables as follows: λ is the magnitude of μ_x^m, ρ is the magnitude of $\mathbf{x}_i$ and $\zeta = \cos(\phi)$ is the cosine of the angle between the two vectors. We then compactly rewrite the bound as in Equation 7.24.

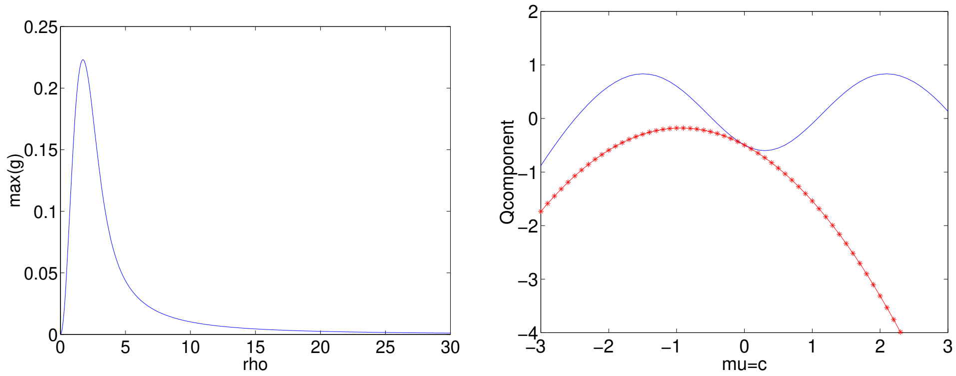
$$w_{im} \quad \geq \quad r_i \alpha_m \frac{e^{-\frac{1}{2}\rho^2}\left(e^{-\frac{1}{2}\lambda^2} e^{\zeta\lambda\rho} - \zeta\lambda\rho - 1\right)}{\lambda^2} + \frac{\hat{h}_{im}}{2} \tag{7.24}$$

The w_{im} is now lower bounded by the right hand side which is a function of two free scalars λ and ζ. We wish to find the maximum of the lower bound at some value of λ and ζ. By taking derivatives of the right hand side with respect to ζ and setting to 0, we find an extremum of the function at $\zeta = \frac{\lambda}{2\rho}$. However, we also note that $\zeta = \cos(\phi)$ and so $\zeta \epsilon [-1, 1]$. If we set $\zeta = 1$ we note that the this solution is *always* greater than the $\zeta = \frac{\lambda}{2\rho}$. So, the extremum we solved for was a *minimum*. Thus, the maximum occurs at the ends of the interval so $\zeta = \pm 1$. Therefore, we have the relationship $\mu_x^m \propto \mathbf{x}_i$. In other words, let us define $\mu_x^m = c\mathbf{x}_i$. The expression for w_{im} then simplifies into a function of 1 free variable (c) as in Equation 7.25.

$$\begin{array}{rcl} w_{im} & \geq & r_i \alpha_m e^{-\frac{1}{2}\rho^2} \frac{e^{-\frac{1}{2}c^2} e^{c\rho} - c\rho - 1}{c^2} + \frac{\hat{h}_{im}}{2} \\ wmin_{im} & = & r_i \alpha_m \max\{e^{-\frac{1}{2}\rho^2} \frac{e^{-\frac{1}{2}c^2} e^{c\rho} - c\rho - 1}{c^2}\} + \frac{\hat{h}_{im}}{2} \\ wmin_{im} & = & r_i \alpha_m e^{-\frac{1}{2}\rho^2} \max\{g(c, \rho)\} + \frac{\hat{h}_{im}}{2} \end{array} \tag{7.25}$$

For each value of ρ we find the value of c which maximizes the value of $g(c, \rho)$. This is done offline (taking a few seconds) using Brent's method and the result is stored in a 1D lookup table (LUT). The values are shown in Figure 7.3(a) where we plot the max of the g function with respect to ρ. Using this table, we can immediately compute the minimum $wmin_{im}$ by looking up the corresponding value of ρ for the current data point and subsequently compute the rest of the parabolic bound's parameters.

An example of a parabolic bound is shown in Figure 7.3(b) under a sub-component of the Q function. This sub-component corresponds to a single data point as we vary one gate's uni-variate mean. Many such parabolic bounds are computed (one for each data point and gate combination) and are summed to bound the Q function in its entirety.



(a) Parabolic Width LUT (b) 1D Mean Bounding Example

Figure 7.3: Look-Up Table and Example Mean Parabolic Bound

- Parabolic Piece-Wise Bound

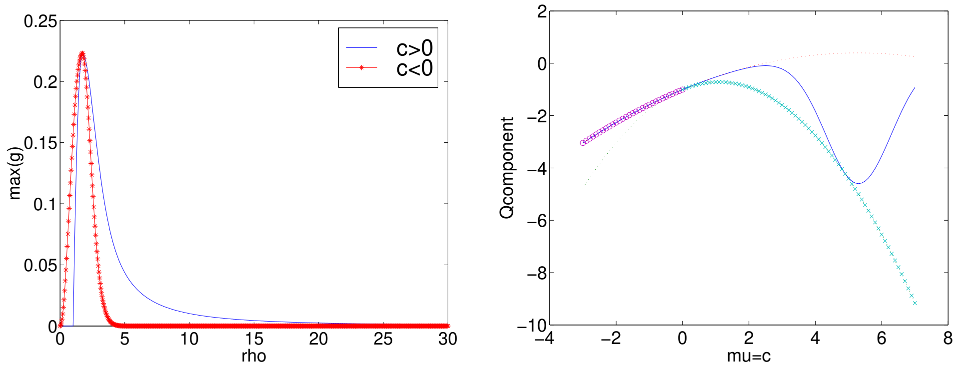
  For greater computational efficiency we compute the gradient of the Q on the mean to get an idea of the *direction* of steepest ascent. We can then know which general direction μ_x^m will move towards for the next iteration. Due to whitening, μ_x^m starts off at the origin and moves from there. Thus, we know if μ_x^m is moving close towards $\mathbf{x}_i$ or farther away from it. Equivalently, we know if c will be positive or negative. The above LUT (Figure 7.3(a)) returns the $wmin_{im}$ which is valid for all possible choices of c. However, if we know the direction c is going to move along, we only need to consider half of the possibilities in using Brent's method to find the max of $g(c, \rho)$. Thus, we obtain two LUTs for c positive and for c negative as shown in Figure 7.4(a). This typically gives us an even tighter bound since we only need to bound the Q function in regions of the landscape where the locus of the maximum could move to. Thus, we are effectively approximating the landscape with two parabolas, which are cut at $c = 0$. One is a bound for the values at $c < 0$ and one for $c > 0$. Figure 7.4(b) depicts this piece wise parabolic bound on a sub-component of Q. The function being bounded is represented with a continuous line. One parabola is composed of 'x' symbols and is a valid bound to the right of zero. The other parabola is composed of 'o' symbols and is valid to the left of zero. The piece-wise bound is composed of these two parabolas spliced at the origin (their continuations are shown as dotted lines for visualization). Thus, if the gradient tells us that we will be moving in a direction of increasing μ, we should compute the parabola made of 'x' symbols. However, if we are moving to the left (because of the sum total influence of the other bounds and functions), we should use the parabola made of 'o' symbols which is a wider and tighter bound.

To obtain a final answer for the update of the gate means μ_x^m we simply maximize the sum of parabolas which bounds the Q in its entirety. For M gates and N data points, we obtain $M \times N$ parabolas each with its own $(wmin_{im}, k_{im}, \gamma_{im})$. The update is in Equation 7.26.

$$\mu_x^m \quad := \quad \frac{\sum_{i=1}^N wmin_{im}\gamma_{im}}{\sum_{i=1}^N wmin_{im}} \tag{7.26}$$

7.2.5 Bounding Matrices: The Gate Covariances

Having shown the update equation for moving each gate's mean, we now turn our attention to the gate's covariance. Recall that each gating function was an unnormalized Gaussian as in Equation 7.14. Gaussians have no constraints on their mean which can be any real vector. Thus, it was possible to just



(a) Piece-Wise Parabolic Width LUT (b) 1D Mean Bounding Example

Figure 7.4: Look-Up Table and Example Mean Piece-Wise Parabolic Bound

bound the Q function with parabolas for the mean variable. However, Gaussians have constraints on their covariances which must remain symmetric and positive semi-definite. Thus, we can not use simple parabolic bounds trivially over all the parameters of the Σ_{xx}^m matrices. Maximizing a parabola might return parameters for Σ_{xx}^m which don't maintain the above constraints. Thus, we explicitly bound the Q function with the logarithm of a single Gaussian density. Maximizing a Gaussian (as in maximum likelihood type calculations) is guaranteed to yield a symmetric positive definite matrix. Thus, we shall use it as a bound as in Equation 7.27. For compactness, we are merely writing Σ_{xx}^m as Σ.

$$
\begin{array}{ccc}
Q(\Theta^t,\Theta^{(t-1)})_{im} & \geq & \log(\text{Gaussian}) \\
\left\{ \begin{array}{l} \hat{h}_{im}(\log \mathcal{N}(\mathbf{y}_i;\nu^m+\Gamma^m\mathbf{x}_i,\Omega^m) \\ \quad +\log\alpha_m\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m) - \log h_{im}) \\ \quad -r_i\alpha_m\hat{\mathcal{N}}(\mathbf{x}_i;\mu_x^m,\Sigma_{xx}^m) + 1/M \end{array} \right\} & \geq & k_{im} - w_{im}\gamma_{im}^T {\Sigma_{xx}^m}^{-1}\gamma_{im} + w_{im}\log|{\Sigma_{xx}^m}^{-1}| \\
 & & \Longrightarrow \text{equality at } \Sigma_{xx}^{m*}
\end{array}
\tag{7.27}
$$

Once again, we assume the data has been appropriately whitened with respect to the gate's previous parameters. Without generality, therefore, we can transform the data and assume that the previous value of the mean is 0 and the covariance is identity. We again solve for the parameters of the log-Gaussian bound using the three requirements (contact, tangentiality and tightness) and obtain the k_{im}, h_{im}, w_{im} as in Equation 7.28.

$$
\begin{array}{lcl}
k_{im} & = & \hat{h}_{im}(\log\mathcal{N}(\mathbf{y}_i;\nu^m+\Gamma^m\mathbf{x}_i,\Omega^m)+\log\alpha_m-\frac{1}{2}{\mathbf{x}_i}^T\mathbf{x}_i-\log h_{im}) \\
 & & \quad -r_i\alpha_m e^{-\frac{1}{2}{\mathbf{x}_i}^T\mathbf{x}_i}+1/M+w_{im}{\gamma_{im}}^T\gamma_{im} \\
\gamma_{im}\gamma_{im}^T & = & \frac{1}{2w_{im}}\left(\hat{h}_{im}-r_i\alpha_m e^{-\frac{1}{2}{\mathbf{x}_i}^T\mathbf{x}_i}\right)\mathbf{x}_i{\mathbf{x}_i}^T+I \\
w_{im} & \geq & r_i\alpha_m\frac{\frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T\mathbf{x}_i-\frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T\Sigma^{-1}\mathbf{x}_i+\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)-\exp(-\frac{1}{2}\mathbf{x}_i^T\Sigma^{-1}\mathbf{x}_i)}{D-tr(\Sigma^{-1})+\log|\Sigma^{-1}|}
\end{array}
\tag{7.28}
$$

As before, we would like an efficient way of solving for the minimal value of w_{im} to get the tightest possible bound. In the definition, w_{im} is lower bounded by a function of the covariance matrix which may contain very many parameters. We wish to find the maximum of this function (call it g) to find the minimum valid w_{im} (which forms the tightest bound). Instead of exhaustively searching for the maximum of g by varying the many parameters of Σ, we note a few simplifications. First, the covariance matrix can

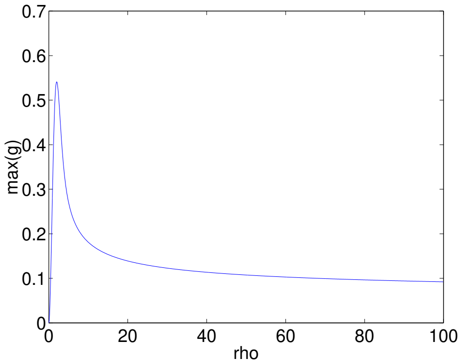


Figure 7.5: Look-Up Table for Covariance Bound

be written in terms of eigenvalues and eigenvectors $\Sigma^{-1} = V^T \Lambda V$. We also note that the denominator does not rely on the eigenvectors and can be written as in Equation 7.29.

$$
\begin{aligned}
g &= \frac{\frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T\mathbf{x}_i - \frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T\Sigma^{-1}\mathbf{x}_i + \exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i) - \exp(-\frac{1}{2}\mathbf{x}_i^T\Sigma^{-1}\mathbf{x}_i)}{D - tr(\Sigma^{-1}) + \log|\Sigma^{-1}|} \\
&= \frac{\frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T\mathbf{x}_i - \frac{1}{2}\exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i)\mathbf{x}_i^T V^T\Lambda V\mathbf{x}_i + \exp(-\frac{1}{2}\mathbf{x}_i^T\mathbf{x}_i) - \exp(-\frac{1}{2}\mathbf{x}_i^T V^T\Lambda V\mathbf{x}_i)}{D - tr(\Lambda) + \log|\Lambda|}
\end{aligned}
\tag{7.29}
$$

The numerator of g can thus use V to further optimize g independently of the denominator. As a consequence, one can vary the eigenvectors to *select* any eigenvalue in Λ by rotating the basis. Thus, the numerator only depend on the minimal and maximal values of the eigenvalue matrix (λ_{min} and λ_{max}). The denominator can then freely vary all other eigenvalues in between without affecting the numerator's range. Thus, we only need to search over different values of λ_{min} and λ_{max} or two parameters. Using this and a further constraint which forces either λ_{min} or λ_{max} to be 1, we end up with a g function which is only a function of the magnitude of the data point $\rho = \mathbf{x}_i{}^T\mathbf{x}_i$ and a single free scalar parameter λ as in Equation 7.30. This result was verified with numerical tests as well.

$$
\begin{aligned}
w_{im} &\geq r_i \alpha_m g(\rho, \lambda) \\
wmin_{im} &= r_i \alpha_m \max\{g(\rho, \lambda)
\end{aligned}
\tag{7.30}
$$

Given a value of ρ from observed data, we need to find the best $wmin_{im}$ by checking all the values of λ. In an offline process which requires a few seconds, we form a table of $wmin_{im}$ for each ρ using Brent's method at each point. Thus, using this 1D lookup table, we can immediately compute the width of the bound given a data point. The values are shown in Figure 7.5 where we plot the max of the g function with respect to ρ. Using this table, we can immediately compute the minimum $wmin_{im}$ by looking up the corresponding value of ρ for the current data point and subsequently compute the rest of the bound's parameters.

Some examples of a log-Gaussian bound are shown in Figure 7.6 under different sub-components of the Q function. Each sub-component corresponds to a single data point as we vary one gate's uni-variate covariance. Many such parabolic bounds are computed (one for each data point and gate combination) and are summed to bound the Q function in its entirety.

To obtain a final answer for the update of the gate covariances Σ_{xx}^m we simply maximize the sum of log Gaussians which bounds the Q in its entirety. For M gates and N data points, we obtain $M \times N$ bounds each with its own $(wmin_{im}, k_{im}, \gamma_{im})$. The update is in Equation 7.31.

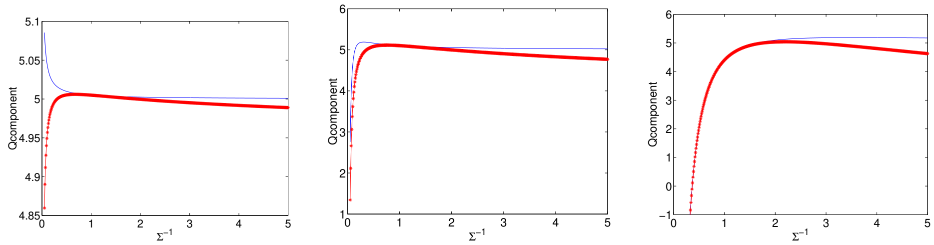


Figure 7.6: Covariance Bounding Examples

$$\Sigma_{xx}^m \quad := \quad \frac{\sum_{i=1}^N wmin_{im} \gamma_{im} \gamma_{im}{}^T}{\sum_{i=1}^N wmin_{im}} \tag{7.31}$$

This covariance is then unwhitened by inverting the transform used to move the data. This undoes the whitening operation which allowed us to transform data and consequently simplify intermediate calculations .

7.2.6 MAP Estimation

We have demonstrated a conditional maximum likelihood algorithm for a conditioned mixture of Gaussians. Essentially, this optimized the likelihood of the data given a model $p(\mathcal{Y}|\mathcal{X}, \Theta)$. If a prior on the model $p(\Theta)$ is given, we can perform MAP estimation simply by computing a bound on the prior as well. Recall that we had bounded the conditional log-likelihood (as in Equation 7.6) using a Q function. We depict the inclusion of a prior in Equation 7.32. Note how this does not change the derivation of the maximum conditional likelihood solution and merely adds one more bound for the prior to the N bounds computed for each data point for the ML^c solution. In fact, the optimization of the prior is identical as it would be for a joint density maximum a posteriori problem (MAP) and the fact that we are optimizing MAP^c does not affect it. Thus, a prior on a conditioned mixture model being update with CEM will can be bounded exactly as a prior on a normal mixture model being updated with EM. In theory, it is thus possible to add to the above CEM derivation a variety of priors including non-informative priors, conjugate priors, entropic priors and so on.

$$\begin{aligned} \Delta l^c &= \log p(\mathcal{Y}|\mathcal{X}, \Theta^t) p(\Theta^t) - \log p(\mathcal{Y}|\mathcal{X}, \Theta^{(t-1)} p(\Theta^{(t-1)})) \\ &= \sum_{i=1}^N \left(\log \frac{\sum_{m=1}^M p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^t)}{\sum_{m=1}^M p(m, \mathbf{x}_i, \mathbf{y}_i|\Theta^{(t-1)})} - \log \frac{\sum_{n=1}^M p(n, \mathbf{x}_i|\Theta^t)}{\sum_{n=1}^M p(n, \mathbf{x}_i|\Theta^{(t-1)})} \right) + N \times \log \frac{p(\Theta^t)}{p(\Theta^{(t-1)})} \\ &\geq Q(\Theta^t, \Theta^{(t-1)}) + Q_{MAP}(\Theta^t, \Theta^{(t-1)}) \end{aligned} \tag{7.32}$$

7.3 Implementation and Interpretation

We now describe the implementation of the CEM algorithm as an iterative process. This typical time per iteration is comparable with a regular mixture of Gaussians model being update with EM. This is partly due to the decomposition of the variables into $\mathbf{x}$ and $\mathbf{y}$. EM uses a large covariance matrix of both variables concatenated together and the matrix inversion in the EM-loop is thus of a higher dimensionality than the CEM matrix inversions of Σ_{xx} and Σ_{yy}. In high dimensions, therefore, when matrix inversions become a serious computational costs, the CEM algorithm can outperform EM in the per-iteration time. Some results will be shown in 2D problems that can be visually verified. Other higher dimensionality tests were also conducted and are reported as well.

Start	CE Step	Compute the Conditional Expectation (the h_{im} and r_i parameters with respect to $\Theta^{(t-1)}$
	M Step	Update the Experts and the Mixing Proportions α_m $\Theta^{(t-1)}$
	CE Step	Update the h_{im} and the r_i due to the Change in the Experts and Mixing Proportions
	M Step	Update the Gate Means
	CE Step	Update the h_{im} and the r_i due to the Change in the Gate Means
	M Step	Update the Gate Covariances
Repeat		Replace Old $\Theta^{(t-1)}$ with the New Θ and Return to Start

Table 7.1: CEM Iteration Loop

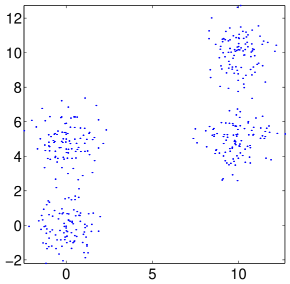
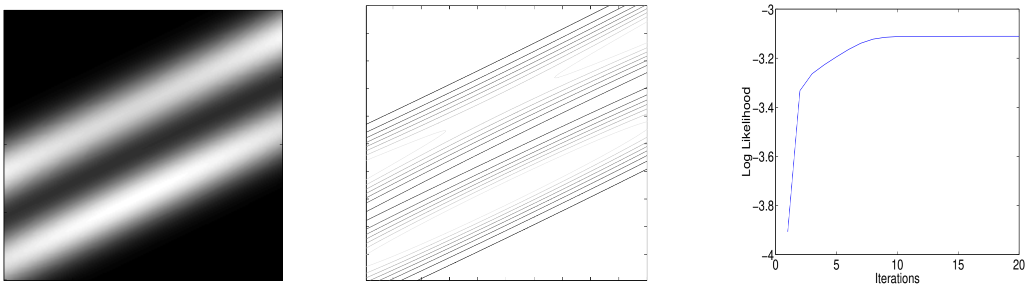


Figure 7.7: 4 Gaussian Data to Model with 2 Gaussians

We assume that the conditional model is initialized using a pseudo-random method which places the initial Gaussians with some coverage of the data. From there the algorithm proceeds in a continuous loop as described by Table 7.1.

Figure 7.7 depicts a set of 2D data points with coordinates (x, y). This data has been sampled from 4 Gaussians with circular covariance. The objective is to fit a conditional model $p(y|x)$ to the above data with a conditioned mixture of Gaussians model with *only 2 Gaussians*. Thus, the algorithm can not simply group 4 clusters but must take advantage of some symmetric structure in the problem to find a more compact probabilistic model. This is thus a clear problem of limited resources where we would like to maximize performance at guessing output y from input x and we do not have the luxury of modeling the whole generative process (i.e. with 4 Gaussians). Thus, information about the particular task is critical here.

In Figure 7.8 we note the maximum conditional likelihood solution which places the Gaussians in a



(a) Conditioned 2 Gaussians (b) Iso-Probability Contours (c) Conditional Log-Likelihood

Figure 7.8: Global and Typical ML^c Estimate using CEM

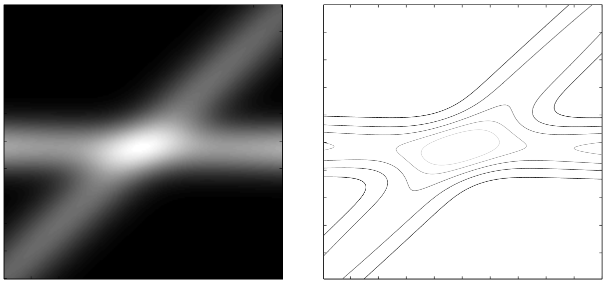
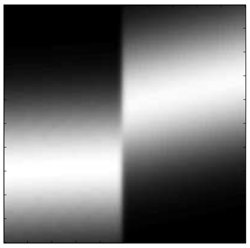
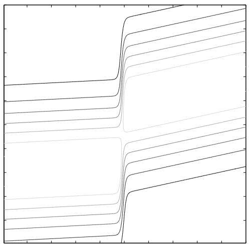


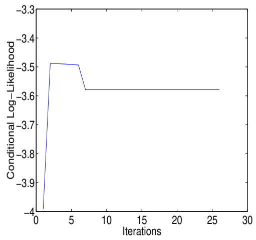
Figure 7.9: Inferior Local ML^c Estimate using CEM



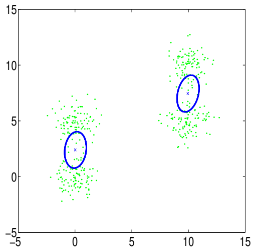
(a) Conditional



(b) Iso-Probability



(c) Conditional Log-Likelihood



(d) Clustering

Figure 7.10: ML Estimate using EM

skewed position to regress the y data given an x value. Note how the conditioned Gaussians give no information about x since the CEM expects it to be given. In Figure 7.9 we see another solution where the CEM was trapped in a slightly inferior local maximum (Conditional Log Likelihood = -3.166).

Figure 7.10 shows the evolution of the EM algorithm on the exact same data set. We again plot the conditional log-likelihood and see that it is *not* being maximized (as expected) while the joint log-likelihood is being maximized. However, given that this is a conditional problem between x-input and y-output, the algorithm is not optimizing for the correct task and is not being discriminative. Thus the Gaussians are wasted modeling x and do not recover the bimodal distribution in y. The conditional log-likelihood is also dramatically inferior (-3.6).

The CEM algorithm, when initialized with a solution of the EM algorithm, typically moves to another solution. This solution is often better than the EM's solution but worse than initialization on its own. This indicates that the local maxima of joint likelihood are not necessarily maxima of conditional likelihood.

7.3.1 Conditional Constraints vs. Joint Constraints

We now briefly discuss the difference in constraints on a conditional model versus a joint model. First and foremost, the conditional model (in theory) provides no information about the density in the covariate variables ($\mathbf{x}$). Thus, it does not allocate any resources in modeling the $\mathbf{x}$ domain unless it indirectly helps model the $\mathbf{y}$ domain. Thus, the M Gaussians (i.e. the model's finite resources) do not cluster around the density in $\mathbf{x}$ unnecessarily.

In addition, note that there are no constraints on the gates to force them to integrate to 1. The mixing proportions are not necessarily normalized and the individual gate models are unnormalized Gaussians. Thus, the gates form an unnormalized marginal density called $f(\mathbf{x})$ which need not integrate to 1. In joint models, on the other hand, the marginal $p(\mathbf{x})$ must integrate to 1.

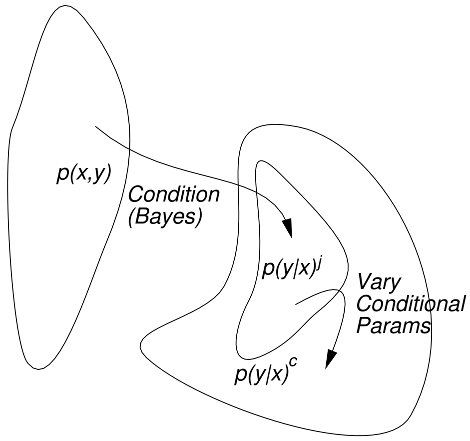


Figure 7.11: Constraints on Joint Density

Finally, we note that the covariance matrix in the gates is independent of the covariance matrix and regressor matrix in the expert. In a full joint Gaussian, these 3 matrices combine into one large matrix and this matrix as a whole must be symmetric and positive semi-definite. However, here, the gate covariance need not be positive semi-definite. The expert covariance is symmetric positive semi-definite *only on its own* and the regressor matrix is arbitrary. Thus, the constraints on the total parameters are fewer than in the joint case and each gate-expert combination can model a larger space than a conditioned Gaussian. Thus, training up a conditional model directly will yield solutions that lie outside the space of the conditioned joint models. This is depicted in Figure 7.11. Note how the additional constraints on the joint density limit the realizable conditional models. This limit is not present if the conditional models can be varied in their final parametric form.

7.4 Applying the Model

Assume we have a fully optimized conditional model of the form $p(\mathbf{y}|\mathbf{x})$ from many examples of data $(\mathbf{x}_i, \mathbf{y}_i)$ through the use of a an algorithm such as CEM. Our typical scenario is to use this model and its estimated parameters to predict an output response given an input observation.

During runtime, we need to quickly generate an output $\mathbf{y}$ given the input $\mathbf{x}$. Observing an $\bar{\mathbf{x}}$ value turns our conditional model $p(\mathbf{y}|\mathbf{x})$ into effectively a marginal density over $\mathbf{y}$ (i.e. $p(\mathbf{y})$). The observed $\bar{\mathbf{x}}$ makes the gates act merely as constants, G_m, instead of as Gaussian functions. In addition, the conditional Gaussians which were original experts become ordinary Gaussians when we observe $\mathbf{x}$ and the regressor term $nu^m + \Gamma^m \mathbf{x}$ becomes a simple mean μ^m. If we had a conditioned mixture of M Gaussians, the marginal density that results is an ordinary sum of M Gaussians in the space of $\mathbf{y}$ as in Equation 7.33.

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x},\Theta) &= \frac{\sum_{m=1}^{M} \alpha_n \hat{\mathcal{N}}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n) \times \mathcal{N}(\mathbf{y};\nu^m+\Gamma^m\mathbf{x},\Omega^m)}{\sum_{n=1}^{M} \alpha_n \hat{\mathcal{N}}(\mathbf{x};\mu_x^n,\Sigma_{xx}^n)} \\
p(\mathbf{y}|\bar{\mathbf{x}},\Theta) &= \frac{\sum_{m=1}^{M} G_m \times \mathcal{N}(\mathbf{y};\mu^m,\Omega^m)}{\sum_{n=1}^{M} G_n}
\end{aligned}
\tag{7.33}
$$

Observe the 1D distribution in Figure 7.12. At this point, we would like to choose a single candidate $\hat{\mathbf{y}}$ from this distribution. There are many possible strategies for performing this selection with varying efficiencies and advantages. We consider and compare the following three approaches. One may select

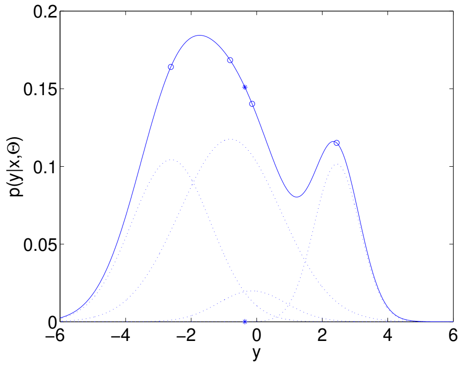


Figure 7.12: Output Probability Distribution, Expectation and Prototypes

a random sample from $p(\mathbf{y})$, one may select the average $\mathbf{y}$ or one may compute the $\mathbf{y}$ with the highest probability.

Sampling will often return a value which has a high probability however, it may sometimes return low values due to its inherent randomness. The average, i.e. the expectation, is a more consistent estimate but if the density is multimodal with more than one significant peak, the $\mathbf{y}$ value returned might actually have low $p(\mathbf{y})$ [5] [2] (as is the case in Figure 7.12). Thus, if we consistently wish to have a response $\hat{\mathbf{y}}$ with high probability, the best candidate is the highest peak in the marginal density, i.e. the arg max.

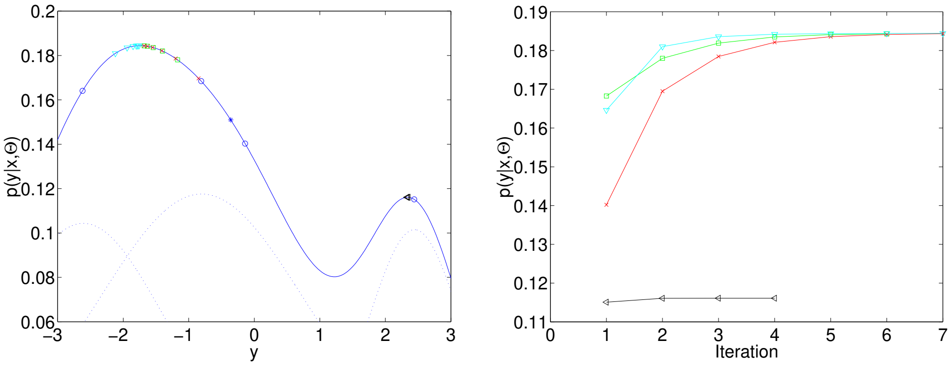
7.4.1 Expectation and Arg Max

We shall now describe a technique for computing the argmax for a mixture model using the bounding techniques introduced earlier. Traditionally, the mean of each Gaussian is evaluated to determine its probability and the best mean is returned as the arg max. This is not always the correct solution and we derive a technique with which the true arg max can be computed.

Observe the Gaussian mixture model shown in Figure 7.12 (what we get *after* training the model and observing a new covariate input $\bar{\mathbf{x}}$. It is composed of 4 Gaussians which are shown alone with dotted lines. We would like to solve for the $\mathbf{y}$ which maximizes the expression. One may be tempted to just try setting $\mathbf{y}$ to the prototype of each Gaussian μ^m and checking to see which one of these is the best. The 4 prototypes points are shown as circles on the pdf and none of them is the maximum. This is due to interaction between the individual models and the maximum could, for instance, lie between two Gaussians. Therefore, this is usually not the arg max. In addition, we show the expectation value on the curve (the average $\mathbf{y}$) by a * symbol and also note that it is not maximal.

We will consider the prototypes as seeds and try optimizing the probability from there. Thus, we should converge to the *global* maximum of probability after investigating all possible attractors. For M Gaussians, there are M attractor basins and these usually have non-zero mutual interaction. Thus, assume we are at a current operating point $\mathbf{y}^{(t-1)}$ (seeded at one of the Gaussian means or prototypes). We wish to find a better $\mathbf{y}^t$ using an iterative approach. Equivalently, we can maximize the logarithm of the difference in probabilities Δlp as in Equation 7.34.

[2] A classic example of this is that of dystal learning to recover the angle at which to throw a ball when trying shoot it a pre-specified distance. Since throwing a ball at 90 or 0 degrees is equivalent (and any other (90-x) or (x) combination), the expected value is to always throw the ball at 45 degrees which is incorrect and would have probability ≈ 0.



(a) (b)

Figure 7.13: Arg Maximization

$$
\begin{aligned}
\Delta lp &= \log p(\mathbf{y}^t|\bar{\mathbf{x}},\Theta) - \log p(\mathbf{y}^{(t-1)}|\bar{\mathbf{x}},\Theta) \\
&= \log \frac{\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^t;\mu^m,\Omega^m)}{\sum_{n=1}^M G_n} - \log \frac{\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)}{\sum_{n=1}^M G_n} \\
&= \log \frac{\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^t;\mu^m,\Omega^m)}{\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)} \\
&\leq \sum_{m=1}^M \frac{G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)}{\sum_{n=1}^M G_n \mathcal{N}(\mathbf{y}^{(t-1)};\mu^n,\Omega^n)} \log \frac{\mathcal{N}(\mathbf{y}^t;\mu^m,\Omega^m)}{G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)}
\end{aligned}
\tag{7.34}
$$

Applying Jensen's inequality yields a bound which is very similar to our previous Q-function in the EM algorithm. Effectively, we propose solving arg maximization using an EM framework with multiple seeds (a total of M seeds for M Gaussians). In fact, the application of Jensen's inequality above is the E-step. The M-step is obtained by taking derivatives with respect to $\mathbf{y}^t$ of the bound and setting them to 0. We obtain the update equation for the $\mathbf{y}$ as in Equation 7.35. This E and the M steps are iterated and $\mathbf{y}$ monotonically converges to a local maximum. For each of the 4 seeds, we show the convergence on the mixture model in Figure 7.13(a) and the monotonic increase in Figure 7.13(b). Given that only a handful of iterations are needed and only M Gaussians are being used, the complete arg maximization process we implemented requires only several milliseconds to find the best $\mathbf{y}$ occupying a vector space of dimensionality greater than 100.

$$
\mathbf{y}^t := \left(\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)\Omega^{m(-1)}\mu^m\right)\left(\sum_{m=1}^M G_m \mathcal{N}(\mathbf{y}^{(t-1)};\mu^m,\Omega^m)\Omega^{m(-1)}\right)^{-1} \tag{7.35}
$$

If time is a critical factor, we can use standard techniques to compute a mere expectation of the mixture model which is given in Equation 7.36. This solution, often has lower probability than the arg max.

$$
\hat{\mathbf{y}} := \frac{\sum_{m=1}^M G_m \mathcal{N}(\mu^m;\mu^m,\Omega^m)\mu^m}{\sum_{m=1}^M G_m \mathcal{N}(\mu^m;\mu^m,\Omega^m)} \tag{7.36}
$$

7.4.2 Standardized Database Performance

To test the CEM algorithm in a typical application, we utilized a standardized regression database from the UCI Machine Learning Repository (Information and Computer Science Department) at the

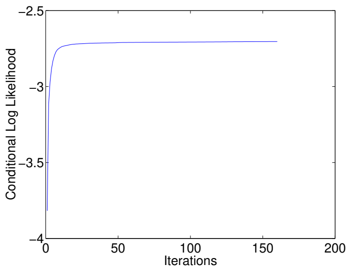


Figure 7.14: Conditional Log-Likelihood on Training Data

Algorithm	3 - Class Accuracy	Regression Accuracy	Training Time
Cascade-Correlation (no hidden nodes)	61.40 %	24.86 %	
Cascade-Correlation (5 hidden nodes)	65.61 %	26.25 %	
C4.5	59.2 %	21.5 %	
Linear Discriminant	32.57%	0.0 %	
k=5 Nearest Neighbour	62.46 %	3.57 %	
Backprop	64 %		
Dystal	55 %		
CEM 1 Gaussian	60.06 %	20.79 %	$<$ 1 minute
CEM 2 Gaussians	62.26 %	26.63 %	$<$ 1 minute
CEM 100 Gaussians	64.46 %	27.39 %	$<$ 20 minutes

Table 7.2: Testing Results

University of California, Irvine. The data set used was the Abalone Data courtesy of the Marine Research Laboratories - Taroona.

The required task is to predict the age of abalone from physical measurements. Scientists determine the age of abalone by a pain-staking process which requires cutting through the shell and the cone, staining and counting the number of rings with a microscope. For each abalone, the dataset contains the age and 8 easily measured physical attributes. These are: sex, length, diameter, height, whole height, shucked weight, viscera weight and shell weight. The database is claimed to be not linearly separable and data set samples are highly overlapped.

The data set is split into 3133 training examples and 1044 test examples. We compare the results of the CEM algorithm (Mixture of Gaussians, annealed at 0.4) when run exclusively on the training data and then tested *once* on the test data. Figure 7.14 displays the conditional log-likelihood for the 100 Gaussian training. Results for age prediction (regression) are shown in Table 7.2. Also, the age was also split arbitrarily into 3 classes to simulate a classification task: Young, Middle-Aged and Old. The CEM algorithm was not retrained for this task and only tested on the classification performance using its numerically regressed estimates of age. These results are shown in comparison with other methods [69] [13] as well in the table. Approximate training times for the CEM algorithm examples are shown as well and compare favorably with other techniques. Each of the 3 CEM algorithm examples was run only once on the test data. In addition, no cross-validation maximization was made or complexity optimization (under fitting and over fitting) so the results for the CEM algorithm would probably be significantly better with such standard techniques. Despite this, the CEM algorithm (for the complex 100 Gaussian case) outperformed all methods on the regression task (the task it was trained for) and fared favorably on the classification task.

Chapter 8

An Integrated Learning and Synthesis System

At this point, a review of the components introduced so far is in order as well as a discussion of the integrated system. The flow between perceptual input, the graphical output, the time series processing and the learning system are presented in detail and different modes of operation are enumerated. The types of operation can be conceptually split into three categories: human-human interaction, human-machine interaction and machine-machine interaction. We specifically describe the ARL framework as it is used for behaviour learning, interaction, prediction, filtering and simulation. In addition, we point out some practical issues in the integrated system such as the dimensionality of the data in the flow and the temporal constraints of each module.

8.1 System Components and Integration

Recall the formalism that was established in the previous chapters. In the discussion of time series processing, a vector notation was employed describing past short term memory as $\mathbf{x}(t)$ and measurements (future) of both users as $\mathbf{y}(t)$. We re-use this notation and assume that two users (user A and user B) can be connected to the ARL system. The two corresponding vision systems independently recover $\mathbf{y}_A(t)$ and $\mathbf{y}_B(t)$ which are fed through the learning system to two graphics systems in real-time. We enumerate the functionality of each component in the system in terms of this notation.

- Perception - Vision

  Generates for user a vector of instantaneous perceptual measurements. The vision system on user A generates $\mathbf{y}_A(t)$ and the vision on user B generates $\mathbf{y}_B(t)$.

- Output - Graphics

  Synthesizes graphically a vector of perceptual measurements, for example either $\mathbf{y}_A(t)$ or $\mathbf{y}_B(t)$.

- Time-Series Memory

  Accumulates both the actions of user A and user B by concatenating $\mathbf{y}_A(t)$ and $\mathbf{y}_B(t)$ together into $\mathbf{y}(t)$. Also the module stores many of these vectors, $\{\mathbf{y}(t-1), \mathbf{y}(t-1), ..., \mathbf{y}(t-T)\}$ into a short term memory. The unit then pre-process the short term memory with decay and dimensionality reduction to form a compact vector $\mathbf{x}(t)$.

- Learning

Learns from the past short term memory $\mathbf{x}(t)$ and the immediate subsequent vector of measurements from both users $\mathbf{y}(t)$ (generated by user A and B). Using the CEM machinery, many such pairs of $\{\mathbf{x}(t), \mathbf{y}(t)\}$ from a few minutes of interaction form a probability density $p(\mathbf{y}|\mathbf{x})$. We can use this model to compute a predicted $\hat{\mathbf{y}}(t)$, the immediate future, for any observed $\mathbf{x}(t)$ short term past.

In summary, the vision systems (one per user) both produce the components $\mathbf{y}_A(t)$ and $\mathbf{y}_B(t)$ which are concatenated into $\mathbf{y}(t)$. This forms a stream which is fed into some temporal representation. Recall that an accumulation of the past $\mathbf{y}(t-1), ..., \mathbf{y}(t-T)$ was denoted $Y(t)$. It was then processed with decay and dimensionality reduction to generate $\mathbf{x}(t)$. Then, a learning system is used to learn the conditional density $p(\mathbf{y}(t)|\mathbf{x}(t))$ from many example pairs of $(\mathbf{x}(t), \mathbf{y}(t))$. This allows it to later compute an estimate $\hat{\mathbf{y}}(t)$ given the current $\mathbf{x}(t)$. Finally, for synthesis, the estimate is broken down into $\hat{\mathbf{y}}_A(t)$ and $\hat{\mathbf{y}}_B(t)$ which are predictions for each user. However, it is critical to note that the estimate into the future $\hat{\mathbf{y}}(t)$ is an instantaneous estimate and, on its own, does not generate any finite length, meaningful action or gesture.

8.2 Modularity

The modularity of the architecture allows flexibility and permits the use of different units and different paths of data. We will investigate the possible modes of operation that can result. These include behaviour learning, interaction, prediction, filtering and simulation. In addition, the perceptual unit and graphical unit can also be swapped for different sensors increasing the flexiblity of the system.

8.3 Human and Human

We now discuss the case when two humans are present and are interacting with each other while the system 'spies' on the interaction. This mode is essential for the imitation type of interaction learning in ARL.

8.3.1 Training Mode

For training, two humans interact with each other in a somewhat natural way while the system accumulates information about the actions and reactions. This is depicted in detail in Figure 8.1.

Here, the operation is straightforward. The two users are interacting and the learning system is being fed $\mathbf{x}(t)$ on one end and $\mathbf{y}(t)$ on the other. Then, once many pairs of data are accumulated, the system uses CEM to optimize a conditioned Gaussian mixture model which represents $p(\mathbf{y}|\mathbf{x})$. Once again, we note the role of the integration symbol which indicates the pre-processing of the past time-series via an attentional window over the past T samples of measurements. This window can be represented compactly in an eigenspace with $\mathbf{x}(t)$.

Typically the two users interact for a few minutes indicating to the system some specific patterns of behaviour via a distribution of $(\mathbf{x}, \mathbf{y})$ pairs. Once the CEM converges to a maximum conditional likelihood solution that approximates this distribution, the interaction of the two users has been learned and can be used to generate predictions.

8.3.2 Prediction Mode

In prediction mode, the system has already learned with CEM and can therefore use the incoming data to forecast a small step into the future. This situation is depicted in Figure 8.2. Here, the output motion of user B is actually being generated by the learning system instead of directly piping out of his vision system. It becomes quickly apparent that nothing too interesting can happen as a result of this mode of operation. The ARL system is simply operating as a filter (i.e. a Kalman filter) since it is only

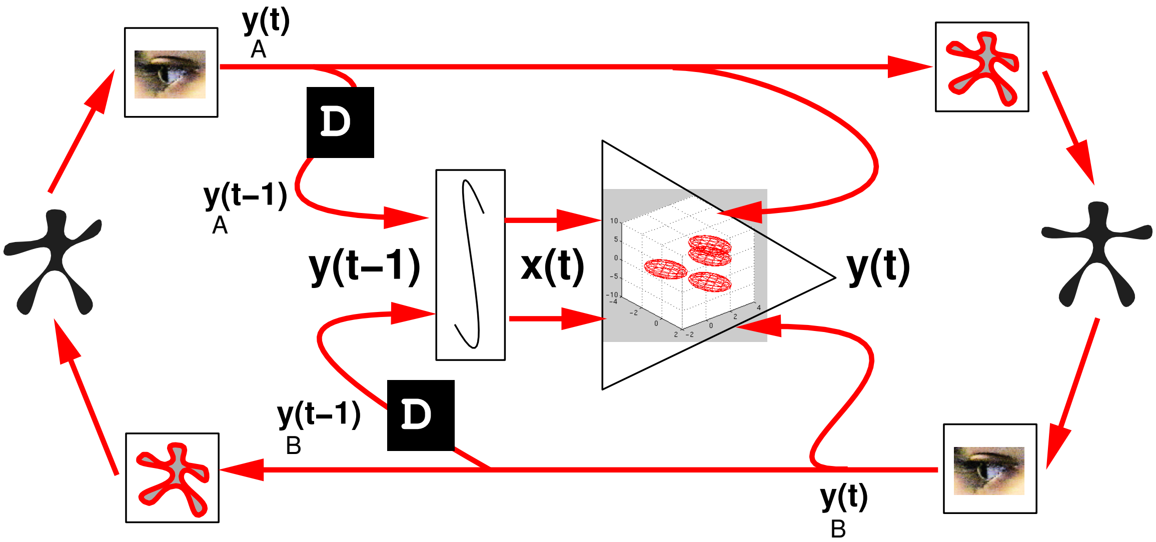


Figure 8.1: Training Mode

generating some slightly modified version, $\hat{\mathbf{y}}_B(t)$ of the original signal $\mathbf{y}_B(t)$. When user B exits the scene, the system merely locks up since the $\mathbf{x}(t)$ is only being fed half of the signal (from user A). The $\mathbf{x}$ vector only contains memory about user A and the system has no memory of its own actions. Thus, only a portion of $\mathbf{x}$ is being updated and poor estimates using the pdf $p(\mathbf{y}|\mathbf{x})$ will result. Thus, this mode of interaction is merely a filtering where the output gestures of user B just seem smoothed.

8.4 Human and Machine

Of course, the main objective of the ARL architecture is to get an interactive system. Therefore, consider the case where user B has stepped out of the scene and his vision system is disabled. The system's role is now to synthesize B's presence in a continuous way instead of merely predicting a small step into the future or filtering user B's signal. Of course, without loss of generality, we are assuming that user B is to be impersonated. However, by symmetry, all the models apply equally well if A leaves the room and B is interacting alone with the system.

8.4.1 Interaction Mode

In Figure 8.3 one can see the system as it synthesizes interactive behaviour with a single user. User A is given the illusion of interacting with user B through the synthesis of the ARL system. The vision system on A still takes measurements and these integrate and feed the learning system. However, the output of the learning system is *also* fed back into the short term memory. It fills in the missing component (user B) inside $\mathbf{x}$. Thus, not only does user A see synthesized output, continuity is maintained by feeding back synthetic measurements. This gives the system the ability to see its own actions and maintain self-consistent behaviour. The half of the time series that used to be generated by B is now being synthesized by the ARL system. The $\mathbf{x}(t)$ is continuously updated allowing good estimates of $\hat{\mathbf{y}}$. In fact, the ARL prediction only computes small steps into the future and these deltas do not amount to anything on their own unless integrated and accumulated. Since the attentional window which integrates the $\mathbf{y}$ measurements is longer than a few seconds, this gives the system enough short term memory to maintain consistency over a wide range of gestures and avoids instability.

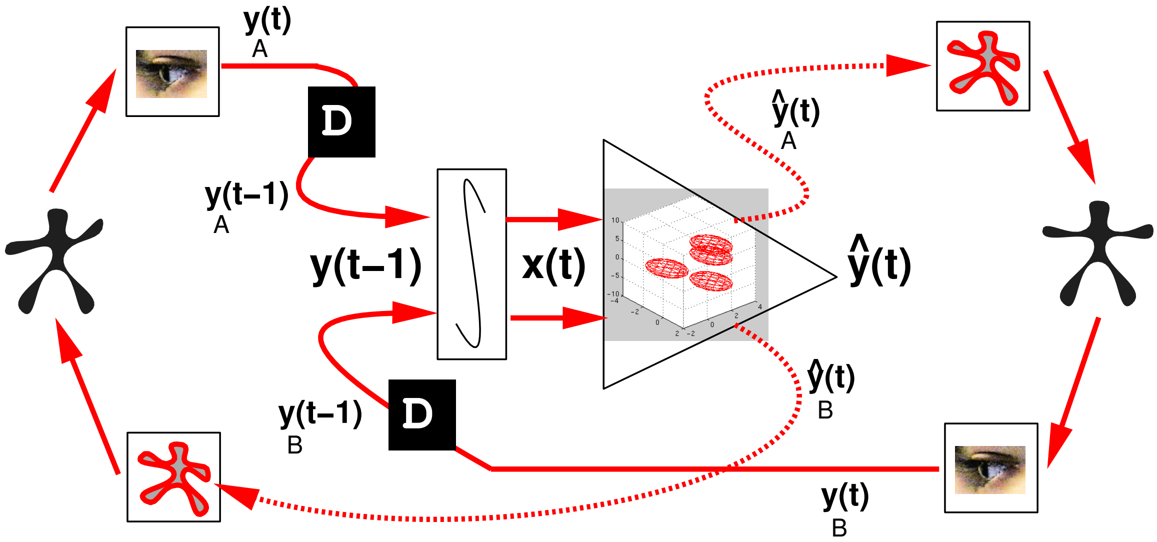


Figure 8.2: Prediction Mode

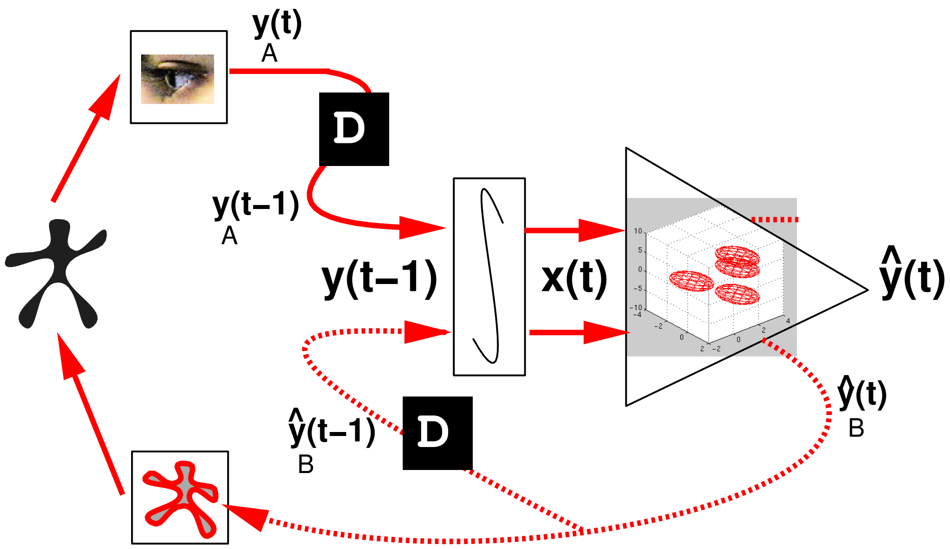


Figure 8.3: Interaction Mode

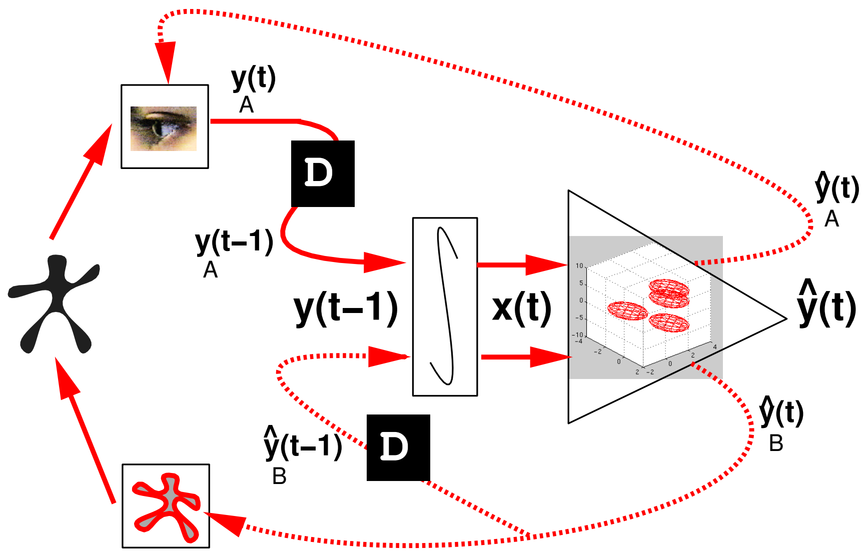


Figure 8.4: Perceptual Mode

Of course, for the initial few seconds of interaction, the system has not synthesized any actions and user A has yet to gesture. Thus, there are no $\hat{\mathbf{y}}$ vectors and no accumulated short term memory. Therefore, the unobserved first few seconds of the time series are set to reasonable default values. The system eventually bootstraps and stabilizes when a user begins interacting with it and output is fed back.

Simultaneously, the real-time graphical blob representation is used to un-map the predicted perceptual (the $\hat{\mathbf{y}}_B(t)$ action) for the visual display. It is through this display that the human user receives feedback in real-time from the system's reactions. This is necessary to maintain the interaction which requires the human user to pay attention and respond appropriately. For the head and hand tracking case, the graphics system is kept simple and merely renders blobs in a 1 to 1 mapping. It displays to the user only what it perceives (three blobs). The ARL system's primary concern is head and hand position and this becomes clear from the coarse display. In addition, the user is not as misled into expecting too much intelligence from such a simple output.

8.4.2 Perceptual Feedback Mode

Of course, the learning system generates *both* a $\hat{\mathbf{y}}_B(t)$ and a $\hat{\mathbf{y}}_A(t)$. Therefore, it would be of no extra cost to utilize the information in $\hat{\mathbf{y}}_A(t)$ in some way while the system is interacting with the user. Recall earlier the brief discussion of the similarity of this output to that of a filter (i.e. a Kalman filter). Instead of explicitly using Kalman filters in the vision systems (as described earlier), one could consider the predicted $\hat{\mathbf{y}}_A(t)$ as an alternative to filtering and smoothing. The ARL system then acts as a sophisticated non-linear dynamical filter. In that sense, it could be used to help the vision tracking and even resolve some vision system errors.

Typically, tracking algorithms use a variety of temporal dynamic models to assist the frame by frame vision computations. The most trivial of these is to use the last estimate in a nearest neighbour approach to initialize the next vision iteration. Kalman filtering and other dynamic models involve more sophistication ranging from constant velocity models to very complex control systems. Here, the the feedback being used to constrain the vision system results from dynamics *and* behaviour modeling. This is similar in spirit to the mixed dynamic and behaviour models in [47]. In the head and hand tracking case, the system continuously feeds back prediction estimates of the 15 tracked parameters (3 Gaussians) in the

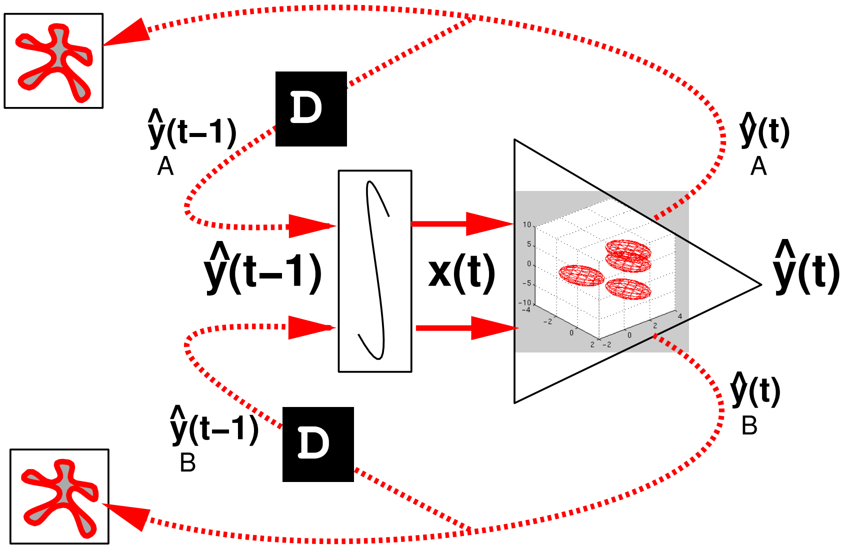


Figure 8.5: Simulation Mode

vision system for improved results.

More significant vision errors can also be handled. Consider the specific case of head and hand tracking with skin blobs. As mentioned earlier, colored gloves were used to overcome some correspondence problems when heads and hands touched and moved by each other. The first training sequences involved no mis-correspondence due to explicit glove labeling of head, left hand and right hand. However, once appropriately trained, the probabilistic model described above feeds back the positions of the Gaussians to the vision. This prevents blob mislabeling by using the whole gesture as a predictor instead of short range dynamics. Thus, it is possible to recognize a blob as a hand from its role in a gesture and to maintain proper tracking. This permits us to reliably do away with colored gloves. In addition, a coarse model of $p(\mathbf{x})$ is available and can be evaluated to determine the likelihood of any past interaction. If permutations of the blobs being tracked by the computer vision are occasionally tested with $p(\mathbf{x})$, any mislabeling of the blob features can be detected and corrected. The system merely selects one of the 6 permutations of 3 blobs that maximizes $p(\mathbf{x})$ and then feeds back the appropriate $\hat{\mathbf{y}}$ estimate to the computer vision. Instead of using complex static computations to resolve these ambiguities, a reliable correspondence between the blobs is computed from temporal information.

8.5 Machine and Machine

Although this is not the intended use of the system, it is possible in principle to have the system simulate behaviour without any external stimulus from humans. Since it has learned from human-human interactive behaviour, it is possible for it to fully re-synthesize this type of behaviour. *Both* user A and user B can be impersonated, effectively acting out a fully virtual interaction scenario for onlookers.

8.5.1 Simulation Mode

As shown in Figure 8.5, the predicted user's action $\hat{\mathbf{y}}_A$ could be fed back into the time series doing away with any human input altogether. Note the perfectly symmetric paths of the data flow. There are no vision systems in operation. Instead, we only see two graphics systems which depict the ARL system's actions. The actions are fed back into its short-term memory which represents the past virtual actions of both user A and user B.

Unlike the previous operation mode which had at least one human generating a component of the signal, here, both components are synthesized. Thus, there is no 'real' signal to ensure some kind of stability. Since both signals are virtual, the ARL system is more likely to exhibit instabilities. There is no real data to 'pull it back down to Earth' and these instabilities can grow limitlessly. Therefore, unless properly initialized, the system will not bootstrap. Furthermore, even when beginning to show some interaction, the system often locks up into some looping meaningless behaviour. Instabilities arise since both halves of the time series are completely synthesized with no real human tracking. Therefore, some modifications are being investigated for zero-user, two-computer configurations. However, this is not the most important mode of operation and remains a lower priority.

Thus we have enumerate several different modes of operation the ARL system can encompass. These include behaviour learning, interaction, simulation, prediction and filtering. The analysis of the ARL framework at a modular level allows such abstractions as well as the use of alternate modules and different data flow paths.

8.6 Practical Issues in an Integrated System

To maintain a real-time integrated system, the many constraints that results from each module must be considered. The perceptual (vision) and output (graphics) systems must both be real-time to avoid lag and the consequent complexities it might cause. In addition, the learning system must generate its prediction $\hat{\mathbf{y}}(t)$ in real-time as well.

The constraints on the speed of the learning system have implications on the complexity it can have. If the pdf used is too complex, computing $\hat{\mathbf{y}}(t)$ might be slow and the system will stall. This strongly reduces the quality of the interaction. Thus, the dimensionality of $\mathbf{x}$ must be chosen carefully. Due to the fact that principal components analysis (PCA) was used, it is straightforward to reduce the dimensionality of the space by using fewer eigenvectors. In addition, due to the use of a conditioned mixture model as a pdf, it is also straightforward to reduce the learning system's complexity by throwing away models.

On the other hand, if high accuracy is required and real-time interaction is not important, more dimensions and more models can be used. Thus, many possible configurations exists as we vary the learning system's power to obtain different solutions. For instance, if only a very simple interaction is to be observed, one might be able to use fewer models and fewer dimensions. However, if the human to human interaction is extremely involved and lasts over many minutes, a more complex model might be in order. The ease with which modeling resources can be increased or decreased allows for this range of operation.

Finally, we address the training of the learning algorithm with CEM. As is well known in the machine learning community, using models that are too complex results in over fitting and causes poorer generalization. Meanwhile, over-simplified models cause under fitting and again poor generalization. Thus, there is another critical issue that arises as dimensionality and complexity are varied.

Given the annealing possibility that was presented earlier, it is possible to avoid some of the over fitting and under fitting problems by using CEM to find a more global estimate. This is typically a good way to address some of these complexity issues. If properly annealed, the learning algorithm is more likely to avoid degenerate and over-fit solutions. However, it is also important to initialize learning algorithms in a good way such that they converge well to a desired solution.

To summarize, the integrated system still has some subtleties that need to be addressed and is not a black box. There are some parameters that influence its efficiency, complexity, effectiveness, etc. and there are some principled ways to address these (refer to [5] [50]).

Chapter 9

Interaction and Results

Having discussed all the components and their integration into a complete system, we illustrate a practical application of ARL. The system is used to acquire interactive behaviour in a constrained scenario and results are presented for evaluation purposes. Of course, many other applications exist given this framework and some further possible scenarios are enumerated as well. Finally, a discussion of the system's limitations is presented. This covers some of the inherent shortcomings of the ARL approach and the conceptual problems in the recovery of behaviour in an automatic manner.

9.1 Application Scenario

It is prudent to train the ARL system in a constrained context to achieve some kind of learning convergence from limited data and limited modeling resources. Thus, the users involved in training the system initially are given some loose instructions on the nature of the interactions they will be performing. The users were given the instructions listed in Table 9.1.

The two users (A and B) begin by playing the above game and A gestures while B responds appropriately. The users are physically separated from each other and only see graphical representations of each other on their screens. The learning algorithm is given measurements of the head and hand positions of both users. These measurements are taken off of the players for several minutes of interaction. These sequences generate many input-output pairs $(\mathbf{x}, \mathbf{y})$. The data pairs are used to train the system which is then able to impersonate player B. Once the training is complete, the B gesturer leaves and the single user remaining is A. The screen display for A still shows the same graphical character except now the actions of the character are synthesized by the ARL system as opposed to the other player.

More specifically, the training process involved between 5 to 10 of each of the above interactions and

Interaction	User	Corresponding Action
1	A	Scare B by moving towards camera
	B	Fearfully crouch down & bring hands in
2	A	Wave hello
	B	Wave back accordingly
3	A	Circle stomach & tap head
	B	Clap enthusiastically
4	A	Idle or Small Gestures
	B	Idle or Small Gestures

Table 9.1: Interaction Instructions

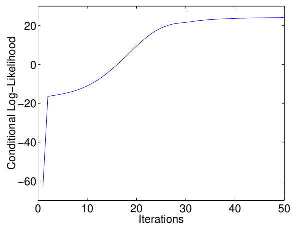


Figure 9.1: Conditional Log Likelihood on ARL Data

lasted roughly 5 minutes. This accounts for slightly over 5000 observations of the 30 dimensional $\mathbf{y}(t)$ vectors. Each of these form an $(\mathbf{x}, \mathbf{y})$ where the $\mathbf{x}$ was the eigen-representation of the past short term memory over T exponentially decayed samples. The dimensionality of $\mathbf{x}$ was only 22 and the short term memory was over 120 samples ($T = 120$ or over 6 seconds). The system used 25 Gaussians for the pdf. The limitations on dimensionality and number of Gaussians where mainly for speed considerations. The learning (CEM algorithm) took approximately 2 hours to converge on an SGI OCTANE for the 5 minute training sequence of interactions. An annealing schedule of exponential decay with a $\tau = 0.01$ per iteration was used. If no annealing is used, an inferior (and less global) solution can be obtained in well under one hour. At the end of the annealed training (roughly 400 iterations) the conditional log-likelihood was approximately 25.7. Figure 9.1 shows the convergence of the annealed CEM algorithm.

9.2 Evaluation

Due to the quantitative and qualitative aspects of the system, two evaluation methods were used. One involved measuring the system's accuracy at prediction (i.e. the prediction mode discussed in the previous chapter) which is measurable numerically. The more important test, the interaction mode performance, was verified visually with real-time interaction with the synthesized ARL behaviours.

9.2.1 Quantitative Prediction

For a quantitative measure, the system was trained as usual on the interaction between the two individuals and learned the usual predictive mapping $p(\mathbf{y}, \mathbf{x})$. Since real-time is not an issue for this kind of test, the system was actually permitted to use more Gaussian models and more dimensions to learn $p(\mathbf{y}, \mathbf{x})$.

Once trained on a portion of the data, the system's ability to perform prediction was tested on the remainder of the sequence. Once again, the pdf allows us to compute an estimated $\hat{\mathbf{y}}$ for any given $\mathbf{x}$ short term memory. The expectation was used to predict $\hat{\mathbf{y}}$ and not the arg max since it is the least squares estimator. The prediction was then compared to the true $\mathbf{y}$ result in the future of the time series and RMS errors were computed. Of course, the system only observed the immediate past reaction of both user A and user B which is contained in $\mathbf{x}$. Thus, the values $\mathbf{y}(t-1), ..., \mathbf{y}(t-T)$ are effectively being used to compute $\mathbf{y}(t)$. In addition, the system is predicting the immediate reaction of *both* users (A and B) in the whole $\mathbf{y}(t)$ vector. For comparison, RMS errors are shown against the nearest neighbour and constant velocity estimates. The nearest neighbour estimate merely assumes that $\mathbf{y}(t) = \mathbf{y}(t-1)$ and the constant velocity assumes that $\mathbf{y}(t) = \mathbf{y}(t-1) + \Delta_t \dot{\mathbf{y}}(t-1)$.

Table 9.2 depicts the RMS errors on the test interaction and these suggest that the system is a better instantaneous predictor than the above two methods. Therefore, it should be useful as a Kalman filter-type of predictor for helping tracking systems.

Nearest Neighbour	Constant Velocity	ARL
1.57 %	0.85 %	0.62 %

Table 9.2: RMS Errors

Figure 9.2: Scare Interaction

9.2.2 Qualitative Interaction

In addition, real-time online testing of the system's interaction abilities was performed. A human player performed the gestures of user A and checked for the system's response. Whenever the user performed one of the gestures in Table 9.1, the system responded with a (qualitatively) appropriate animation of the synthetic character (the gesture of the missing user B). By symmetry, the roles could be reversed such that the system impersonates user A and a human acts as user B. However, the role of user A is more active and user B has a more reactive or passive role. Therefore, it seems more interesting for a human player to be in charge and to trigger the system into performing reactions.

In Figure 9.2, a sample interaction where the user 'scares' the system is depicted. Approximately 500ms elapse between each frame and the frames are arranged lexicographically in temporal order. The user begins in a relaxed rest state and the synthetic character as well. Then, the user begins performing a menacing gesture, raising both arms in the air and then lowering them. The synthetic character responds by first being taken aback and then crouching down in momentary fear. This is the behaviour that was indicated by examples from the human-to-human training. Moreover, the responses from the system contain some default pseudo random variations giving them a more compelling nature. The character can sometimes get fidgety and will shake a bit or flail before entering the crouching position.

Figure 9.3, contains a gesture where the system has to respond to a greeting wave by the user. The user begins in a relaxed rest state and this induces that same rest state in the synthetic character.

Figure 9.3: Waving Interaction

Then, almost immediately when the user begins waving, the system also waves back. Both perform the periodic gesture for a few frames. However, when the user stops waving, the system decays its wave and stabilizes back to rest state. In fact, this is the desired behaviour since user B was typically in a rest state whenever user A did no particular motion. Also note that the synthetic character begins waving immediately when the user starts extending his arm sideways. This indicates that the gesture is easily discriminated just from this initial motion. Unlike the previous gesture in Figure 9.2, the waving involves more of a gesture-to-gesture mapping. The scaring motion is merely a single beat motion and does not involves any periodicity or oscillations which are difficult to represent and to synthesize stably by feedback interaction.

A more involved form of interaction is depicted in Figure 9.4. Here, the user stimulates the character by circling his stomach while patting his head. The system's reaction is to clap enthusiastically when the user accomplishes this slightly tricky and playful gesture. Once again, the system stops gesturing when the user is still (as is the case at the beginning and at the end of the sequence here). The oscillatory gesture the user is performing is rather different from the system's response. Thus, there is a higher-level mapping: oscillatory gesture to different oscillatory gesture.

In the above examples, the user is *delegating* tasks to the animated character since it is not a simple 1-to-1 mapping between the current measurements to output. The user produces a complex action and the system responds with a complex reaction. The response depends on user input as well as on the system's previous internal state. The mapping thus associates measurements over time to measurements

Figure 9.4: Clapping Interaction

over time which is fundamentally a higher dimensional problem.

Also, note that the system acquired these behaviours from a *minimal* amount of data. Only **5** minutes of training and a handful of exemplars of each gesture were given to the ARL system. This is impressive considering that the gestures and interactions where not segmented or ordered. Useful associations had to be teased out of a continuous stream of interactions automatically. Larger training sets were not explored at this stage due to time constraints. However, given these initial results, a larger data set is an important next step. The learning time should also increase however the ARL system might yield more interesting results and more complex learning. In addition, the CEM algorithm could be sped up considerably, thus encouraging the use of more data.

9.3 Alternate Applications

Other applications beyond head and hand interactions are also under investigation. We shall enumerate a few concepts and leave it to the reader's imagination to conjure other scenarios. One of the virtues about the ARL system and its perceptual, data-driven nature is that it is flexible. There are no explicit mechanisms here to exclusively learn head and hand dynamics. The only systems that were specific to head and hand motion were the vision and graphics system. The time series processing and learning algorithm did not specifically require such types of data. In the learning system, there were no cognitive or kinematic models that concerned gesticulations or hand gesture constraints. Although such models could have been helpful additions in this scenario, they could be detrimental when the ARL learns a different modality (i.e. facial motion). The lack of such hard-wired models increases the generality of the approach. Thus, a large space of time varying measurements and outputs can be handled and the behaviours recovered could have had a markedly different structure.

9.3.1 Alternate Modalities

An interesting set of alternate modalities could be explored in addition to head and hand tracking. Audio is a particularly interactive and expressive channel and should lend itself well to compelling behaviour synthesis scenarios. Coarse textural properties of an audio stream (such as wah-wahs or simple sounds) could be used to generate action-reaction pairs for interaction learning.

9.3.2 Intra-Modal Learning

Another important concept also arises that is equally well supported by the framework: intra-modal learning. By intra-modal learning, we are referring to the acquisition of a coupling between gestures or actions in one domain with those of another domain. For instance, User A could feed the system with audio measurements (pitch energy, textural properties) while User B could feed the system with visual gestures. As the system learns how to predict the time-series of both measurements, it begins to form a mapping between audio and video. Thus, instead of learning that a clap should follow when the user rubs his stomach (as demonstrated above), the system could trigger clapping when the user sings a nice melody into a pitch tracker. Evidently, there are many unresolved issues here but the important notion to stress is that User A and User B *do not* have to have the same type of measurements. In other words, their respective measurements ($\mathbf{y}_A$ and $\mathbf{y}_B$) could contain observations of different types of data.

9.3.3 Non-Human Training Data

One other note is that the systems described here are not specific to human interaction (although it is a compelling task). The interaction of other processes is also of interest. This includes the behaviour of animals ranging from the simple to the complex. One could consider learning the interaction of a cat and mouse as they run around the floor. If a simple overhead vision algorithm could determine their coarse locations, it would be possible to analyze some of their interactive dynamics and predict some trajectories. These concepts could be extended to the interaction of any coupled phenomenon provided a good representation appropriate for the ARL system is available. This could include two airplanes maneuvering in a dog-fight. Given their relative positions, it would be possible to compute non-linear trajectories and simulate the evolution of the engagement (if training data from previous dog-fights is learned).

9.4 Limitations

Despite the many possibilities of this work and the promising initial results, there are some serious limitations that simply can not be over-emphasized. Although some issues are beyond the scope of this work, we discuss them briefly and put them forth as topics of future concern.

9.4.1 Difficulty with Discrete Events

Due to the representation of the continuous time series, there is no explicit machinery for addressing discrete events. Consider for example, a binary switching event such as the output of a classifier. This could be the output of a speech detection system which determines if the user speaking or not-speaking. In a time series, such a phenomenon will typically be represented as a step function. In the ARL, this could be potentially harmful due to the projection on the eigenspace. The onset of a step function is a highly non-linear and localized transition and eigenspace projection will unnaturally smooth it into a sigmoid like curve. Thus, one would require an alternative way of representing sudden changes which might involve fundamental changes to the time series representation.

9.4.2 Lack of Modularity

The various components of the short term memory window in the ARL system do not have a modular property. For example, the system could observe three gestures in a certain order: gesture A, then gesture B, then gesture C. Even though all three are in the system's short term memory, it will not parse these into 3 separate components and recognize them independently. Therefore, there is no higher order abstraction about the evolution through event A, B then C. This abstraction and modularity of signals is reminiscent of linguistic processing. Therefore, the rearrangement of different chunks or modular components can not be understood by the system. Grouping lower order signals into sophisticated hierarchies and dependencies involves the ability to segment and rearrange them. These mechanisms do not exist in the system as is. There might be possibilities to reformulate the architecture such that this type of learning can be encompassed.

9.4.3 Constant Length Memory and State

Once again, the short term memory does provide some self-consistency but it is only a rudimentary form of state information. The memory is finite and covers a fixed window of a few seconds. It is not a full state model since the *states* here do not correspond to meaningful transitions or fundamentally different modes of operation. Thus, if no events of significance occur for a few seconds, the ARL system forgets its current state and starts off fresh. A finite state automaton will not 'forget' its current discrete state and might remain in it indefinitely until an appropriate transition is triggered. In addition, the continuous representations of the ARL's short-term memory causes some spatially driven clustering in the $\mathbf{x}$ eigenspace. In a Hidden Markov Model (HMM), on the other hand, states are clustered in terms of their output probabilities and their generative characteristics. This is a more functional notion of state. Therefore, events that occur adjacently in time but have very different outputs will be partitioned by an HMM. However, the ARL clustering might not separate the two and cluster the events due to their spatio-temporal proximity (i.e. *not* their functional proximity). The notion of state can be included in the CEM algorithm if it is extended to include Hidden Markov Models in the pdf (as in addition to Gaussians).

9.4.4 Well-Behaved, Smooth Representations are Critical

Representation is critical in the ARL system since it must be carefully selected to achieve learning. Invariance in the system must be introduced a priori into the representation. For example, if a feature is spurious and contributes no information about the interaction, it will waste resources. The system will wastefully attempt to model and predict it inside the $\mathbf{y}$ vector. In addition, the representations must be smooth and must not have ambiguities. For example, during initial phases of development, the ARL system employed a different representation of the head and hand blobs. The head and hands were described by their mean, major axis, minor axes and rotation (in radians). Unlike the square-root covariance shape descriptor (our current representation), the rotation value had some unusual singularities. The 0 and 2π values are identical in radian notation. Thus, the system would generate non-linear steps from 0 to 2π as the blobs would rotate and these transitions were difficult to span using the eigenspace temporal processing techniques. Thus, it is critical to pick a representation which contains the desired invariants, is well behaved, is smooth and has no spurious components.

9.4.5 No High Level Goals, Motivations, or Sequences of Actions

The Action-Reaction Learning framework is clearly only recovering lower-order reactionary behaviour. The actions and reactions are mapped in an almost direct manner and the short-term memory really only contains the last stimulus that triggered the system. Thus, higher order behaviours that involve complex chains of actions are not addressed. For instance, there are no mechanisms for dealing with sequences of actions that are strung together into some more complex meta-action. There is also no representation of the actions forming sub-goals and these forming then goals in a sequential evolution. The ARL system does not identify goals, plans or motivations which are some of the essential concepts in higher order behaviour. Therefore, we only addressed a basic definition of behaviour as a stimulus-response mapping. The transition to higher order behaviour is an interesting issue and remains a topic of future work.

Chapter 10

Conclusions and Contributions

10.1 Current and Future Work

The following are future projects that are being developed to extend the ARL system.

10.1.1 Continuous Online Learning

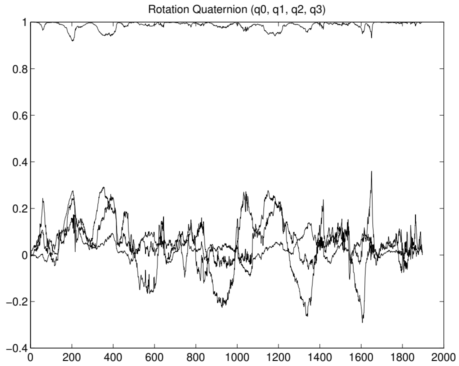
While learning in the current ARL system implementation is a batch process, it is possible to extend it into an online version. The system could then accumulate more training data and learn while it is operating in interaction mode. The CEM algorithm could, in principle, run as a background process while new data is acquired and while the system synthesizes interactions. Using some straightforward reformulations, an online CEM would update its mixture of conditional models dynamically as it obtains new samples. Recall that, in interaction mode, the system is interacting with only a single human. However, fundamentally, the same type of training data can still be recovered: $(\mathbf{x}, \mathbf{y})$ pairs. Even though some components of the data are synthetic, half are real and result from a human who is engaging the system. Significant learning is possible with this data as the system acquires a model of what the human user does in response to its synthetic actions. Of course, the system has to be initially trained offline with human-human interactions to bootstrap some behaviour. Eventually, though, the system can be placed in an *interactive learning* mode. Herein the system would continue learning by dynamically acquiring new responses to stimuli and includes these in its dictionary of possible reactions. This would make it adaptive and its behaviour would be further tuned by the engagement with the single remaining user. This mode of operation is currently under investigation.

10.1.2 Face Modeling for Interaction

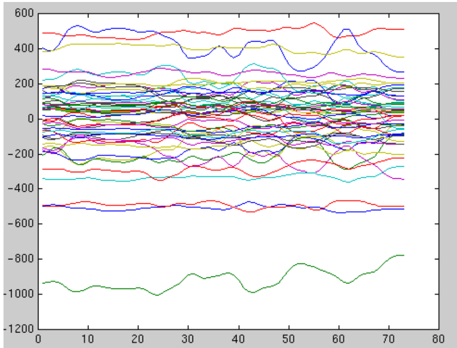
An alternative visual input is also being evaluated. Face modeling can be used to recover subtle facial detail (beyond blob tracking) for a more convincing interaction. A system which automatically detects the face and tracks it has been implemented [27]. It is capable of tracking the 3D rotations and movements of a face using normalized correlation coupled with structure from motion. In addition, at each moment in time, it computes an eigenspace model of the face's texture. This texture description is used to infer corresponding 3D deformations statistically [28]. This system generates a real-time temporal sequence which includes XYZ translations, 3D rotations as well as a set of texture and deformation scalar values (in an eigenspace). Figure 10.1 depicts the face tracking algorithm and examples of the temporal sequences being output in real-time.

To synthesize an output, a 3D renderer reconstructs a 3D facial model in real-time using estimated deformation coefficients, texture coefficients, rotations and translations. The sample output is shown in

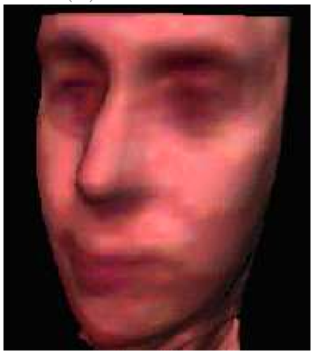
(a) Tracking



(b) Quaternions



(c) 3D Eigen Deformations



(d) 3D Synthesis

Figure 10.1: 3D Face Modeling and Tracking

Figure 10.1(d). The data representing each static frame can again be a time series (with 50 dimensional features) and the above ARL system analysis is currently being applied to this platform.

10.2 Conclusions

We have demonstrated a real-time system which learns two-person interactive behaviour automatically by modeling the probabilistic relationship between a past action and its consequent reaction. The system is then able to engage in real-time interaction with a single user and impersonates the missing person by estimating and simulating the most likely action to take. The system is data driven, autononomous, and perceptually grounded. In addition, the imitation-based behaviour learning is capable of synthesizing compelling interaction and synthetic behaviour with minimal structural specifications. The subsequent interaction is real-time, non-intrusive and computationally efficient. These were the objectives outlined in the introduction and were met in the ARL design process.

We have demonstrated a synthetic character which was able to engage in waving, clapping and other interesting gestures in response to the user. The synthetic character learned passively without any specific behavioural programming. In addition, a probabilistic model of the synthetic behaviour was performed and RMS errors were used to evaluate its usefulness as a prediction system. It was shown to form a superior predictor than nearest-neighbour and constant velocity assumptions. Finally, a novel approach for function optimization, GBM (Generalized Bound Maximization) was presented. It was used to derive CEM (a Conditional Expectation Maximization algorithm) for estimating maximum conditional likelihood probability densities. The algorithm was demonstrated to be better suited to computing conditional densities (Gaussian mixture models, in particular) than the EM algorithm. The CEM algorithm also performed successfully on standard test databases. More detailed implementations were discessed for the CEM algorithm and further derivations were cast using GBM machinery. The resulting learning algorithm was shown to be a monotonically convergent conditional density estimator and succesfully utilized for the Action-Reaction Learning framework.

The main contributions of the thesis are:

- The Action-Reaction Learning Paradigm

  The framework and the implementation of the Action-Reaction learning system was proposed and implemented to automatically acquire and synthesize human behaviour from perceptual data.

- Real-Time Head and Hand Tracking

  An expectation-maximization-based head and hand tracker was developed for robust real-time recovery of gestures even under occlusion and contact of head and hands.

- Temporal Representation

  A compact time series representation of past interaction data (i.e. a short term memory) was developed to represent the input to the learning system. The representation was shown to be useful for various sorts of temporal data.

- Conditional versus Conditioned Joint Estimation

  An analysis of the differences between conditional density estimation and conditioned joint density estimates was presented and a Bayesian formalism for both was developed.

- Generalized Bound Maximization

  A bounding technique was introduced which extends some of the variational bounding principles and other algorithms with an emphasis on quadratic bounds. Local and annealed optimization results were shown on a wide class of functions.

- Conditional Expectation Maximization

  An algorithm was derived for computing the maximum conditional likelihood estimate for probability densities. In addition, the implementation for a conditioned mixture of Gaussians was presented in detail.

- Integration and Modes of Operation

  The ARL was formulated as a modular framework which involves the integration of various interchangeable components and data flow between them. Different modes of operation were developed including human-human, human-computer and computer-computer types of interaction. The ARL architecture was shown to encompass interaction, learning, simulation, filtering and prediction of behaviour.

The Action-Reaction-Learning framework analyzed and synthesized human behaviour from perceptual data. A probabilistic conditional model of behaviour was uncovered by learning input and output interactions discriminantly. There were no underlying generative models of behaviour and the user did not specify explicit rules or behavioural mechanisms. The system simply learned behaviour by looking at humans from the outside.

Chapter 11

Appendix

11.1 Conditional and Unconditional Bayesian Integration

In this appendix, we carry out the result shown earlier relating the Bayesian estimation of conditional and joint densities. In presenting these two inference problems, we discussed how they might lead to different solutions despite the fact that they both involve exact techniques (integration and conditioning). In addition, some speculation about the superiority of the direct conditional estimate was made. In the following, we present a specific example to demonstrate this difference and to argue in favor of the conditional estimate $p(y|x)^c$ versus the conditioned joint estimate $p(y|x)^j$.

To prove this point, we use a specific example, a simple 2-component mixture model. Assume the objective is to estimate a conditional density, $p(y|x)$. This conditional density is a conditioned 2-component 2D Gaussian mixture model with identity covariances. We try to first estimate this conditional density by finding the joint density $p(x,y)$ and then conditioning it to get $p(y|x)^j$. Subsequently, we try to estimate the conditional density directly to get $p(y|x)^c$ without obtaining a joint density in the process. These are then compared to see if they yield identical solutions.

Consider a joint density as a two-element 2D Gaussian mixture model with identity covariance and equal mixing proportions shown in Figure 11.1. We wish to fit this model to data using Bayesian integration techniques. The result will not be significant on its own since this is a trivial learning example. However, we shall check for inconsistencies between this result and direct conditional density estimation to prove a general statement about Bayesian inference. Equation 11.1 depicts the likelihood of a data point (x,y) given the model and Equation 11.2 depicts a wide Gaussian prior (with very large σ^2) on the parameters (m_0, n_0, m_1, n_1). As shown earlier, we wish to optimize this model over a data set $(\mathcal{X}, \mathcal{Y})$. This computation results in a model $p(x,y)$ as in Equation 11.3.

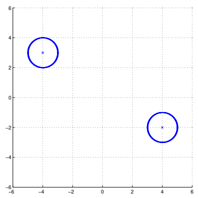


Figure 11.1: 2D Mixture Model

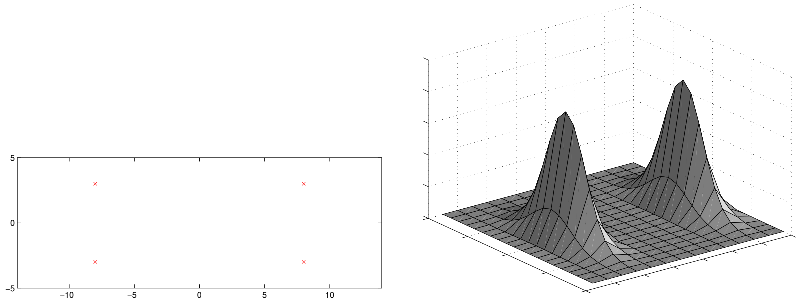
$$
\begin{aligned}
p(x,y|\Theta) &= \frac{0.5}{(2\pi)^{2/2}\sqrt{|I|}}e^{-\frac{1}{2}((x-m_0)^2+(y-n_0)^2)} + \frac{0.5}{(2\pi)^{2/2}\sqrt{|I|}}e^{-\frac{1}{2}((x-m_1)^2+(y-n_1)^2)} \\
p(x,y|\Theta) &= \frac{1}{4\pi}e^{-\frac{1}{2}((x-m_0)^2+(y-n_0)^2)} + \frac{1}{4\pi}e^{-\frac{1}{2}((x-m_1)^2+(y-n_1)^2)}
\end{aligned} \tag{11.1}
$$

$$
\begin{aligned}
p(\Theta) &= \frac{1}{(2\pi)^{4/2}\sqrt{|\sigma^2 \times I|}}e^{-\frac{1}{2}((m_0)^2+(n_0)^2+(m_1)^2+(n_1)^2)/\sigma^2} \\
p(\Theta) &= \frac{1}{5\pi^2}e^{-\frac{1}{2\sigma^2}((m_0)^2+(n_0)^2+(m_1)^2+(n_1)^2)}
\end{aligned} \tag{11.2}
$$

$$
\begin{aligned}
p(x,y) &= p(x,y|\mathcal{X},\mathcal{Y}) \\
&= \int p(x,y|\Theta)p(\Theta|\mathcal{X},\mathcal{Y})d\Theta \\
&= \int p(x,y|\Theta)\Pi_{i=1}^{N} p(\Theta|x_i,y_i)d\Theta \\
&= \int p(x,y|\Theta)\Pi_{i=1}^{N} \frac{p(x_i,y_i|\Theta)p(\Theta)}{p(x_i,y_i)}d\Theta \\
&\propto \int p(x,y|\Theta)p(\Theta)^N \Pi_{i=1}^{N} p(x_i,y_i|\Theta)d\Theta \\
&\propto \int (\tfrac{1}{4\pi}e^{-\frac{1}{2}((x-m_0)^2+(y-n_0)^2)} + \tfrac{1}{4\pi}e^{-\frac{1}{2}((x-m_1)^2+(y-n_1)^2)}) \\
&\quad \times(\tfrac{1}{5\pi^{2N}}e^{-\frac{N}{2\sigma^2}((m_0)^2+(n_0)^2+(m_1)^2+(n_1)^2)}) \\
&\quad \times \Pi_{i=1}^{N}(\tfrac{1}{4\pi}e^{-\frac{1}{2}((x_i-m_0)^2+(y_i-n_0)^2)} + \tfrac{1}{4\pi}e^{-\frac{1}{2}((x_i-m_1)^2+(y_i-n_1)^2)})d\Theta \\
&\propto \int (e^{\alpha}+e^{\beta})(e^{\delta})\Pi_{i=1}^{N}(e^{\gamma_i}+e^{\omega_i})d\Theta \\
&\propto \int (e^{\alpha}+e^{\beta})(e^{\delta})((e^{\gamma_1}+e^{\omega_1})(e^{\gamma_2}+e^{\omega_2})...(e^{\gamma_N}+e^{\omega_N}))d\Theta \\
&\propto \int\int\int\int (e^{\alpha}+e^{\beta})(e^{\delta})((e^{\gamma_1}+e^{\omega_1})(e^{\gamma_2}+e^{\omega_2})...(e^{\gamma_N}+e^{\omega_N}))dm_0dn_0dm_1dn_1 \\
&\propto \int\int\int\int \sum_{\sigma} e^{\alpha+\delta+\sum_{i=1}^{N}\text{choose}_{\sigma}[\gamma,\omega]_i} + \sum_{\sigma} e^{\beta+\delta+\sum_{i=1}^{N}\text{choose}_{\sigma}[\gamma,\omega]_i} dm_0dn_0dm_1dn_1
\end{aligned} \tag{11.3}
$$

In Equation 11.3 we are effectively summing over all the permutations σ of the assignments of the N data points to the 2 different models. For each σ, we select a different assignment of the i data points. Each point gets assigned to one of 2 Gaussians (one related to γ and the other related to ω). The summation over the data in each exponential can be further simplified as in Equation 11.4 and then analytically integrated. The integrals are summed over all possible assignments of the data points to one of the two Gaussians (i.e. M^N possibilities or integrals where $M = 2$ models here). Essentially, we are iterating over all possible permutations where the data points are assigned to the two different Gaussians all 2^N different ways and estimating the Gaussians accordingly. Evidently this is a slow process and due to the exponential complexity growth, it can not be done for real-world applications. Figure 11.2 shows some data and the Bayesian multivariate mixture model estimate of the probability density $p(x, y)$. Figure 11.3 shows the conditional density $p(y|x)^j$.

$$
\begin{aligned}
&\int\int\int\int e^{\beta+\delta+\sum_{i=1}^{N}\text{choose}_{\sigma}[\gamma,\omega]_i} dm_0dn_0dm_1dn_1 = \\
&\quad \int\int\int\int \exp(-\tfrac{1}{2}((x-m_0)^2+(y-n_0)^2+\gamma_{xx}-2\gamma_x m_0+\gamma_N m_0^2+ \\
&\qquad \gamma_{yy}-2\gamma_y n_0+\gamma_N n^2+\omega_{xx}-2\omega_x m_1+\omega_N m_1^2+\omega_{yy}-2\omega_y n_1+\omega_N n_1^2))dm_0dn_0dm_1dn_1 \\
&\text{where :} \\
&\ \gamma_N = \textstyle\sum_{i=1}^{N} \text{chose}[\gamma] \\
&\ \gamma_x = \textstyle\sum_{i=1}^{N} \text{chose}[\gamma]x_i \quad \gamma_{xx} = \sum_{i=1}^{N} \text{chose}[\gamma]x_i^2 \\
&\ \gamma_y = \textstyle\sum_{i=1}^{N} \text{chose}[\gamma]y_i \quad \gamma_{yy} = \sum_{i=1}^{N} \text{chose}[\gamma]y_i^2 \\
&\ \omega_N = \textstyle\sum_{i=1}^{N} \text{chose}[\omega] \\
&\ \omega_x = \textstyle\sum_{i=1}^{N} \text{chose}[\omega]x_i \quad \omega_{xx} = \sum_{i=1}^{N} \text{chose}[\omega]x_i^2 \\
&\ \omega_y = \textstyle\sum_{i=1}^{N} \text{chose}[\omega]y_i \quad \omega_{yy} = \sum_{i=1}^{N} \text{chose}[\omega]y_i^2 \\
&\text{analytic integration :} \\
&\int\int\int\int e^{\beta+\delta+\sum_{i=1}^{N}\text{choose}_{\sigma}[\gamma,\omega]_i} dm_0dn_0dm_1dn_1 = \\
&\quad \exp(-(\tfrac{1}{4}\omega_{yy}\gamma_N\omega_N+\tfrac{1}{4}\omega_{xx}\gamma_N\omega_N+\tfrac{1}{4}\gamma_{yy}*\gamma_N\omega_N+\tfrac{1}{4}x^2\gamma_N\omega_N-\tfrac{1}{4}\omega_x^2\gamma_N+\tfrac{1}{4}\gamma_{xx}\gamma_N\omega_N \\
&\qquad +\tfrac{1}{4}y^2\gamma_N\omega_N-\tfrac{1}{4}\omega_y^2\gamma_N+\tfrac{1}{4}\omega_{yy}\omega_N-\tfrac{1}{4}\gamma_x^2\omega_N+\tfrac{1}{4}\gamma_{xx}\omega_N+\tfrac{1}{4}\gamma_{yy}\omega_N+\tfrac{1}{4}\omega_{xx}\omega_N \\
&\qquad\quad -\tfrac{1}{2}x\gamma_x\omega_N-\tfrac{1}{4}\omega_x^2-\tfrac{1}{2}y\gamma_y\omega_N-\tfrac{1}{4}\gamma_y^2\omega_N-\tfrac{1}{4}\omega_y^2)/(\tfrac{1}{2}+\tfrac{1}{2}\gamma_N)/\omega_N)\pi^2/\omega_N/(\tfrac{1}{4}\gamma_N+\tfrac{1}{4})
\end{aligned} \tag{11.4}
$$



(a) Training Data (b) Joint Density Estimate

Figure 11.2: Joint Density Estimate

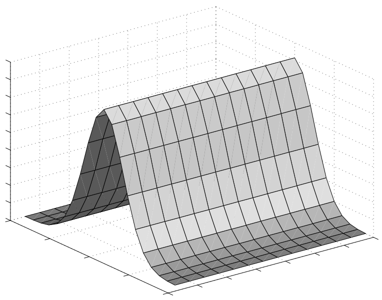


Figure 11.3: Conditioned Joint Density Estimate

By solving another integration, we can directly compute the conditional density $p(y|x)^c$. The conditional density has the form shown in Equation 11.5. This is just a regular 2-component conditioned mixture of Gaussians model with identity covariances. Assume that we are using the same prior as before. In addition, note the presence of the exact same parameters m_0, n_0, m_1, n_1 which reside in the conditioned parametrization of Θ which can be called Θ^c.

$$p(y|x,\Theta^c) = \frac{p(x,y|\Theta)}{p(x|\Theta)} = \frac{\frac{1}{4\pi}e^{-\frac{1}{2}((x-m_0)^2+(y-n_0)^2)}+\frac{1}{4\pi}e^{-\frac{1}{2}((x-m_1)^2+(y-n_1)^2)}}{\frac{1}{2\sqrt{\pi}}e^{-\frac{1}{2}(x-m_0)^2}+\frac{1}{2\sqrt{\pi}}e^{-\frac{1}{2}(x-m_1)^2}} \tag{11.5}$$

The resulting Bayesian integration is depicted in Equation 11.6. Unfortunately, integration can only be completed analytically for the parameters n_0 and n_1. Thus, the integration over the other 2 parameters is performed using numerical approximation techniques. The inner integral of n_0 and n_1 causes the exponentially complex assignment permutation seen above and this is compounded with the computation of the integral numerically by a grid approach. This is therefore an even more cumbersome computation than the joint density Bayesian estimate and is only shown here as an example. There exist more efficient numerical integration techniques such as superior quadrature approaches or Monte-Carlo methods however this Bayesian integration approach is typically too intensive for any real-world applications. It should be noted that typically, Bayesian density estimation, Bayesian sampling techniques and Bayesian integration are quite cumbersome except in very special situations.

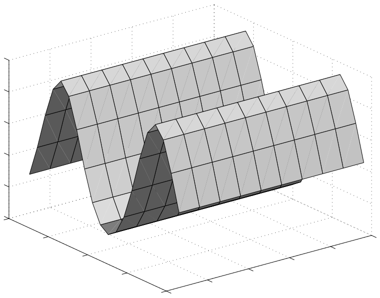


Figure 11.4: Direct Conditional Density Estimate

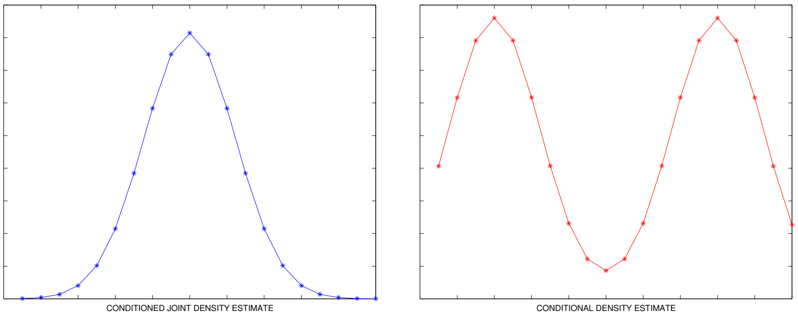


Figure 11.5: Slices of $p(y|x)^j$ and $p(y|x)^c$ at $x = -5$

$$
\begin{aligned}
p(y|x) &= p(y|x, \mathcal{X}, \mathcal{Y}) \\
&\propto \int p(y|x,\Theta^c)\Pi_{i=1}^N p(y_i|x_i,\Theta^c)p(\Theta^c)d\Theta^c \\
&\propto \int p(y|x,\Theta^c)p(\Theta^c)^N \Pi_{i=1}^N p(y_i|x_i,\Theta^c)d\Theta^c \\
&\propto \int p(y|x,\Theta^c)p(\Theta^c)^N \Pi_{i=1}^N p(x_i,y_i|\Theta^c)(\Pi_{i=1}^N p(x_i|\Theta^c))^{-1}d\Theta^c \\
&\propto \int\int\int\int p(x,y|\Theta^c)(p(x|\Theta^c))^{-1}p(\Theta^c)^N \Pi_{i=1}^N p(x_i,y_i|\Theta^c)(\Pi_{i=1}^N p(x_i|\Theta^c))^{-1}dm_0dn_0dm_1dn_1 \\
&\propto \int\int (p(x|\Theta^c)(\Pi_{i=1}^N p(x_i|\Theta^c)))^{-1}\int\int p(x,y|\Theta^c)p(\Theta^c)^N \Pi_{i=1}^N p(x_i,y_i|\Theta^c)dn_0dn_1dm_0dm_1
\end{aligned}
\tag{11.6}
$$

The same data is thus fitted with the conditional model which produces the conditional distribution $p(y|x)^c$ shown in Figure 11.4. Surprisingly, this is quite different from the conditioned joint density. In fact, if we consider a slice of the conditional densities at an arbitrary x value, we obtain the y-distributions shown in Figure 11.5. This indicates that the directly computed conditional model was able to model the bi-modal nature of the data while the conditioned joint density model was not. In fact, $p(y|x)^c$ seems like a better choice than $p(y|x)^j$.

The above suggests the following. Consider the case of two Bayesian statisticians (A and B) who are asked to model a conditional density (i.e. in a classification task or a regression task) from data. Statistician A assumes that this conditional density arises from a joint density. He then estimates this density using full Bayesian inference. He then conditions this joint density and obtains the final conditional density he was asked to produce. Statistician B assumes *nothing* about the origins of the conditional density and estimates it directly. He only uses a parametric form for a conditional density, it is just a function. At the end of the day, the two have different models even though all the manipulations they performed where valid equalities (Bayesian inference and conditioning are exact manipulations). Thus, by a strange by-product of the paths the two statisticians took, they got two different answers: $p(y|x)^c \neq p(y|x)^j$.

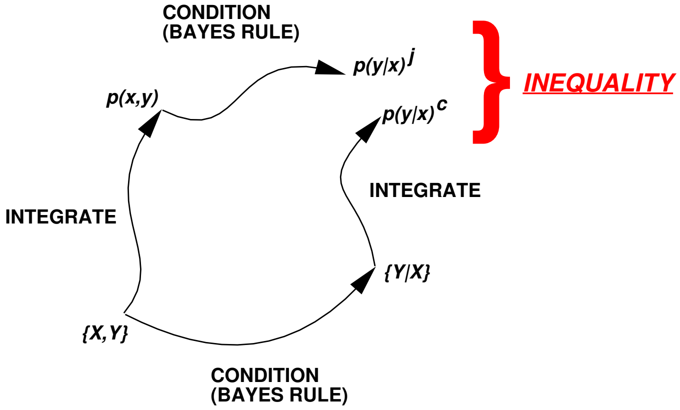


Figure 11.6: Inconsistency in the Bayesian Inferences

Typically $p(y|x)^c$ seems to be a more robust estimate and this is probably because no extra assumptions have been made. In assuming that the original distribution was a joint density which was being conditioned, statistician A introduced unnecessary constraints [1] from the space of $p(x,y)$ and these prevent the estimation of a good model $p(y|x)$. Unless the statisticians have exact knowledge about the generative model, it is typically more robust to directly estimate a conditional density than try to recover some semi-arbitrary joint model and condition it. Figure 11.6 graphically depicts this inconsistency in the Bayesian inference shown above. Note here that the solutions are found by fully Bayesian integration and *not* by approximate MAP or ML methods. Thus, the discrepancy between $p(y|x)^j$ and $p(y|x)^c$ can not be blamed on the fact that MAP and ML methods are just efficient approximations to exact Bayesian estimation.

[1] In fact, in assuming a joint density, the model is a parent of the $\mathcal{X}$ data instead of being a co-parent with it.

Bibliography

[1] A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3-d shape estimation from blob features. In *International Conference on Pattern Recognition (ICPR)*, 1996.

[2] A. Azarbayejani, C. Wren, and A. Pentland. Real-time 3-d tracking of the human body. In *Proceedings of IMAGE'COM 96*, May 1996.

[3] N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press, 1993.

[4] J. Bates, B. Loyall, and S. Reilly. An architecture for action, emotion and social behaviour. Technical Report CMU-CS-92-144, School of Computer Science, Carnegie Mellon University, 1992.

[5] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford Press, 1996.

[6] B. Blumberg, P. Todd, and P. Maes. No bad dogs: Ethological lessons for learning. In *From Animals To Animats, Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, 1996.

[7] A. Bobick. Computers seeing action. In *Proceedings of British Machine Vision Conference*, 1996.

[8] A. Bobick. Movement, activity, and action: The role of knowledge in the perception of motion. In *Proceedings of Royal Society: Special Issue on Knowledge-based Vision in Man and Machine*, 1997.

[9] C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[10] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proc. ACM SIGGRAPH 97*, 1997.

[11] R. Brooks. Intelligence without reason. Technical Report A.I. Memo No. 1293, M.I.T. Artificial Intelligence Laboratory, 1991.

[12] R. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20(2-4), 1997.

[13] D. Clark, Z. Schreter, and A. Adams. A quantitative comparison of dystal and backpropagation. In *Australian Conference on Neural Networks*, 1996.

[14] S. R. Cooke, B. Kitts, R. Sekuler, and M. Mataric. Delayed and real-time imitation of complex visual gestures. In *Proceedings of the International Conference on Vision, Recognition, Action: Neural Models of Mind and Machine*, 1997.

[15] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B39, 1977.

[16] S. Elliott and J. R. Anderson. The effect of memory decay on predictions from changing categories. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 1995.

[17] L. Emering, R. Boulic, S. Balcisoy, and D. Thalmann. Real-time interactions with virtual agents driven by human action identification. In *First ACM Conf. on Autonomous Agents'97*, 1997.

[18] A. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden markov models with mixed states. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction*, 1993.

[19] J. Funge. *Making Them Behave: Cognitive Models for Computer Animation.* PhD thesis, University of Toronto, Graduate Department of Computer Science, 1997.

[20] N. Gershenfeld and A. Weigend. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993.

[21] Z. Ghahramani and M. Jordan. Learning from incomplete data. Technical Report 1509, MIT Artificial Intelligence Lab, 1995.

[22] G. Hager and P. Belhumeur. Real time tracking of image regions with changes in geometry and illumination. In *CVPR96*, pages 403–410, 1996.

[23] G. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8(1), January 1997.

[24] M. Isaard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Sixth International Conference on Computer Vision*, 1998.

[25] T. Jaakkola and M. Jordan. Computing upper and lower bounds on likelihoods in intractable networks. Memo 1571, MIT A.I. Laboratory, 1996.

[26] T. Jebara. Action reaction learning home page. http://jebara.www.media.mit.edu/people/jebara/arl.

[27] T. Jebara and A. Pentland. Parametrized structure from motion for 3d adaptive feedback tracking of faces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[28] T. Jebara, K. Russel, and A. Pentland. Mixtures of eigenfeatures for real-time structure from texture. In *Proceedings of the International Conference on Computer Vision*, 1998.

[29] F. Jensen. *An Introduction to Bayesian Networks*. Springer, 1996.

[30] M. Jordan. *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[31] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[32] M. Jordan and L. Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural Networks*, 8:1409–1431, 1993.

[33] L. Kaelbling and M. Littman. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[34] P. Killeen. Mathematical principles of reinforcement. *Behavioral and Brain Sciences*, 17, 1994.

[35] K. Lashley. The problem of serial order in behavior. In L. Jefress, editor, *Cerebral Mechanisms in Behavior*, pages 112–136, New York, 1951. The Hixon Symposium, John Wiley.

[36] S. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

[37] P. Maes, T. Darrel, B. Blumberg, and A. Pentland. The alive system: Wireless, full-body interaction with autonomous agents. *Special Issue on Multimedia and Multisensory Virtual Worlds, ACM Multimedia Systems*, 1996.

[38] J. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, 1988.

[39] M. Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *From Animals to Animats: International Conference on Simulations of Adaptive Behavior*, 1992.

[40] M. Mataric. Learning motor skills by imitation. In *Proceedings, AAAI Spring Symposium Toward Physical Interaction and Manipulation*, 1994.

[41] M. Mataric. Cooperative multi-robot box-pushing. In *Proceedings, IROS-95*, 1995.

[42] X. Meng and D. Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2), 1993.

[43] M. Mozer. Neural net architectures for temporal sequence processing. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction*, 1993.

[44] N. Oliver, A. Pentland, F. Berard, and J. Coutaz. Lafter: Lips and face tracker. In *Computer Vision and Pattern Recognition Conference '97*, 1997.

[45] A. Ortney, G. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, UK, 1988.

[46] A. Pentland. Machine understanding of human action. Technical Report 350, M.I.T. Media Laboratory - Vision and Modeling, 1995.

[47] A. Pentland and A. Liu. Modeling and prediction of human behavior. In *IEEE Intelligent Vehicles 95*, 1995.

[48] A. Popat. *Conjoint Probabilistic Subband Modeling*. PhD thesis, M.I.T. Media Laboratory, 1997.

[49] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C, Second Edition*. Cambridge University Press, 1994.

[50] J. Rissanen. Modelling by the shortest data description. *Automatica*, 1978.

[51] G. Rizzolatti, L. Fadiga, B. Galles, and L. Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3, 1997.

[52] R. Rockafeller. *Convex Analysis*. Princeton University Press, 1972.

[53] D. Roy, M. Hlavac, M. Umaschi, T. Jebara, J. Cassell, and A. Pentland. Toco the toucan: A synthetic character guided by perception, emotion, and story. In *Visual Proceedings of Siggraph*, page 66, 1997. Additional authors added after press time: B. Tomlinson, C. Wren.

[54] J. Rustagi. *Variational Methods in Statistics*. Academic Press, 1976.

[55] J. Rustagi. *Optimization Techniques in Statistics*. Academic Press, 1994.

[56] B. Schiele and A. Waibel. Gaze tracking based on face color. In *International Workshop on Automatic Face and Gesture Recognition*, pages 344–349, 1995.

[57] K. Sims. Evolving virtual creatures. In *Proceedings of SIGGRAPH '94*, volume 26, 1994.

[58] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *International Workshop on Automatic Face and Gesture Recognition*, 1995.

[59] D. Terzopoulos, X. Tu, and G. R. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[60] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6), 1993.

[61] K. Thorisson. *Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills.* PhD thesis, Massachusetts Institute of Technology, Media Laboratory, 1996.

[62] E. Thorndike. Animal intelligence. an experimental study of the associative process in animals. *Psychological Review, Monograph Supplements*, 2(4):109, 1898.

[63] S. Thrun and L. Pratt. *Learning to Learn.* Kluwer Academic, 1997.

[64] Y. Tong. *The Multivariate Normal Distribution.* Springer-Verlag, 1990.

[65] E. Uchibe, M. Asada, and K. Hosoda. State space construction for behaviour acquisition in multi agent environments with vision and action. In *Proceedings of the International Conference on Computer Vision*, 1998.

[66] N. Ueda and R. Nakano. Deterministic annealing variant of the em algorithm. In *Advances in Neural Information Processing Systems 7*, 1995.

[67] E. Wan. Time series prediction by using a connectionist network with internal delay lines. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction*, 1993.

[68] J. Watson. Psychology as the behaviorist views it. *Psychological Review*, 20:158–17, 1913.

[69] S. Waugh. *Extending and benchmarking Cascade-Correlation.* PhD thesis, University of Tasmania, Computer Science Department, 1995.

[70] J. Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the Association for Computing Machinery*, 9, 1966.

[71] A. Wilson and A. Bobick. Recognition and interpretation of parametric gesture. In *International Conference on Computer Vision*, 1998.

[72] A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 24, 1988.

[73] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), 1997.

[74] L. Xu and M. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8:129–151, 1996.

[75] Y. Yacoob and L. Davis. Learned temporal models of image motion. In *Proceedings of the International Conference on Computer Vision*, 1998.