TOPICS IN COMPUTATIONAL HIDDEN STATE MODELING

Peter N. Yianilos

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE

June 1997

Abstract

Motivated by the goal of establishing stochastic and information theoretic foundations for the study of intelligence and synthesis of intelligent machines, this thesis probes several topics relating to hidden state stochastic models.

Finite Growth Models (FGM) are introduced. These are nonnegative functionals that arise from parametrically-weighted directed acyclic graphs and a tuple observation that affects these weights. Using FGMs the parameters of a highly general form of stochastic transducer can be learned from examples, and the particular case of stochastic string edit distance is developed. Experiments are described that illustrate the application of learned string edit distance to the problem of recognizing a spoken word given a phonetic transcription of the acoustic signal. With FGMs one may direct learning by criteria beyond simple maximum-likelihood. The MAP (maximum *a posteriori estimate*) and MDL (minimum description length) are discussed along with the application to causal-context probability models and unnormalized noncausal models. The FGM framework, algorithms, and data structures describe hidden Markov models, stochastic context free grammars, and many other conventional similar models while providing a unified and natural way for computer scientists to learn and reason about them and their many variations. A software system and scripting language is proposed to serve as an assembly language or sorts for many higher level model types.

This thesis also illuminates certain fundamental aspects of the nature of normal (Gaussian) mixtures and the reparameterization of related optimization problems. The use of conditional normal mixtures is proposed as a tool for image modeling, and issues relating to the estimation of their parameters are discussed.

Acknowledgments

I must begin by thanking my first mentor Joeseph Vargo, and my parents and grandparents who valued education and always encouraged me to follow my own interests. I also thank my advisor Bob Tarjan and Bruce Maggs for urging me to complete my graduate studies at Princeton.

Many of the results of this thesis arose from a collaboration with Eric Ristad and I acknowledge his important influence on and contributions to this research. I am also grateful for the participation and advice of my other committee members Andrew Appel, Bob Sedgewick, and Vince Poor.

Finally, this work would not have been possible without the support and patience of my wife Angela and our children Nicholas, Jonathan, Geoffrey, and Lauren.

Contents

Chapter 1

Introduction

Determining the computational nature of animate intelligence is perhaps the greatest single challenge facing the field of computer science. Many approaches are possible, and will no doubt be necessary if this riddle is to be solved. These range from the study of biological nervous systems, to analysis of the tasks that creatures perform – and frequently include a constructive component in which a machine is built or programmed to exhibit behavior that is in some sense intelligent. Here a distinction emerges. When the focus is entirely on successful performance of a particular task, one is led to consider designs that in all likelihood shed little light on the central question above. In its pure form this pursuit is then the study of *artificial intelligence*. That is, machines and software engaged in a masquerade – resembling in some respects the object of their emulation, but made of essentially different *stuff*. By contrast the study of *synthetic intelligence* strives towards constructions that while man-made, are nevertheless intelligent in the natural and animate sense. Of course in practice the distinction between synthetic and artificial is often hard to make, and really represents a difference in research direction and motivation. Moreover, while we may seem to attach a pejorative connotation to “artificial” in the discussion above, it certainly may be that machines will emerge from this work that are unquestionably intelligent in a deeply different sense from that with which we are today familiar. Nevertheless, our main interests lie along the synthetic direction.

Natural intelligence is impressively robust in the presence of noise and uncertainty. One view is that such factors should be dealt with by an *outer wrapper* of techniques, exposing an inherently discrete and symbolic problem at the core. Indeed there is a long history of investigators focusing on problems such as theorem proving and logical reasoning, while avoiding the less easily framed problem of robustness. Our view is that the manner in which nature deals with this more elusive problem may represent the central idea behind intelligence, not just a front-end noise filter.

Stochastic modeling techniques represent a formal approach to the problems of noise and uncertainty – and have been somewhat successful when applied to difficult problems such as speech recognition and other signal and image processing tasks.

Chapter 2 introduces the Finite Growth Model (FGM) framework which spans many existing model classes and opens up important new possibilities. Among these is the notion of stochastic transduction in which a machine converts one observation into another. The probability of transduction between two objects can be thought of as an indication of their similarity. A characteristic of natural intelligence is its use of nontrivial metrics, i.e. notions for similarity. Another striking feature is that these metrics are sometimes learned. Both of these are possible within the stochastic transduction paradigm. We remark that the well-known concept of string edit distance may be viewed as a single state memoryless transducer, and our work provides a convenient way to optimize its cost parameters. Speech recognition may be viewed as a grand transduction from signal to text. Chapter 3 describes experiments that represent a first step towards approaching the problem using this formalism. Finite growth models also include the class of hidden Markov models and stochastic context free grammars which can provide a means to discover hidden structure in a a set of observations. This corresponds to another salient characteristic of natural intelligence. FGMs also allow the model designer to cope with the learning-theoretic considerations of overtraining and generalization by building data-appropriate models resulting from optimization criteria such as that of minimum description length (MDL) or the maximum *a posteriori* probability estimate (MAP).

While interesting and perhaps practical, the stochastic modeling techniques we introduce would seem to be more *artificial* than synthetic – since a common language for this field is that of linear algebra, an unlikely component of our biological endowment. Beyond this observation, it seems unlikely that nature would confine herself to the use of strict probabalistic models.

An important contribution of chapter 2 is the presentation of FGMs, and by inclusion many specific stochastic model classes, in terms of weighted graphs and a related optimization problem. These constructions need not represent causal probability models – or probabilities at all. Nevertheless the celebrated Baum-Welch and EM algorithms are shown to still apply, and we argue that their essential message is one of decomposition. That is, breaking up a particular graph-based global optimization problem into a set of local problems such that progress on the local problems will necessarily advance the global objective. These observations paint FGMs in a far more connectionist light and it is for this reason that we see it as at least plausible that nature might employ related principles.

Chapter 4 sketches the design of a software library and language for FGMs. Experimentation in expedited by effective tools, and we argue that our design should be viewed as an *assembly language* for computational hidden-state stochastic modeling.

The material of chapters 5 and 6 is of a more esoteric and perhaps artificial nature. Chapter 5 exposes a fascinating and somewhat counterintuitive degeneracy in the relationship between the prior and *a posteriori* distributions arising from a mixture of normal densities. This degeneracy is then exploited to prove a reparameterization theorem that provides a modicum of theoretical justification to learning approaches that proceed by reweighting the input pattern set. In chapter 6 we discuss approaches to the learning of continuous context models.

We submit that a stochastic and information-theoretic paradigm for intelligence is emerging; although it is not clear whether the insights it generates pertain to the synthetic or only to the artificial face of the problem.

Chapter 2

Finite Growth Models

Finite growth models (FGM) are nonnegative functionals that arise from parametrically-weighted directed acyclic graphs and a tuple observation that affects these weights. The weight of a source-sink path is the product of the weights along it. The functional's value is the sum of the weights of all such paths. The mathematical foundations of hidden Markov modeling (HMM) and expectation maximization (EM) are generalized to address the problem of functional maximization given an observation. Probability models such as HMMs and stochastic context free grammars are examples that satisfy a particular constraint: that of summing or integrating to one. The FGM framework, algorithms, and data structures describe these and other similar stochastic models while providing a unified and natural way for computer scientists to learn and reason about them and their many variations.

Restricted to probabilistic form, FGMs correspond to stochastic automata that allow observations to be processed in many orderings and groupings – not just one-by-one in sequential order. As a result the parameters of a highly general form of stochastic transducer can be learned from examples, and the particular case of string edit distance is developed.

In the FGM framework one may direct learning by criteria beyond simple maximum-likelihood. The MAP (maximum *a posteriori probability estimate*) and MDL (minimum description length) are discussed along with the application of FGMs to causal-context probability models and unnormalized noncausal models.

2.1 Introduction

Hidden discrete-state stochastic approaches such as hidden Markov models (HMM) for time series analysis, stochastic context free grammars (SCFG) for natural language, and statistical mixture densities are used in many areas including speech and signal processing, pattern recognition, computational linguistics, and more recently computational biology. These are parametric models and are typically optimized using the Baum-Welch, inside-outside, and expectation maximization (EM) algorithms respectively. They share a common mathematical foundation and are shown to be instances of a single more general abstract recursive optimization paradigm which we refer to as the *finite growth model* framework (FGM) involving non-negative bounded functionals associated with finite directed acyclic graphs (DAG). These functionals need not be probabilities and an FGM need not correspond to a stochastic process.

With FGMs interesting new kinds of stochastic models may be designed, existing models may be improved, settings that are not strictly probabalistic in nature may be addressed, and it becomes easier to design and reason about a wide range of problems. This chapter[1] introduces and develops the FGM framework and then applies it to:

1. the definition of *k-way stochastic transducers* whose parameters may be conveniently optimized based on training data;

2. the special case of 2-way string transduction which is shown to correspond closely with the notion of *string edit distance* where parameter optimization corresponds to learning insert, delete, and substitute costs that best explain a training corpus of similar strings;

[1]This chapter first appeared as a technical report [RY96a].

3. the inclusion of model complexity tensions such as *minimum description length* (MDL) and *maximum a posteriori probability* (MAP) estimation within models such as HMMs while retaining the ability to easily optimize them;

4. show that context dependent observation functions may be used to extend the power of conventional stochastic models while preserving the convenience of parameter reestimation;

5. consider problems that are not strictly probabilistic in nature such as parameter estimation for unnormalized noncausal image models and investment portfolio optimization; and

6. provide a unified framework for understanding existing models such as SCFGs and HMMs along with their many variations by giving time and space efficient reductions to FGMs.

Graphical models [Pea88, SHJ96] represent a branch of related work in which the independence structure of a set of random variables is the main focus. One can express such models using FGMs – as well as considerably more general settings such as that in which no two variables are independent (their graphical model is fully connected), but they are jointly constrained in interesting ways.

An FGM is a directed acyclic graph with a single source and sink together with a collection of non-negative bounded weight functions $\{w_i\}$ that are associated with edges – one per edge. The value of a source-sink path is the product of the weights along it and the FGM's value is the sum over all such paths of their values. Section 2.2 presents a formal development but we will sketch the basic ideas here.

The weight function $w_i(z_e|\Psi_i)$ associated with edge e accepts an argument z_e and parameters Ψ_i. The objective of FGM optimization is to maximize the FGM's value over its parameters $\{\Psi_i\}$ which are assumed to be independent. The argument z_e is regarded as fixed. The same weight function may be associated with several edges but will in general assume different values since z_e is a function of the edge to which it is attached.

If each edge has a distinct associated weight function then optimization is immediately reduced to the task of independently maximizing the weight functions in isolation. The interesting situation arises when this is not the case. Here a change to a single parameter set Ψ may affect weights throughout the DAG. The values of many source-sink paths may be affected and an interaction between members of $\{\Psi\}$ arises, i.e. simple independent maximization is no longer possible.

Surprisingly it is still possible to decompose the optimization problem into certain independent subproblems. This is the key mathematical message of Baum-Welch and EM generalized to the FGM framework. However, even exact solution of the subproblems does not yield a maximum for the FGM. But a new combined parameter set does result that strictly improves it — unless one has already converged to a limiting value. Iteration is then used to *climb* to such a limiting value. Unlike gradient descent with line search this approach confidently moves to a sequence of increasingly better points in parameter space and in many cases reduces to a strikingly simple and intuitive algorithm. The cost functions may themselves be FGMs and the computation of an improved parameter set simply depends upon the ability to solve the primitive maximization problems at the bottom of the recursive hierarchy.[2]

The mathematical idea behind this decomposition begins with a simple equality having to do with the log of a sum of terms. Let $S = T_1 + \ldots + T_n$ and define proportions $P_i = T_i/S$. Notice $S = T_i/P_i$. For any nonnegative vector R with unity sum:

$$\begin{aligned}\log S &= \sum_i R_i \log S && (1)\\ &= \sum_i R_i \log T_i/P_i && (2)\\ &= \sum_i R_i \log T_i - \sum_i R_i \log P_i && (3)\end{aligned}$$

[2]We must remark that it is important to realize that convergence to a limiting value for the FGM is not the same as convergence of its parameters.

Observe that the second term is maximized when $P = R$ since both are stochastic vectors.

In our setting each term T_i corresponds to a source-sink path, and its value is not constant, but rather is a function of some parameter set Ψ'. Optimizing the FGM is then the problem of maximizing their sum over Ψ'. This is the same as optimizing their log sum; connecting the equality above to the problem of FGM optimization.

Accordingly we parameterize the equality by writing $S(\Psi') = T_1(\Psi') + \ldots + T_n(\Psi')$ and $P(\Psi') = T_i(\Psi')/S(\Psi')$ where our objective is to maximize S over Ψ'. The search for a maximum takes place iteratively. During each iteration we denote the *current* parameters by Ψ and vary Ψ' looking for an improved set.

To express this iterative approach the equality is specialized by defining R to be P evaluated at the current parameters Ψ, i.e. $R \triangleq P(\Psi)$. Then:

$$\log S(\Psi') = \sum_i P_i(\Psi) \log T_i(\Psi') - \sum_i P_i(\Psi) \log P_i(\Psi')$$

The right term may be ignored because any departure from Ψ will decrease it whereby $\log S$ and therefore S are increased. We then focus on maximizing the left term which corresponds to the Q function of the HMM/EM literature.

Elements of the remaining summation correspond to source-sink paths through the FGM's DAG and the proportions $P_i(\Psi)$ are the relative contribution of each path to the FGM's value. Each term $T_i(\Psi')$ is then a product of of weight functions and breaks up into a sum of individual log terms. The proportions are then distributed over them and the final step is to group terms by weight function. The problem is then decomposed as required into independent subproblems each involving a single weight function.

The argument above is made in much greater detail in section 2.2 and the formal definition of an FGM presented there is more involved in two respects. First, there are two weight functions, not one, associated with each edge. These are abstractions of the normally separate transition and observation generation functions of a stochastic automaton. Second, the z_e in our discussion above is defined to be some function of an observation tuple x. A simple such function is that of coordinate projection where, for example, the function's value might be that of a single dimension of x.

The contribution of section 2.2 is the development of a framework that has nothing directly to do with stochastic models or probability in which the essential ideas of Baum-Welch and EM nevertheless apply. Our view is that the essential content of these ideas is one of decomposability of certain optimization problems that are captured in a somewhat general way by our definition of an FGM.

The early literature, especially theorem 2.1 of [BPSW70], reveals an awareness that the mathematics of HMMs apply beyond strictly probabalistic settings. This direction of generalization does not however appear to have been followed until now. Thinking seems to have been dominated by the motivating problem: the optimization of Markovian stochastic models. Because of the limited computational capabilities of the time and their highly mathematical viewpoint a great deal of emphasis was also placed on the primitive weight functions for which exact maximization may be performed in *closed form*, and those which give rise to unique global maxima. By contrast we devote very little attention to the the specific characteristics of these primitive functions and view them instead as an abstract types that by assumption support a set of required operations.

In section 2.3 the FGM framework is specialized to the case of probabilistic models. The z_i above result from projections that select one or more coordinate dimensions such that along each source-sink path each dimension is selected exactly once. Also, the weight functions are restricted to be probability functions or densities. It is not important that the dimensions occur in any fixed order as one follows alternative source-sink routes; nor is it required that the dimensions be selected one at a time. The contribution of this outlook is that FGM-based stochastic models can generate observations in many orderings and groupings – not just one-by-one in sequential order as in HMMs – a capability exploited later in our discussion of stochastic transducers.

Section 2.4 begins by considering the case of infinite stochastic processes. Viewed generatively these models produce output of unbounded size but are used in practice to evaluate the probability of finite observations and are trained using a finite set of finite observations. Restricted in this way they correspond to FGMs that capture their *truncated* form. Moreover the conventional evaluation and learning algorithms associated with them arise naturally from this viewpoint.

Baum-Welch and EM are today generally regarded as *maximum-likelihood* parameter estimation techniques. A more modern *learning theoretic* perspective recognizes the important tension between model complexity and available training data. An important contribution of section 2.4 is the description of a construction that modifies the FGM associated with a stochastic model so that alternative criteria such as MDL or MAP can guide optimization. The optimization is still based on Baum-Welch/EM so we suggest that these should *not* be viewed as statistical ML techniques and that their broader utility is revealed by our more general functional optimization perspective. The applicability of EM to MAP was first described in [DLR77]. Nevertheless, the parameters of models such as HMMs are still estimated in the ML sense.

The observation functions associated with each state in a conventional Hidden Markov model must satisfy an independence assumption: that their state dependency is limited to the current state (the Markov condition). Section 2.4 observes that conditioning on earlier portions of the observation sequence does not violate this assumption and demonstrates that using the FGM framework, it is easy to see that parameter reestimation for such context dependent HMMs decomposes into primitive context optimization problems. This generalizes Brown's use [Bro87] of the previous speech window to condition generation of the current one. Thus the power of causal context models such as statistical language models may be added to the HMM framework and may lead to improved performance.

The section also presents another illustration that the reach of FGMs extends in a useful way beyond conventional and strictly stochastic models. Causal context models express the probability of an observation as a product of conditional probabilities such that each dimension is predicted once and does appear earlier as a conditioning variable. These restrictions are constrictive in settings such as image modeling where modeling each pixel based on its surrounding context seems natural. Simply relaxing the requirements that probabalistic modeling places on an FGM's source-sink projection functions proves that FGM optimization will improve even noncausal models like the image example above.

Stochastic models describe the manner in which an object is generated. This object is conventionally imagined to be something like an image, an audio recording, or a string of symbols. If instead one models pairs of such objects the focus shifts from describing the nature of an individual to uncovering the relationship between them. The pairs may be thought of as instances of the *are similar* concept and learning an effective model to describe them is an example of what the authors have called the *metric learning* paradigm.

Fixing the left element of the pair one might then ask what generative sequences could explain the right. This is the essential idea of 2-way stochastic transduction. Section 2.5 shows that a somewhat general automaton-based definition for k-way transduction may be reduced to problems within the FGM framework. One may view speech recognition as a transduction, perhaps in several stages, and the application of learned transducers to this field represents an interesting area for future work.

Perhaps the best known example of 2-way transduction is that of *string edit distance* which is defined as the least costly way to transform one string into another via a given set of weighted insertion, deletion, and substitution editing operations. The development in section 2.5.1 begins with a natural reformulation into stochastic terms first noted by Hall and Dowling in [HD80, P.390-1]. We refer to the result as *stochastic edit distance* [RY96b].

Our focus is on the interesting problem of learning insert, delete, and substitute costs from a training corpus $(s_1, t_1), \ldots, (s_n, t_n)$ of string pairs. The objective is to minimize the total stochastic edit distance of the collection. In [RY96b] the authors give such a learning algorithm independent of the FGM framework and report on experiments. Until now the costs which parameterize edit distance algorithms have been prescribed not learned. Section 2.5.1 of this chapter describes the reduction of this learning problem to the FGM optimization framework and may be viewed as an adjunct to [RY96b] providing a proof of correctness.

The optimized costs are often quite different from the initial ones. It is in this sense that we *learn* them. In previous work, the costs have been stipulated after study of the problem domain. Even worse, maximally uninformative unit costs (i.e., Levenshtein distance) are sometimes stipulated when the problem domain is nontrivial. It is now possible to improve any educated guess, or start virtually from scratch from unit costs, to arrive at a set of optimized ones. Algorithms 7 and 8 of section 2.5.1 run in time proportional to the product of the two string lengths, matching the complexity of the conventional edit distance computation. Since edit distance has found widespread application our discovery of a simple way to learn costs may lead to improved performance in many problem areas.

Section 2.5.1 also observes that the notion of string edit distance may be strengthened so that the cost of an edit operation depends not only on its target but also on context. For example, such context-sensitive string edit distances allow the cost of inserting a symbol to depend on the symbol left-adjacent to the insertion site. The result remains an FGM on one therefore the ability to easily improve costs given a training corpus is retained.

Several kinds of hidden discrete-state stochastic models are in widespread use today. Hidden Markov Models (HMM) were introduced in the 1960's [BE67, BPSW70, Bau72] and are typically applied in time-series settings when it is reasonable to assume that the observed data are generated or approximated well by the output of an automaton which changes state and generates output values stochastically. A matrix-based approach is typically used to formalize this notion, and [Por88] and [HAJ90] provide nice overviews. Normal (Gaussian) mixtures find application in statistical pattern recognition where items to be classified are represented as real-valued vectors. Stochastic context free grammars (SCFG) lead to a natural probabilistic interpretation of natural language parsing. Other model types exist, and are easily formed by combining and modifying the standard ones.

The situation is complicated by *parameter tying*; the practice of reducing the number of parameters in a model by equating two or more different sets of formal parameters to a single set. Observations are sometimes discrete and sometimes continuous. Entire models may be mixed together or recursively combined. Some states or transitions may be *nonemitting* meaning that no observation is generated. Finally, *nonuniform* time-series models may predict several future data points. All of these factors make it difficult to write down a single high-level framework with which to reason about and verify the correctness of a model's associated algorithms.

The contribution of FGMs is to move instead to a lower-level representation which is common to all of the models and complicating variations above. As noted earlier isn't really the original model which is represented, but rather its truncation to the given collection of finite observations. By understanding FGMs it is possible to reason about the others as mappings from their individual description language to an FGM given finite observations.

Section 2.6 relates the models above to FGMs by describing a reduction for each. As further illustration it considers the portfolio optimization problem and shows that the simple iterative algorithm reported in [Cov84] for optimization arises by reducing the problem to an FGM.

The work described in this chapter began in 1993 with our efforts to model handwriting. The focus then was on transduction, in an attempt to evaluate and learn a notion for similarity for the off-line case. This effort exposed many limitations of the conventional modeling outlook – problems that extend far beyond handwriting. To address these limitations we had to break several *rules* of conventional modeling only to ultimately discover that these rules are in many cases unnecessary artifacts of the history that led to HMMs and EM. After three years of generalization, and exercises in specialization, we arrived at the FGM outlook reported here.

2.2 Finite Growth Models

In this section we develop the theory of Finite Growth Models (FGM) as a class of nonnegative functionals represented as annotated directed acyclic graphs. The next section focuses on the special case of probability models. Our formal definition of an FGM amounts to an intricate recursive *data structure*, in the computer science sense, that has been crafted to span a wide range of applications. As a result it may seem at first to be rather abstract and have little to do with stochastic modeling, but each feature is an abstraction of some characteristic required by a problem we considered. It is the mathematical engine at the center of these problems. In particular much of our development and notation parallels that of Hidden Markov Models (HMM).

Let x be a tuple observation with components $x_1, \ldots, x_d$, and $\mathcal{X}$ denote the space of all such observations. We will not specify the type of each component since for much of our development it matters only that appropriate functions exists on the space of values that a component may assume. Components may therefore represent discrete values such as letters of the alphabet, continuous values such as real numbers, or other more complex structures. Within the observation tuple, component types need not be the same. In simple *time series* probability-model settings, components do have the same type, and position within the tuple corresponds to the passage of time.

Definition 1 *A <u>finite growth model</u> F for a d-dimensional observation space $\mathcal{X}$, is an 8-tuple (V, E, M, C, S, m, c, s), such that:*

1. *(V, E) is a finite DAG with vertices $V = v_1, \ldots, v_n$ having a single source v_1 and sink v_n. For $e \in E$ we denote by $s(e)$ the index of its source vertex, and by $d(e)$ the index of its destination vertex.*

2. *M is a finite collection $\{M_i\}$ of parameterized nonnegative bounded <u>observation</u> functionals with corresponding parameters $\{\Psi_i\}$. We write $M_i(y|\Psi_i)$ to denote evaluation of the ith observation functional, where the domain is not yet specified.*

3. *C is a finite collection $\{C_i\}$ of discrete-domain parameterized nonnegative bounded <u>choice</u> functionals. The size of the domain of $C_i(j)$ is denoted $|C_i|$, and the domain consists of the integers $1, \ldots, |C_i|$. The parameters of C_i are denoted Υ_i and we write $C_i(j|\Upsilon_i)$ to denote evaluation of the ith choice functional.*

4. *S is a finite collection $\{S_i\}$ of <u>selection</u> functions with domain X where the range is not yet specified.*

5. *$m : E \to \{1, \ldots, |M|\}$ is a function associating an element of M with each edge.*

6. $c : E \to \{1, \ldots, |C|\} \times \mathbb{N}$ *is a function that associates each edge with a choice function and a member of its range; which is formalized by denoting the projections of* c *by* c_1, c_2 *and requiring:*

   (a) $s(e_1) = s(e_2) \Rightarrow c_1(e_1) = c_1(e_2), \forall e_1, e_2 \in E$, *so that an element of* C *is in effect associated with each nonsink vertex, and;*

   (b) *For each nonsink vertex* v, c_2 *restricted to* $\{e|s(e) = v\}$ *is 1-1 and onto the range of* $C_{c_1(v)}$.

   The choice value corresponding to an edge e *is then* $C_{c_1(e)}(c_2(e)|\Upsilon_{c_1(e)})$ *and is denoted* $C(e|\Upsilon)$ *for brevity where it is understood that in this context* Υ *refers to* $\Upsilon_{c_1(e)}$ *– but the subscript is sometimes written for clarity.*

7. $\varsigma : E \to \{1, \ldots, |S|\}$ *is an function that associates each edge with a selection function such that for all edges* e, *the range of* $S_{\varsigma(e)}$ *matches the domain of* $M_{m(e)}$. *The observation value corresponding to an edge* e *is then* $M_{m(e)}(S_{\varsigma(e)}(x)|\Psi_{m(e)})$ *and is denoted* $M(e, x|\Psi)$ *for brevity where it is understood that in this context* Ψ *refers to* $\Psi_{m(e)}$ *– but the subscript is sometimes written for clarity.*

The combined observation and choice parameters are denoted Ψ *and* Υ *respectively, and* $\Phi \triangleq (\Psi, \Upsilon)$ *denotes the parameters of* F. *The* <u>*value*</u> *of a source-sink path is the product of the observation and choice functional values along it. The* <u>*value*</u> $F(x|\Phi)$ *of the FGM is the sum over all source-sink paths, of the value of each. A choice or observation model whose value is the constant* 1 *does not affect an FGM's value and is therefore said to be* <u>*void*</u>.

Notice that since an FGM is itself a nonnegative parameterized functional, the definition above may be applied recursively to define an FGM whose constituent functionals are themselves FGMs.

Direct evaluation of an FGM's value by enumerating all source-sink paths is of course not practical. Fortunately $F(x|\Phi)$ may be efficiently computed using dynamic programming. To express this precisely requires some notation. Let $v_1, \ldots, v_n$ be a topologically sorted vertex listing. Following the convention of the HMM literature, we define corresponding real variables $\alpha_1, \ldots, \alpha_n$. The sets of edges *into* and *out from* a vertex v are denoted by $\mathcal{I}(v)$ and $\mathcal{O}(v)$ respectively. Using the notation above, the contribution of an edge e to the value of a path containing it, is $C(e|\Upsilon) \cdot M(e, x|\Psi)$ The product of these contributions is the probability of the path. The complete dynamic program is then given by the following simple algorithm:

Algorithm 1

procedure *forward*

$\alpha_1 = 1$

for $i = 2, \ldots, n$

$\alpha_i = \sum_{e \in \mathcal{I}(v_i)} \alpha_{s(e)} \cdot C(e|\Upsilon) \cdot M(e, x|\Psi)$

This may be recognized as the forward step of the Baum-Welch HMM *forward-backward* algorithm [BE67, Bau72, Por88] adapted to the more general FGM setting. Following execution α_n has value $F(x|\Phi)$.

Algorithm 1 is said to *evaluate* the FGM using its current set of parameters. The optimization problem we consider seeks a parameter set that maximizes the FGM's value

$$\overline{\Phi} = \text{argmax}_\Phi F(x|\Phi) \tag{4}$$

but we will be content to find in some sense locally optimal solutions, and in some cases to simply make progress. Here x is fixed and it is the FGM's parameters that are varied.

The end-to-end concatenation of two FGMs clearly results in another whose value is the product of the two, so that given a set of observations $x_1, \ldots, x_m$, the optimization problem

$$\overline{\Phi} = \text{argmax}_\Phi \prod_{i=1}^{m} F(x_i|\Phi) \tag{5}$$

is really just an instance of Eq. 4. Viewing $\{x_i\}$ as training examples the optimization then *learns* a parameter set, or in the language of statistics and assuming the FGM corresponds to a probability model, *estimates* its parameters.

Our approach to this problem extends the idea of Baum-Welch to FGMs. Several years following the development of HMMs, essentially the same mathematical ideas expressed in the their algorithm were rediscovered as the expectation maximization (EM) algorithm for mixture densities [DLR77, RW84]. This work focused on parameter estimation for mixture densities, and has had considerable influence on a somewhat independent community. In what follows we will use the phrase *Baum-Welch/EM* to refer to the adaptation of these results to the FGM setting.

The HMM and EM literature deals with probabalistic models, and Baum-Welch/EM is presented as a method for reestimating their parameters. Our mathematical contribution is the recognition that the method is not essentially a statement about probability but rather should be viewed as an iterative approach to the maximization of arbitrary parameterized nonnegative functionals of a certain form. The form consists of a sum of terms each one of which is a product of subterms. An FGM's DAG is a representation of these forms which, for many with appropriate structure, has the advantages of succinct representation and computational economy. In the interesting case, the subterms are partitioned or grouped into families sharing common parameters. The approach then consists of focusing on each family in isolation thus localizing the problem. Individual family maximization problems are constructed that include certain weighting factors. These factors are constant with respect to the family's optimization but are computed using the entire DAG and are therefore influenced by all families. Thus global information is used to decompose the original problem – localizing it to individual families. If parameters are found that improve any of the individual family problems, the FGM's value will strictly increase. The method is iterative since even exact maximization of every family's problem may not maximize the FGM's value. Instead the process is repeated.

To formalize this idea of simply making progress with respect to a given maximization problem, the notation:

$$\text{argclimb}_{a||b} f(a) \triangleq \{a | f(a) \geq f(b)\}$$

is introduced. Notice that the improvement need not be strict and that the result is a set of values.

Our definition of an FGM is clearly an abstraction of stochastic modeling. We remark that it is possible to start from a definition that is yet more general and in a sense simpler in which only choice models are used and a variable number may be attached to each edge. But then connecting our results to the mainly stochastic problems we are interested in becomes more complex, and it is more difficult to see the connection between our generalization and the original literature.

We therefore develop Baum-Welch/EM for FGMs starting from a lemma that is stated so as to illustrate its correspondence with the EM literature. It is a statement about improper mixture densities, i.e. where the whole and its constituent parts are not necessarily probabilities. This is relevant to FGMs because they may be regarded as immense mixtures of contributions from every source-sink path. We follow earlier lines in its proof without requiring proper densities. The lemma may also be derived starting from theorem 2.1 of [BPSW70] in its most general form.

Lemma 1 *(Baum-Welch/EM) Let* $f(x|\Phi) \triangleq \sum_{i=1}^{k} g(x|\omega_i, \Psi) h(\omega_i|\Upsilon)$ *be a finite parameterized mixture where* $\{g(\cdot|\omega_i)\}$ *and* h *are nonnegative parameterized functionals (but not necessarily probability functions or pdfs),* ω_i *selects a component of the mixture, and* $\Phi \triangleq \{\Psi, \Upsilon\}$*. Also define probability* $P(\omega_i|x, \Phi) \triangleq g(x|\omega_i, \Psi) h(\omega_i|\Upsilon) / f(x|\Phi)$*. Finally, assume* $f(x|\Phi) > 0$*. Then:*

$$\overline{\Phi} \in \text{argclimb}_{\Psi', \Upsilon' || \Psi, \Upsilon} \left(\sum_{i=1}^{k} P(\omega_i|x, \Phi) \log g(x|\omega_i, \Psi') + \sum_{i=1}^{k} P(\omega_i|x, \Phi) \log h(\omega_i|\Upsilon') \right)$$

is an improved parameter set. That is, $f(x|\overline{\Phi}) \geq f(x|\Phi)$*, with the inequality strict unless* Φ *is already optimal. It is understood that* Ψ', Υ' *vary over their defined domain.*

proof: We begin by establishing the identity:

$$\begin{aligned} E_{\omega|x,\Phi}(\log f(x|\Phi')) &= \sum_{i=1}^{k} P(\omega_i|x,\Phi)\log f(x|\Phi') \\ &= \log f(x|\Phi')\sum_{i=1}^{k} P(\omega_i|x,\Phi) \\ &= \log f(x|\Phi') \end{aligned}$$

where ω is viewed as a discrete random variable distributed as $P(\omega_i|x,\Phi)$ and $E_{\omega|x,\Phi}$ denotes expectation with respect to it. The point is merely that $f(x|\Phi')$ is independent of ω since the former depends on Φ' not Φ.

Next with $f(x,\omega_i|\Phi') \triangleq g(x|\omega_i,\Psi')h(\omega_i|\Upsilon')$ we have $\log f(x|\Phi') = \log f(x,\omega|\Phi') - \log P(\omega|x,\Phi')$ since $f(x,\omega|\Phi') = P(\omega|x,\Phi')f(x|\Phi')$. So:

$$\begin{aligned} \log f(x|\Phi') &= E_{\omega|x,\Phi}\log f(x|\Phi') \\ &= E_{\omega|x,\Phi}\log f(x,\omega|\Phi') - E_{\omega|x,\Phi}\log P(\omega|x,\Phi') \\ &= \sum_{i=1}^{k} P(\omega_i|x,\Phi)\log f(x,\omega_i|\Phi') - \sum_{i=1}^{k} P(\omega_i|x,\Phi)\log P(\omega_i|x,\Phi') \end{aligned}$$

It follows from Jensen's inequality that the second summation is minimized by $\Phi = \Phi'$. This can also be seen by recognizing it as $-[D(\Phi\|\Phi') + H(\Phi)]$, where $D(\cdot\|\cdot)$ is the Kullbak-Leibler distance, and $H(\cdot)$ denotes entropy (see [CT91]). Thus maximizing the first term, written $Q(\Phi,\Phi')$ in the literature, will surely increase $\log f(x|\Phi')$ and hence $f(x|\Phi')$. But:

$$\sum_{i=1}^{k} P(\omega_i|x,\Phi)\log f(x,\omega_i|\Phi') = \sum_{i=1}^{k} P(\omega_i|x,\Phi)\log g(x|\omega_i,\Psi') + \sum_{i=1}^{k} P(\omega_i|x,\Phi)\log h(\omega_i|\Upsilon')$$

and the right side is recognized as the object of the argclimb operator in the lemma's statement. $\square$

The conditional expectation operator $E_{\omega|x,\Phi}$ can cause confusion but we use it to make clear the correspondence with the EM literature. Our discussion in section 2.1

starting with Eq. 1 represents an alternative approach that is free of explicit mention of expectation and probability.

To apply lemma 1 to FGM optimization we introduce two more simple algorithms. The first computes $F(x|\Phi)$ by sweeping through the DAG's vertices in reverse rather than forward order, establishing the values of a new set of real variables $\beta_1, \ldots, \beta_n$:

Algorithm 2

procedure *backward*

$\beta_n = 1$

for $i = n-1, \ldots, 1$

$\beta_i = \sum_{e \in \mathcal{O}(v_i)} \beta_{d(e)} \cdot C(e|\Upsilon) \cdot M(e, x|\Psi)$

This corresponds to the backward step of the forward-backward algorithm. After it has run, β_1 equals $F(x|\Phi)$, and therefore α_n.

We denote by $\omega_1, \ldots, \omega_N$ the source-sink paths through F, and by $E(\omega)$ the set of edges along a particular path ω, and by $\Omega(e)$ the set of paths that include edge e. The contribution of a single path ω to the overall value of the FGM is denoted $F(x, \omega|\Phi)$ and given by:

$$F(x, \omega|\Phi) = \prod_{e \in E(\omega)} C(e|\Upsilon) \cdot M(e, x|\Psi)$$

So that:

$$F(x|\Phi) = \sum_{i=1}^{N} F(x, \omega_i|\Phi)$$

Next define $\gamma_\omega \triangleq F(x, \omega|\Phi)/F(x|\Phi)$, and then for each edge e define:

$$\gamma_e \triangleq \sum_{\omega \in \Omega(e)} \gamma_\omega$$

Notice that by definition the γ_e arise from a partition of the set of all source-sink paths whence it is clear that:

Proposition 1 *The sum of the γ_e for any cut of the DAG is unity.*

The α and β variables may be used to compute these γ_e via a third simple dynamic program corresponding to the *expectation step* of EM:

Algorithm 3

procedure *gamma*
  for $i = n-1, \ldots, 1$
    for $e \in \mathcal{O}(v_i)$
      $\gamma_e = (\alpha_{s(e)} \cdot C(e|\Upsilon) \cdot M(e,x|\Psi) \cdot \beta_{d(e)})/F(x|\Phi)$

where either α_n or β_1 may be substituted for $F(x|\Phi)$. As a practical matter algorithms 3 and 2 may be combined.

We are now in position to state precisely how the objective of optimizing an FGM is decomposed into smaller problems.

Definition 2 *An* <u>*inexact Baum-Welch/EM reestimate*</u> $\overline{\Phi} = (\overline{\Psi}, \overline{\Upsilon})$ *of the current parameters* $\Phi = (\Psi, \Upsilon)$ *of an FGM* F *given an observation* x *is formed by solving two sets of independent optimization problems:*

1. *For each* i *the improved parameters* $\overline{\Psi}_i$ *of observation model* M_i *are given by:*

$$\overline{\Psi}_i \in \text{argclimb}_{\Psi'_i \| \Psi_i} \sum_{e \in m^{-1}(i)} \gamma_e \cdot \log M(e, x|\Psi'_i)$$

2. *For each* i *the improved parameters* $\overline{\Upsilon}_i$ *of choice model* C_i *are given by:*

$$\overline{\Upsilon}_i \in \text{argclimb}_{\Upsilon'_i \| \Upsilon_i} \sum_{e \in c_1^{-1}(i)} \gamma_e \cdot \log C(e|\Upsilon'_i)$$

where the γ_e *are computed based on* $\Phi = (\Psi, \Upsilon)$*. The* <u>*exact Baum-Welch/EM reestimate*</u> *is formed by replacing the* argclimb *operations with* argmax*. By* $m^{-1}(\cdot)$ *and* $c^{-1}(\cdot)$ *we mean the set valued inverses of functions* m *and* c*.*

The *exact* form corresponds to the historical definition of the Baum-Welch/EM reestimate. However, since this chapter's focus is on decomposition, and not mathematical issues relating to particular choice and observation models, we will use "Baum-Welch reestimate" to refer to our *inexact* form defined above. This allows our framework to include models for which exact solution is not convenient. Also it is interesting to note that one may choose to not reestimate one or more models, and still improve the overall FGM.

Theorem 1 *The Baum-Welch/EM reestimate $\overline{\Phi}$ satisfies $F(x|\overline{\Phi}) \geq F(x|\Phi)$ with the inequality strict provided that progress is made in at least one subsidiary problem.*

Remark: this theorem's message is one of decomposition. That is, that the subsidiary problems may be considered in isolation, and any progress made will strictly contribute to progress in the overall problem.

proof: We write:

$$F(x|\Phi) = \sum_{i=1}^{N} \underbrace{\left(\prod_{e \in E(\omega_i)} M(e, x|\Psi_i) \right)}_{g(x|\omega_i, \Psi)} \cdot \underbrace{\left(\prod_{e \in E(\omega_i)} C(e|\Upsilon_i) \right)}_{h(\omega_i|\Upsilon)}$$

which makes clear the correspondence between the FGM and lemma 1. Now by their definitions $\gamma_{\omega_i} = P(\omega_i|x, \Phi)$ so from the lemma we have:

$$\overline{\Phi} \in \operatorname{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E(\omega_i)} M(e, x|\Psi'_i) \right) + \sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E(\omega_i)} C(e|\Upsilon'_i) \right) \right]$$

$$\overline{\Phi} \in \operatorname{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{N} \sum_{e \in E(\omega_i)} \gamma_{\omega_i} \log M(e, x|\Psi'_i) + \sum_{i=1}^{N} \sum_{e \in E(\omega_i)} \gamma_{\omega_i} \log C(e|\Upsilon'_i) \right]$$

The double summations above enumerate every edge along every path though the FGM. To complete the proof we have only to enumerate them in a different order:

$$\overline{\Phi} \in \operatorname{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{|M|} \sum_{e \in m^{-1}(i)} \sum_{\omega \in \Omega(e)} \gamma_\omega \log M(e, x|\Psi'_i) + \sum_{i=1}^{|C|} \sum_{e \in c_1^{-1}(i)} \sum_{\omega \in \Omega(e)} \gamma_\omega \log C(e|\Upsilon'_i) \right]$$

Recall $\gamma_e = \sum_{\omega \in \Omega_e} \gamma_\omega$ so:

$$\overline{\Phi} \in \mathrm{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{|M|} \left(\sum_{e \in m^{-1}(M_i)} \gamma_e \log M(e, x|\Psi'_i) \right) + \sum_{i=1}^{|C|} \left(\sum_{e \in c_1^{-1}(i)} \gamma_e \log C(e|\Upsilon'_i) \right) \right]$$

and the parenthesized terms are the independent subproblems of definition 2. $\Box$

We now change our focus to computational and data structural issues. The first part of our discussion deals with the intricacies of recursion. That is, when an FGM invokes another FGM as one of its observation models. Here, proper algorithm and data structure design can lead to polynomial rather than exponential complexity as illustrated by the case of stochastic context free grammars described in section 2.6.2. We will confine ourselves to finite recursion. The second issue has to do with the arithmetic demands of FGM computation.

The idea behind reducing the complexity of recursive FGM operations is a simple one: avoid duplicate work. If a particular combination of an observation model and selection function occurs multiple times, the combination should only be evaluated once. Since choice models do not depend on the observation, each should be evaluated only once. The computation is then ordered so that when an edge is considered, its choice and observation functions have already been evaluated and their values may be looked up in constant time.

To formalize this idea recall that $M(e, x|\Psi)$ is defined as $M_{m(e)}(S_{\varsigma(e)}(x)|\Psi_{m(e)})$. Notice that edge e appears as a function argument. We say $M(e_1, x|\Psi) \equiv M(e_2, x|\Psi)$ if their definitions are the same. Each equivalence class is referred to as a *model application* and expressing algorithms in terms of these classes avoids duplicate work. If for two edges $\varsigma(e_1) \neq \varsigma(e_2)$ then by this definition $M(e_1, x|\Psi) \not\equiv M(e_2, x|\Psi)$ – even if $S_{\varsigma(e_1)}$ is the same function as $S_{\varsigma(e_2)}$. For this reason we adjust the definition to account for equivalences that may exist among the $\{S_i\}$.

This adjustment is subtle and we therefore illustrate it by example. Suppose that the observation model associated with an edge e_i of the top level FGM is itself an FGM F_a and that selection function S_q is used by e_i. Now within F_a assume that some edge e_j uses an observation model m with selection function S_r. Then to evaluate m, selection functions S_q and S_r are composed. Now focus on another top level edge

e_k to which an FGM F_b is attached using selection function S_t. Within F_b consider an edge e_ℓ using the same observation model m as does e_j, but in combination with selection function S_v. To evaluate m here S_t and S_v are composed. So m should only be evaluated once if $S_r(S_q(\cdot))$ and $S_v(S_t(\cdot))$ are the same function. This may be the case despite the fact that S_r and S_v are not. Thus, whenever possible, equivalence classes should be defined so as to account for possibly equivalent compositions of selection functions. This is easily implemented for the class of *projection* selection functions introduced in the next section.

Mathematically definition 1 states that each edge has an attached observation model via mapping m. Computationally the edge instead selects (in practice points to) a model application structure which identifies the observation model and the corresponding selection function. We denote the set of observation model applications by $\{\mathcal{M}_i\}$. Inclusion within the recursive hierarchy induces a partial ordering on this set where it is important to realize that a given model application may be referenced at more than one recursive level. We extend this ordering by specifying that all primitive observation models are dominated by any FGM. In what follows we will then assume that $\{\mathcal{M}_i\}$ is indexed in topological order so that the top-level FGM is first and the primitive models form the list's end.

We also introduce *choice model application structures* $\{\mathcal{C}_i\}$. Since choice models do not depend on x or involve a selection function, these structures are in 1:1 correspondence with the choice models themselves and are identically indexed. Many references may exist to a choice model, and its corresponding application structure avoids redundant model evaluation by recording the model's current value. It also accumulates γ contributions during expectation step processing so that a single call may be made to maximize the model. This is explained later in greater detail.

If an FGM F_c is a recursive child of a parent F_p, one alternative is to eliminate the recursion and expand the child *inline* within the parent by adding vertices and edges to its DAG. If F_c is invoked along edge e between vertices i and j then the first step of this inline expansion disconnects e from j and replaces the invocation of F_c with a void model. The child F_c is then inserted between the disconnected end and vertex j.

When evaluating F_p this inline expansion clearly isn't necessary since one can evaluate F_c in *isolation*, record its value in the model application structure, and in constant time refer to that value where needed in F_p. The same is true for parameter reestimation although a brief argument is needed.

Proposition 2 *Suppose edge e of FGM F_p invokes model F_c, also an FGM. Then if F_c is expanded inline within F_p, the γ values of its edges are exactly those computed in isolation multiplied by γ_e.*

proof: Let e connect vertices i and j and let ω denote some source-sink path of F_c and γ_ω its weight. If F_c is expanded inline the value of ω will be $\alpha_i C(e|\Upsilon)\gamma_\omega \beta_i / F_p(x|\Phi)$. In isolation it is $\gamma_\omega / F_c(x|\Phi)$. But $\gamma_e = \alpha_i C(e|\Upsilon) F_c(x|\Phi) \beta_i / F_p(x|\Phi)$ from which the proposition follows. $\Box$

So if the same model application $\mathcal{M}_i$ is pointed to by many FGM edges, the right computational approach is to first accumulate their γ values and then to process $\mathcal{M}_i$. Here we emphasize that this approach is more efficient *and* mathematically identical to inline expansion. So if at the bottom of the recursion the required argmax problems are solved, then at all higher levels they are too.

An observation model M_i that is not an FGM is said to be *primitive*, and is represented by an abstract type for which the following operations are defined:

1. $M_i : evaluate(z)$
2. $M_i : Estep(z, \Gamma)$
3. $M_i : Mstep()$

Here z is an element of the range of a selection function and Γ is a non-negative weight propagated down as described in proposition 2. The *evaluate* function returns the value of M_i at z. Notice that the model's parameters Ψ_i are hidden by the abstract type. In practice a mutator may be provided to set them and a selector provided for readout. The names *Estep* and *Mstep* follow the EM literature even though FGMs are not restricted to maximum-likelihood estimation of probability functions and densities.

Given a sequence of values $z_1, \ldots, z_m$, and nonnegative *multiplicities* $\Gamma_1, \ldots, \Gamma_m$ the *Estep* and *Mstep* procedures address the following optimization problem:

$$\overline{\Psi}_i = \mathrm{argmax}_{\Psi'_i} \prod_{j=1}^{m} M_i^{\Gamma_j}(z_j|\Psi'_i) \tag{6}$$

which generalizes Eq. 5. The *Estep* procedure is called for each (z_i, Γ_i) and then *Mstep* is called to reestimate the model's parameters. For some models the result is the unique optimum parameter set, but in general the process is repeated. For example, if the z_i are letters of the alphabet, the Γ_i are positive *frequencies*, and the model's value is a probability for each letter, then *Estep* need only record the Γ values provided and *Mstep* can easily determine the unique optimal model. That is:

$$P(i) = \frac{\Gamma_i}{\sum_j \Gamma_j}$$

We point out that if the probability of symbol i is zero prior to reestimation, then γ_i and therefore its reestimate $P(i)$ are zero. Such parameters in this simple discrete model are then effectively disabled since their corresponding choice of alphabet symbol will always have probability zero.

A simple unique solution also exists for the normal density and others. In the discrete example above the model is constrained to produce values which sum to one over the alphabet. Normal densities are constrained to integrate to one. These constraints are obviously important so that the overall optimization problem has a bounded solution. But it is important to observe that their precise nature is irrelevant and invisible to the FGM framework. In particular there is no need for a model to represent a probability function There is also no need to restrict one's attention to models for which unique optimum solutions are readily available. As noted earlier, any progress made for even one model, will translate to progress in the overall FGM optimization.

A choice C_i is represented by an abstract type for which the following operations are defined:

1. $C_i : evaluate(j)$

2. $C_i : Mstep(\Gamma[\ldots])$

Here j is the index of an edge and the *evaluate* function returns its value under C_i given the model's current parameters Υ_i. The *Mstep* procedure is passed the vector $(\Gamma_1, \ldots, \Gamma_{|C_i|})$ and addresses the following optimization problem:

$$\overline{\Upsilon}_i = \text{argmax}_{\Upsilon'_i} \prod_{j=1}^{|C_i|} C_i^{\Gamma_j}(j|\Upsilon'_i) \tag{7}$$

If C_i is a probability function then the problem is exactly that of Eq. 6 and an exact solution is available. In both cases the solution is mathematically optimal but other problem constraints might be incorporated into C_i or M_i that sometimes forbid it. For example, one might prohibit probabilities from falling below some fixed threshold. This amounts to a change in parameterization and we will see in a later section that this capability may be used to alter the meaning of *optimal* in useful ways. Note that we might instead have treated a choice model as an observation model over the natural numbers but decided that their roles are sufficiently distinct to warrant separate designs.

We now return to our discussion of model applications and begin by describing the information that must be associated with these structures. An observation application structure $\mathcal{M}_i$ contains:

1. $\mathcal{M}_i : model$ — the integer index of the observation model associated with this application.

2. $\mathcal{M}_i : selector$ — the integer index of associated selector function.

3. $\mathcal{M}_i : value$ — a real number representing the most recent model application evaluation.

4. $\mathcal{M}_i : \Gamma$ — an accumulator to which values are added by the possibly many places within the overall model that refer to $\mathcal{M}_i$. In the case of primitive models we may then call *Estep* a single time. For FGMs the value corresponds to γ_e of proposition 2 and is used to propagate γ values down the recursive hierarchy.

A choice application structure $\mathcal{C}_i$ contains:

1. $\mathcal{C}_i : value[\ldots]$ — a vector of real numbers representing the choice model's value at each integer of its domain based on its current parameters.

2. $\mathcal{C}_i : \Gamma[\ldots]$ — a vector accumulator to which values are added by the possibly many places within the overall model that refer to $\mathcal{C}_i$.

No *model* variable is needed since indexing corresponds to that of the choice models themselves.

Recall that the mathematical definition 1 of an FGM associates edges directly with elements of M, C, S via functions m, c, s – there are no model application structures. While the FGM along with its associated model application structure lists might be generated incrementally, it is simpler to imagine that the FGM is first specified *mathematically* and then *compiled* into a form that includes these lists. This compiled form is denoted $\hat{F}$ and merges the M, C, S sets of each FGM into single densely enumerated sets. In compiled form FGM edges do not point directly to observation models but rather to observation model application structures. The m function is replaced by m_a to effect this change.

We present evaluation, expectation step, and maximization step algorithms for compiled FGMs. These are not recursive functions but rather operate by traversing the global model application lists built during compilation.

Before any of these are called the FGM must be initialized. This procedure *initialize*$(\hat{F})$ consists of setting the Γ variables to zero in all choice model application structures and then evaluating each choice model and recording its value there.

Evaluating an FGM and performing an expectation step are carried out by bottom-up (reverse order) and top-down (in order) traversals respectively of the model application lists. Evaluation returns a single value for $\hat{F}$ given observation x and also records the value of all subsidiary models within the observation model application structures.

Algorithm 4

function $evaluate(\hat{F}, x)$
    for each $\mathcal{M}_i$ in descending index order
        if "primitive"
            $j = \mathcal{M}_i : selector$
            $k = \mathcal{M}_i : model$
            $\mathcal{M}_i : value = M_k : evaluate(S_j(x))$
        else "an FGM"
            $\mathcal{M}_i : value =$ result of *algorithm 1*
    return $\mathcal{M}_1 : value$

where algorithm 1 refers to the appropriate observation and choice application structure value variables rather than computing $C(e|\Upsilon)$ and $M(e, x|\Psi)$.

As for primitive observation models the FGM expectation step procedure accepts a Γ argument. Supplying a value of 1 for each x observed corresponds to the optimization problem of Eq. 5. In general these Γ values generalize the problem by appearing as exponents to the FGM's value as in *Eq.* 6. The FGM is first evaluated so that the observation model application structures contain current values. Their Γ values are then set to zero except for the first which corresponds to the top-level FGM and is set to the function's Γ argument. The main loop proceeds top-down to propagate Γ contributions to lower observation model application structures. Eventually primitive structures are reached and a primitive expectation step is performed. The FGM's value is returned as a convenience since it was necessary to compute it.

Algorithm 5

function $Estep(\hat{F}, x, \Gamma)$

$evaluate(\hat{F}, x)$

for each $\mathcal{M}_i$

if $i = 1$, $\mathcal{M}_i : \Gamma = \Gamma$, else $\mathcal{M}_i : \Gamma = 0$

for each $\mathcal{M}_i$ in increasing index order

if "primitive"

$j = \mathcal{M}_i : selector$

$k = \mathcal{M}_i : model$

$M_k : estep(S_j(x), \mathcal{M}_i : \Gamma)$

else "an FGM"

Set $\{\gamma_e\}$ using algorithms 1,2,3

for each edge e

$\mathcal{M}_{m_a(e)} : \Gamma += \gamma_e \cdot \mathcal{M}_i : \Gamma$

$\mathcal{C}_{c_1(e)} : \Gamma[c_2(e)] += \gamma_e \cdot \mathcal{M}_i : \Gamma$

return $\mathcal{M}_1 : value$

The FGM maximization procedure maximizes the primitive observation models and all choice models. The FGM is then reinitialized.

Algorithm 6

function $Mstep(\hat{F})$

for each M_i

$M_i : maximize()$

for each C_i

$C_i : maximize(\mathcal{C}_i : \Gamma[\ldots])$

$initialize(\hat{F})$

Observe that we do not specify that the models are maximized in any particular order since if their parameter sets are disjoint so is the act of maximizing them.

Our notation thus-far has suggested that the parameter sets are in fact independent but in general they need not be. For example, two models might share the same parameters but compute two different functionals depending on them. In this case the two corresponding optimization subproblems of definition 2 are simply combined into one. From an implementation viewpoint this kind of situation can be handled entirely at the model type level through communication between them that is invisible at higher levels. So the fact that algorithm 6 specifies no order for maximization does not imply that we are limited to the case of disjoint parameter sets.

Analysis of the algorithms above is straightforward from the structure of the compiled FGM $\hat{F}$. Let N_E denote the sum over nonprimitive observation model applications, of the number of edges in the FGM corresponding to each. Then both evaluation and expectation step processing require $O(N_E) + T_p$ time where T_p denotes the time needed to perform the required primitive observation model type operations. These are described by the final entries in the list of observation model applications. When these primitive operations are $O(1)$ time the overall complexity is just $O(N_E)$ since their number cannot exceed N_E. Each choice model is evaluated once during *initialize*. The *maximize* operation time-complexity is just that required to perform a maximization step for each observation and choice model in the system. Ignoring the internal requirements of the observation and choice models, it is obvious that $\hat{F}$ requires $O(N_E)$ space.

The algorithms above are conceptually straightforward, but an important numerical issue must be faced in their implementation. As one sweeps through the DAG, the α and β variables can easily become too large or small to represent as conventional floating point values. This is true even of the final FGM value. If the FGM is a probability model, exponent-range underflow is the issue and corresponds to an extremely small probability of generating any particular observation. For complex problems this behavior is the rule not the exception since element probabilities in general decline exponentially with dimension. Logarithmic representation is one solution but the authors have also explored a floating point representation with extended exponent range as reported in [RY94]. Because γ variables are normalized by $F(x)$

they may be adequately represented using standard floating point.

We now briefly discuss the computational complexity of FGM maximization. A restricted class of FGMs is identified for which maximization is shown to be NP-complete. Members of this class have constant choice functions that assume value 1. There are two selection functions s_0 and s_1 that assume constant values 0 and 1 respectively. Notice that there is no dependency on any observation. Each observation model has a single parameter bit b and given boolean argument x assumes value $(b \wedge \bar{x}) \vee (\bar{b} \wedge x)$. The FGM then assumes a bounded integral value and the maximization problem is well-posed. The corresponding decision problem is: does there exist a set of parameter bits such that the FGM's value is greater than a given integer i? This problem is clearly in NP since given a set of parameter bits, FGM evaluation, which runs in linear time, may be used to answer the question. We will refer to this setting as the LFGM (logical FGM) problem.

Theorem 2 *LFGM is NP-complete.*

proof: The NP-complete SAT problem is easily reduced to LFGM by first associating an observation model with each variable. If the variable $v, \bar{v}$ occurs in a clause then selector s_1 or s_0 is used respectively. Clauses gives rise to trivial FGMs in which each term corresponds to a distinct edge from source to sink with the appropriate observation model and selection function attached. The FGMs for each clause are then concatenated to form the final FGM. The set of clauses are then satisfied if and only if the FGM assumes a value greater than zero. The reduction's complexity is linear whence LFGM is NP-complete. □

It is possible to construct reductions like that above where the observation models are continuous not discrete functions; but further discussion of the complexity of continuous optimization is beyond the scope of this chapter. These continuous formulations correspond to stochastic *continuous embedding* approaches to SAT and represent an interesting area for future work.

This concludes our development of FGMs as general mixture-forms and in closing we remark that our framework is certainly open to additional generalization and refinement. For example one might introduce choice models that also depend on the observation (discussed later), or replace the m, c, s edge association functions with distributions. Another approach is to dispense with choice models altogether since they may be emulated using observation models. Many such variations are possible but our goal was a development somewhere between empty generalization and over specialization.

2.3 FGM-based Probability Models

In this section we consider specializations of Finite Growth Models such that their value represents a probability. The associated DAG and attached models may then be viewed as an acyclic automaton directing the stochastic generation of objects. Stochastic modeling is the original motivation for FGMs and much of our nomenclature is drawn from this viewpoint.

The first step is to specialize the selection functions of definition 1 to projections. That is, functions that select some subset of the dimensions of tuple observation x. A projection is characterized by a subset g of the dimension indices of $\mathcal{X}$. We associate some subset g_e with every edge e of the FGM and then denote by $\pi_{g_e}(x)$, the *projection* of observation tuple x corresponding to selection of the components in g_e. This then is the selection function associated with e. For some edge e suppose $g_e = \{2, 4\}$. Then $\pi_{g_e}(x)$ is a tuple with two components which are equal in value to the second and fourth components of x. We assume for now that $g_e \neq \emptyset$ but will later see that this is unnecessary. A composition of projections, applied to x, is again a projection characterized by some dimensional subset; whence it is straightforward to define equivalence for the purpose of identifying truly distinct model applications for recursively combined FGMs. The use of projections corresponds to the generation of observations in arbitrary groupings and orderings – a fact we will rely on later in our discussion of stochastic transduction.

Next we assume that all choice and observation models are probability functions or densities. Each C_i then corresponds to a stochastic *choice* among edges to follow.

Each M_i corresponds to stochastic *generation* of some portion of x (since our selectors are projections).

Finally we specialize the g_e subsets so that along any source-sink path their intersection is void and their union is the complete set of dimensions. Each path then represents one way in which x might be stochastically generated or *grown* and corresponds to a permutation of the dimensional indices $1, \ldots, d$. The "finite growth model" framework is so named because we imagine that complete observations are grown, starting from the empty set, in many possible orderings, as specified by the model's underlying graph. The value of each source-sink path through the FGM is the probability of following that path *and* generating x. The FGM's value is then a probability function $P(x|\Phi)$ on $\mathcal{X}$, i.e. its sum/integral is one. When it is necessary to make clear that we are referring to an FGM with the restrictions above, we will use the term *stochastic finite growth model* (SFGM).

After algorithm 1 has run, α_n gives $P(x|\Phi)$. However the other α values have meaning as well. In general α_i is the probability of arriving at v_i *and* observing that part of x corresponding to the paths between v_i and the source. Note that because each source-sink path corresponds to a permutation of the observation tuple, every path from the source to v_i must generate the same observation components — possibly however in different orders. Dually, α_i is the probability that random operation of the machine encounters v_i, while at the same time generating those observation components accounted for by the source to v_i paths. Algorithm 1 sums over all paths through the FGM but is easily modified to find instead a single optimal path. This is referred to as the *Viterbii decode* of the FGM and answers the question: what is the single most likely explanation for the observation?

Algorithm 2 also computes $P(x|\Phi)$ as β_1. In general β_i is the probability of observing that part of x corresponding to the paths between v_i and the sink, given passage through v_i. Dually, β_i is the probability that the machine passes through v_i and proceeds on to generate those observation components accounted for by the v_i to sink paths. Denoting by L the part of x corresponding to paths from the source to v_i, and by R the part of x corresponding to paths from v_i to the sink, $\beta_i = P(R|v_i)$ and

$\alpha_i = P(L, v_i)$. But $P(R|v_i) = P(R|L, v_i)$ because an SFGM is a Markov process. So $\alpha_i \cdot \beta_i = P(L, v_i, R)$, i.e. the probability of generating x and passing through v_i.

Observe that the meaning of β_i is not simply a time-reversed interpretation of α_i. This is because an SFGM has an intrinsic directionality. In-bound DAG edge probabilities need not sum to one so simply reversing edge direction does not leave a well-formed SFGM.

For SFGMs the γ_e values computed by algorithm 3 are the probabilities $P(e|x) = P(e, x)/P(x)$; where 'e' refers to the event of a transition over edge e. That is, γ_e gives the *a posteriori* probability of following edge e conditioned on observation (generation) of x. Given an SFGM and any x, the γ_e values must satisfy the following constraint which can in practice be used as an implementation self-check. It is a specialization of proposition 1 to SFGMs:

Proposition 3 *Given a stochastic finite growth model, an observation with nonzero probability, and γ values computed by algorithms 1, 2, and 3, then:*

1. *If $|g_e| = 1, \forall e$, then $\sum_e \gamma_e = d$.*
2. *In general, $\sum_e \gamma_e \cdot |g_e| = d$.*

proof: The key idea is that the edges that observe a given observation dimension induce a cut in the DAG but we present the proof from a probabilistic viewpoint. Assume $|g_e| = 1, \forall e$. Because every source sink path path generates each observation component exactly once, the set of edges E_j which generate a particular component j, corresponds to a set of mutually exclusive and exhaustive events. Hence $\sum_{e \in E_j} P(e|x) = \sum_{e \in E_j} \gamma_e = 1$. There are d observation components, and these partition E into disjoint sets $\{E_i\}$. So $\sum_{e \in E} \gamma_e = d$, establishing the first part of the proposition. In general, the $\{E_i\}$ are not disjoint, and the number of sets to which an edge e belongs is $|g_e|$. Multiplying each γ_e by $|g_e|$ in effect gives each set its own copy, so that the sum remains d. $\square$

We required above that $g_e \neq \emptyset$. If necessary this restriction may be effectively removed and one may include so called *nonemitting* edges in a model. To do so the observation tuple is augmented with *dummy* components each capable of assuming only a single value. Nonemitting edges then observe these components assigning them probability one. The result is a probability model on the augmented observation space, and since its extension was trivial, on the original space.

The m and c functions of our FGM framework may be used to implement the *parameter tying* concept from the stochastic modeling literature. When $m(e_1) = m(e_2)$ the observation models attached to these edges are said to be tied. If the edges leaving vertex v_1 map under c to the same choice model as those leaving another vertex v_2, the choice models at v_1 and v_2 are said to be tied. In crafting a model there are two reasons for parameter tying. The first is that it reduces the number of free parameters in the model and as such combats overtraining. The second is that problem domain knowledge may suggest that two models represent the same underlying phenomenon.

Equation 5 formalizes the problem of parameter optimization given a training set $x_1, \ldots, x_m$ by forming a cascade of identical FGMs. More generally we may write:

$$\overline{\Phi} = \mathrm{argmax}_{\Phi} \prod_{i=1}^{m} F_i(x_i|\Phi) \tag{8}$$

where now a possibly different FGM is associated with each training observation but all are tied to a single underlying collection of choice and observation models. This is an important conceptual point since it represents the technique used to deal with training examples of varying dimension. For example, each x_i might represent a finite time series. Typically the FGMs are instances of a single design – varied in order to accommodate the structure of the observation tuple. Equivalently the entire cascade may be regarded as a single FGM operating on a single observation formed by concatenation of the entire training set. This cascading and concatenation is of conceptual value only. In practice one merely performs expectation steps for each pair (x_i, F_i) followed by a maximization step. There is no need to actually form the cascade and concatenation. In an SFGM setting Eq. 8 represents the training set's *likelihood* and our goal of maximizing it corresponds to *maximum-likelihood* parameter estimation of statistics.

Ultimately one is faced with the task of maximizing primitive models. For simple discrete models and many continuous ones (notably normal densities) each independent maximization problem arising from theorem 1 has a unique globally optimal solution and satisfies certain other conditions. Lemma 1 and theorem 1 may then be strengthened to say that strict progress is made unless one is already at a critical point in parameter space. Even this does not imply parametric convergence however. See [BPSW70, DLR77, RW84] for discussion of the convergence and other properties of the EM algorithm. These mathematical issues including rate of convergence, while important, are not our focus and we will not consider them further. Clearly relating various conditions on primitive models to the resulting behavior of the overall FGM represents an important area for future work.

We saw in the last section that simple discrete models are easily maximized. If M_i is a multivariate normal density then maximization is also easily accomplished since the weighted sample mean and weighted sample covariance define the maximum-likelihood parameter set. The weights are γ_e/Γ where Γ denotes the sum of all γ_e associated with M_i. In general, any density for which a maximum-likelihood estimate is readily available may be used as an observation model, e.g. beta densities for random variables confined to $[0, 1]$. See [HAJ90] for a treatment of HMMs including a compact discussion of the discrete and continuous optimization problems discussed above.

2.4 Beyond Finite Probability Models

SFGMs as defined in the previous section are probability functions on an associated finite dimensional observation space. They arise by restricting the FGM framework in several ways. We now relax certain of these restrictions and show that FGMs may be applied to a much broader class of stochastic modeling problems and settings.

2.4.1 Truncated Stochastic Processes

In contrast to an SFGM, *stochastic models* are frequently defined as either infinite generative processes, or processes that generate finite objects of unbounded size. Time series methods such as hidden Markov models are an example of the first kind and stochastic context free grammars are an example of the second. In the first case the probability of a finite object refers to the aggregated probability of all infinite objects that include it – in the case of strings the combined probability of all infinite strings with a particular finite prefix.

Processes such as these have an associated infinitely deep finite branching structure which is conceptually *truncated* to a finite DAG that explains observations of some fixed length. This DAG is regarded as part of an FGM whose primitive parameterized models correspond to those of the original process. Given a set of finite observations, a set of corresponding FGMs is constructed as discussed in earlier sections. In this way the parameters of an infinite process that best explain a set of finite observations may be learned using FGMs.

This truncation is conveniently effected within the SFGM framework through the introduction of *null observation models*. These are simply functionals that assume the value zero everywhere and as such lie trivially within the FGM framework. When generation via the original infinite branching process would result in an observation component beyond the finite set of interest, a void model is attached to that edge and it is redirected to the FGM's sink. In this way the processes infinite graph becomes a finite DAG with a single source and sink. This is illustrated by our discussion of stochastic transducers and hidden Markov models in later sections.

A more subtle issue is addressed by the introduction of *void choice models* which assume value 1 for each outgoing edge. They are useful when the events associated with future generation may be partitioned into mutually exclusive sets. Our discussion of stochastic context free grammars in a later section provides illustration.

2.4.2 Alternative Maximization Criteria

Maximum likelihood (ML) parameter estimation is a *best-fit* strategy and is therefore susceptible to overtraining. For example, if a discrete model never observes a given symbol, its ML parameters assign the symbol probability zero. The result can be poor generalization. Also, our discussion thus far has tacitly assumed that a single stochastic model represented as an FGM is used to explain a set of observations. Here too if this model is very complex in comparison to the available training data, generalization will suffer. The *model selection problem* consists of somehow choosing or forming a model of appropriate complexity.

These two issues are highly related and we will now briefly describe how both may addressed within the FGM framework. The result is a conceptually clear and computationally tractable approach to implementing more sophisticated estimation schemes for both existing and new stochastic models. That EM could accomplish this in general was observed in [DLR77] but these penalized likelihood functions do not appear to have significantly impacted the use of EM in practice. We consider two related approaches: maximum *a posteriori* probability (MAP) estimation, and minimum description length (MDL). Our discussion begins with MAP.

Earlier sections have shown that Baum-Welch/EM may be used to address the problem of finding Φ that maximizes $F(x|\Phi)$. The form of this objective corresponds to maximum-likelihood estimation but we will see that it actually not so limited. Given a prior $P(\Phi)$ on model parameters the MAP objective is to instead maximize $P(\Phi|x)$. From the Bayes' rule it follows that this is the same as maximizing $P(x|\Phi)\cdot P(\Phi)$ since this quantity is proportional to $P(\Phi|x)$, i.e. the denominator is fixed. We resolve Φ into its constituent parts $\{\Psi_i\}$ and $\{\Upsilon_i\}$ corresponding to the FGM's observation and choice models respectively, and define the overall prior based on independent parts:

$$P(\Phi) \triangleq \left(\prod_{i=1}^{|M|} P(\Psi_i)\right)\left(\prod_{i=1}^{|M|} P(\Upsilon_i)\right)$$

This product may itself be regarded as a linear FGM denoted F_P with null observation models, and 1-way choice models having values corresponding to each product term. Alternatively each term may be represented as an observation model that assumes a constant value for each setting of its parameters. The two FGMs F and F_P are then combined end-to-end, or in the case of multiple observations F_P is attached to the end of $F_1, \ldots, F_M$. The FGM F computes $P(x|\Phi)$ and F_P computes $P(\Phi)$ so that the cascade thus formed has value $P(x|\Phi)\cdot P(\Phi)$. Maximizing it will then effect MAP-guided learning.

The minimum description length principle states that one should choose a model and its parameters such that the combined cost of *describing* these followed by the encoded observations is minimized. The related notion of Kolmogorov complexity deals with the smallest such description and is generally not computationally tractable. By contrast MDL practitioners are content to fix some parameterized scheme of representation and optimize the parameters. This does not fully implement the MDL principle but captures the important tension that must exist between model and data complexity. The cost of describing the observations is $-\log_2 F(x|\Phi)$ bits where it is assumed that F is a probability function. The cost of describing the model's structure is upper bounded by choosing some reasonable representation and counting its bits. The cost of describing a real valued parameter may be upper bounded by choosing some number of quantization levels Q, assuming uniform quantization (typically), and describing $-\log 2Q$ bits per parameter. Many variations are possible. The final codelength is denoted $L(\Phi)$ and the total description length is $L(\Phi) + -\log_2 F(x|\Phi)$. As in the case of MAP above we assume that $L(\Phi)$ is formed by summing separate contributions from primitive models. As before these may be regarded as a linear FGM where the value of the attached models is just the corresponding codelength exponentiated. Notice that the the MDL and MAP constructions are essentially identical and for this reason we view MDL as a Bayesian approach in which the prior is not necessarily a probability model.

The important message is that the original maximization problem may be decomposed into separate primitive problems even when the guiding criterion is not maximum-likelihood. Any primitive model for which maximization may be conveniently performed subject to the selected criterion, may be used.

Since the basic FGM is fixed in the discussion above, the cost of describing its representation in the MDL framework is also fixed with respect to FGM maximization. However an important part of the MDL outlook is selecting a model of appropriate complexity. This tension is captured in the FGM framework by forming a finite mixture of models over a range of complexities. Each model is made up of a basic FGM augmented with an MDL extension as described above. The cost of describing the model's design now enters the optimization problem since it differs across mixture components. Rather than selecting a single model complexity, reestimation will learn a distribution on complexity. Each mixture component is the *a posteriori* probability of its corresponding complexity given the observations. Similarly the MAP setting above may be extended to include the learning of a distribution on model complexity. The model with maximum *a posteriori* probability may be selected if it is important to identify a single *best* model. Finally, in the case of MDL the pedantic implementor will account for the complexity of the initial mixture in the overall optimization.

The result is not MDL at all in its original sense since a single model and parameter set is not selected. It is more properly regarded as MAP with MDL-inspired priors.

Our discussion above started by placing all penalty terms together at the front of an FGM. In the model selection example, however, some were placed instead following an initial choice. This may be seen as a special case of a general strategy in which penalty terms are placed as deep as possible within the DAG. That is, just before their first use.

Our discussion has then established:

Theorem 3 *Let F be an SFGM or arise from the truncation of an infinite stochastic process. Also assume that the primitive models employed support MDL (or MAP) maximization. Then a new FGM F' may easily be constructed with at most twice the number of edges such that its maximization corresponds to MDL (or MAP) guided maximization of F.*

The practical significance of this theorem is Baum-Welch/EM may be used to efficiently implement estimation criteria more sophisticated than ML with which it is historically associated – and via FGMs it is clear how this can be done for a wide variety of models.

2.4.3 Alternative Discrete Models

The trivial discrete probability model on an alphabet of k members has $k-1$ nonnegative free parameters that sum to at most unity. Many other possibilities exist. For example, if alphabet symbols correspond to states in a truncated Poisson process then the Poisson probability function $P(x|\lambda)$ may be used. This is true of a choice model as well and has application in the modeling of event durations. In general we remark that the use of nontrivially parameterized choice functions represents an interesting direction that has received only limited attention.

2.4.4 Observation Context Conditioned Probability Models

In our SFGM framework each edge generates some subset of the observation tuple's components such that as one moves from source to sink each is generated once. The values generated depend only on the edge over which generation is transiting. This corresponds to the first-order Markov assumption of stochastic modeling. In particular the value cannot depend upon route followed to reach the edge. But dependence on observation components generated earlier along the path does not violate the Markov assumption. Formally the observation models are then conditional probability functions where the conditioning is on earlier observation components – *not* earlier edge sequences. Each source-sink path then corresponds to a causal chain of conditional probabilities so that the FGM's value is a probability.

Brown observes in his thesis [Bro87] that the HMM output independence assumption may be relaxed to allow generation of the current speech sample window to depend on the previous one. He then uses Baum-Welch reestimation without reproof. His focus is on minimizing the number of model parameters – and we suspect that it is only for this reason that he did not propose longer context dependencies.

Our contribution is then the formalization of this message in general form within the FGM framework. That is: the problem of optimizing FGMs that employ conditional observation models is reduced to corresponding primitive conditional maximization problems.

2.4.5 Noncausal Unnormalized Models

Given a $k \times k$ pixel real-valued image one may model each pixel position as a function of some surrounding context. If each model's value is a probability and the context-induced dependency graph has no cycles, then the individual pixel probabilities may be multiplied to result in a *causal* probability model. Causal models are commonly built by fixing a scanning sequence and choosing contexts with respect to the implied temporal ordering. Since there is frequently no single natural scanning sequence the causality restriction may limit model effectiveness by preventing all of a pixel's neighbors from contributing to its prediction.

If the causality restriction is discarded each pixel's model is strengthened but their product is no longer a probability. This corresponds to relaxing the requirement that the projection functions in an SFGM correspond to a permutation of the observation dimensions. Noncausal neighborhood systems have proven useful in image processing [CJ93].

Since reestimation must nevertheless climb we have the result that even a noncausal unnormalized models like that sketched above may be improved within the FGM framework.

2.4.6 Dynamic Choice Function

As remarked earlier we might have allowed choice functions to depend on the observation x. Given choices $c_1, \ldots, c_k$ the FGM may then follow each with probability $p(c_i|x)$. In contrast with the conventional static notion of state transition probability, such choice models are dynamic – responding to the observation. This provides interesting new modeling flexibility. If the portion of x used to influence the choice is generated earlier in the DAG (closer to the sink), then one can build probability models using dynamic choices. We remark, however, that useful models may exist that violate this assumption and therefore generate values that without normalization do not represent probabilities.

2.5 Stochastic Transducers

The notion of *transduction* has its roots in classical formal language theory (e.g. [Ber79]) and represents the formalization of the idea of a machine that converts an input sequence into an output sequence. Later work within the syntactic pattern recognition outlook addressed the induction problem for a particular subclass of finite transducer [OGV93]. More recently a stochastic approach was taken in [BF96] where hidden Markov models were used to capture an input-output relationship between synchronous time series.

Finite growth models can be used to derive Baum-Welch/EM based learning algorithms for *stochastic transducers*. This interesting class of stochastic models capture in a somewhat general way the relationship between dependent symbol streams. In particular there is no assumption that the streams are synchronous, and the symbols singly or jointly generated during transduction can depend upon all context information, i.e. that in every series involved.

Many problems of pattern recognition and artificial intelligence such as speech or handwriting recognition may be viewed as transductions from an input signal to an output stream of discrete symbols – perhaps via one or more intermediate languages. In this conceptual framework it is interesting to note that speech synthesis is merely transduction in the reverse direction. In this section we first discuss transduction in general terms, and then focus in detail on the simple *memoryless* transducer that corresponds to the widely-used notion of string edit distance.

Definition 3 *A k-way <u>stochastic transducer</u> over alphabets $\Sigma_1, \ldots, \Sigma_k$ is a stochastic automaton $A = (S, M)$ where S denotes its states and M its matrix of transition probabilities – such that associated with each transition e is a probability function p_e on $\Sigma_1^* \times \ldots \times \Sigma_k^*$ such that $p_e(\epsilon, \ldots, \epsilon) = 0$ where ϵ denotes the empty string. In the generative view of transduction the machine outputs nonempty k-tuples which are appended to k initially empty output strings. If any of the p_e values depend not just on e but also on the output generated thus-far, then the transducer is said to have <u>memory</u>. Otherwise it is <u>memoryless</u>. Automaton A is assumed to start from state* *zero, and the possibility that the alphabets are continuous is left open.*

Given observed strings $S = (s_1, \ldots, s_k)$ the transducer T assigns a probability $P_T(S)$ which represents the sum over generation sequences that generate S, of the probability of each. The FGM formalism provides a natural way to understand and reason about this complicated process and we now describe the reduction of a stochastic transducer and associated finite observation to an FGM.

Denoting the states of A by $\{S_i\}$ the transducer's infinitely deep temporal branching structure is truncated to form an FGM with states $\{S_i^{\ell_1,\ldots,\ell_k}\}$ where each ℓ_j is a nonnegative integer and $S_0^{0,\ldots,0}$ is the source. The superscript indices correspond to the number of symbols that have been written to the transducer's k output tapes.

The FGM contains another set of states $\{T_{i,j}^{\ell_1,\ldots,\ell_k}\}$ defined similarly except that the subscript refers to a pair of states in the original automaton. Edges exist between these two state sets but not between members of each. There are edges leaving each state $S_i^{\ell_1,\ldots,\ell_k}$ leading to $T_{i,j}^{\ell_1,\ldots,\ell_k}, j = 1, \ldots, |S|$. Row i of stochastic transition matrix M provides the corresponding choice model where the matrix entries are the parameters. The observation model attached to each such edge assumes the constant value 1.

Directed edges exist between T states and S states reflecting the writing of k-tuples to the output tape. For each transition e from state S_i to S_j in the original automaton, several FGM edges may be required because the function p_e may generate output of differing lengths. More formally for all $i_1, \ldots, i_k \in \mathbb{N}^k$ an edge exists leaving $T_{i,j}^{\ell_1,\ldots,\ell_k}$ leading to $S_j^{\ell_1+i_1,\ldots,\ell_k+i_k}$. Each such edge generates observations according to p_e, but only string tuples with lengths $i_1, \ldots, i_k \in \mathbb{N}^k$ are allowed. Void choice models are used for all edges from T to S.

Finally positions in the string-tuple observation are enumerated and appropriate projectors serve as the FGMs selection functions and associate observation models with the generation particular positions in the output tape. Notice that along a source-sink path the corresponding projections are disjoint and do cover all positions but do not in general do so in a simple left-right fashion. This completes our reduction of T to an FGM.

We now evaluate the complexity of FGM operations by counting edges. In general each function p_e may generate output tuples of unbounded total length. But in some cases such as that of string edit distance discussed shortly, the length of the string generated in each tuple position is bounded by some value D.

If L denotes the length of the longest string in the observation tuple, the FGM has $L^k|S|$ states $\{S_i^{\ell_1,\ldots,\ell_k}\}$ and there are $|S|$ edges leaving each of them leading to T states. The T states number $L^k|S|^2$. The out-degree of a T state is clearly $O(D^k)$ making the total number of edges $O(L^k D^k |S|^2)$ and this gives the time and space complexity of FGM operations assuming constant time for primitive model operations. The out-degree of a T state is at most L^k in the FGM so an alternative expression $O(L^{2k}|S|^2)$ applies even when f has infinite support.

Having reduced T to an FGM we identify several important operations and briefly describe their corresponding FGM computation:

1. A <u>transducer learning</u> step consists of improving the parameters of T (those of M and of each p_e) based on a training set $(s_1^1, \ldots, s_k^1), \ldots, (s_1^m, \ldots, s_k^m)$. This computation is performed using standard FGM Baum-Welch/EM steps.

2. There are several <u>transducer evaluation</u> operations:

    (a) <u>Joint</u> — yields the probability of the observation tuple. This is accomplished by standard FGM evaluation and amounts to computing the joint probability of the strings that comprise it.

    (b) <u>Marginal</u> — gives the probability of some subset of the tuple's dimensions, i.e. marginalizes over the remaining dimension. This is easily computed by computing the appropriate marginal of each primitive observation model. That is, if such a model involves one or more marginalized tuple dimensions, the corresponding primitive marginal becomes the value of the observation model.

    (c) <u>Conditional</u> — gives the conditional probability of some subset of the tuple's dimensions given the others. This is best computed by dividing the joint probability by the corresponding marginal.

3. The <u>transduction decode</u> operation consists of maximizing over free-tuple positions their probability conditioned on the other position which are fixed. In the case of 2-way transduction this means fixing one string and finding the most likely second. Since the denominator is constant in the quotient defining the conditional we have only to maximize the numerator, i.e. the joint probability of the selected tuple positions given the other fixed positions. This maximization is performed via Viterbii decode where the fixed positions are regarded as constant. This identifies the most likely transduction path. The result is built by following this path, and for each attached observation model outputting values corresponding to its mode.

A *cascade* operation may also be defined where the output of one transducer becomes the input to another. The development above has then established

Theorem 4 *By reduction to an FGM the evaluation, learning step, and decode operations for a stochastic transducer may be performed in $O(L^k D^k |S|^2)$ or $O(L^{2k}|S|^2)$ time and space assuming constant time primitive model operations. The constant time assumption holds in particular given the use of canonical discrete probability functions for all models.*

Note that the space complexity of evaluation and decode is actually smaller since only a forward pass is performed and there is no need to maintain a fully formed DAG in memory. Also, our earlier comments regarding alternative maximization criteria apply to transducers as well and other generalizations are possible including multiple start states and parameter tying.

2.5.1 String Edit Distance

The edit distance between two finite strings s, t over finite alphabet Σ is the least costly way to transform s into t via single-symbol insertions, deletions, and substitutions. The non-negative costs of these primitive edit operations are parameters to the algorithm. These costs are represented as a table $c_{i,j}$ of size $|\Sigma| + 1 \times |\Sigma| + 1$, where row and column 0 correspond to an additional alphabet member ϵ representing the null character. See [SK83] for review or [HD80] for a compact discussion. The entry in table position i,j gives the cost of substituting symbol j of s for symbol i of t. The first row gives insertion costs, the first column gives deletion costs, and the remaining entries specify substitution costs. The table need not be symmetric. Recall that a simple dynamic program finds the edit distance in $O(|s| \cdot |t|)$ time.

The task of learning an optimal cost table is clearly under-constrained, because the string edit distance for all strings may be trivially minimized by setting all edit costs to zero. Our outlook is stochastic and we therefore interpret each cost as $-\log_2()$ of the corresponding event probability. For example, an entry in the top row expresses the probability of choosing to insert a particular symbol into the left string. The natural constraint then requires that the probabilities that underly the cost table must sum to one. Expressed in terms of costs $c_{i,j}$, this translates to:

$$\sum_{i,j} 2^{-c_{i,j}} = 1$$

Rather than imagining an editing process that transforms s into t, we equivalently imagine that the pair (s,t) is *grown* in a series of steps of three kinds: joint steps, left steps, and right steps. This outlook was first introduced in section 3.2.2 of [HD80]. Our contribution is its further development to include the learning of costs and the construction of more sophisticated distance functions as described later in this section. In [RY96b] the authors implement the learning of costs and give experimental results. This report also

The process is formally a 2-way stochastic transducer with a single state, no memory, and $D = 2$. The initial state of the process is $(\emptyset, \emptyset)$. A joint step concatenates symbols to both elements of the pair. A right step adds a symbol to only the right element of the pair, while a left step adds a symbol to the left element. In the language of string edit distance, a joint step corresponds to a substitution operation. When a joint step adds the same symbol to both strings, one is substituting a symbol for itself. A right step corresponds to an insertion operation, and a left step corresponds to a deletion operation.

Figure 1 depicts the table now denoted $\chi_{i,j}$ after conversion to probabilities. The zero entries correspond to infinite costs. The stochastic choice between left, right, and joint generation is implicit in this table. That is, χ combines choice and observation probabilities. For example, the sum of the top row may be interpreted as the probability of choosing to perform a left insertion.

Definition 4 *The "stochastic edit distance" between strings s and t, is given by $-\log_2(P((s,t)|\chi)$. Here the probability refers to the infinite generative stochastic process in which symbols are inserted at the end of the right string, or at the end of the left string, or jointly at the end of both. Specifically, it is the probability that pair (s,t) will occur as an intermediate stage of generation, during a random trial.*

Figure 2 depicts the FGM corresponding to stochastic edit distance between strings of length 4 and 5. Its 3-way branching pattern follows from the structure of the corresponding growth process and matches exactly the dependency structure of the dynamic program for conventional edit distance.

Notice that all but the sink and the bottom and rightmost vertices have ternary out-degree, and that the edges leaving them are drawn with solid lines. These are all in tied. Both their choice and observation models are specified by χ corresponding to normal operation of the generative stochastic process. Edges drawn with dotted lines implement truncation of the process and employ null models. These might have been depicted instead as individual long edges directly to the sink.

The $-\log_2()$ operation converts the probability to a non-negative "distance", which we will later see corresponds closely to the standard notion of edit distance. Also notice that in the definition, we refer to the probability of the ordered pair (s,t). This is an important distinction since χ need not be symmetric, and it might be that $P((s,t)|\chi) \neq P((t,s)|\chi)$.

It is also possible to interpret $P((s,t)|\chi)$ with respect to finite event spaces. If we denote by $S_{|s|,|t|}$ the FGM state corresponding to the argument string lengths, then the probability defined above may be written: $P((s,t), S_{|s|,|t|}|\chi)$. That is, the probability of generating s and t while passing through FGM vertex with coordinates $(|s|,|t|)$.

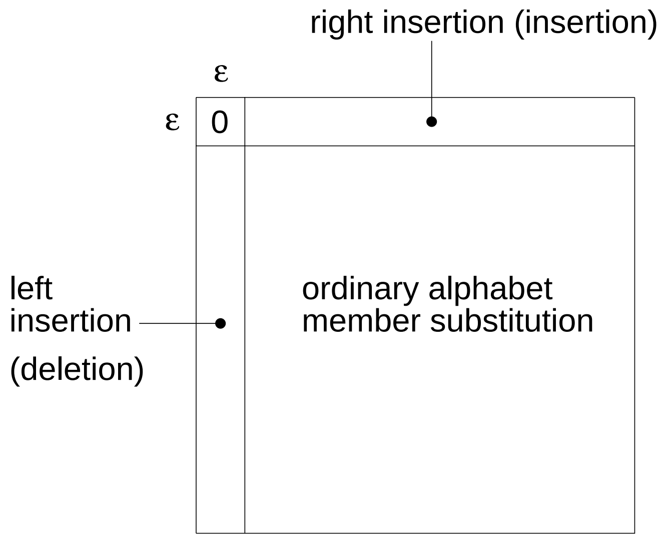

Figure 1: Augmented table of edit distance probabilities. This table sums to one, and the corresponding edit costs are $-\log_2()$ of its values.

It is straightforward to compute the probability of event $S_{|s|,|t|}$. Division then yields the conditional probability $P((s,t)|\chi, S_{|s|,|t|})$ which amounts to conditioning on string lengths. It is also possible to define the FGM differently in order to directly implement conditioning on string lengths [RY96b].

The stochastic edit distance $-\log_2 P((s,t)|\chi)$ is highly related but not identical to conventional edit distance. If stochastic table χ is converted by $-\log_2()$ to a table of non-negative costs then conventional edit distance corresponds to the log-probability of the *single best path* through the FGM – i.e. the Viterbi decode. Stochastic edit distance is the sum over *all paths*. Basing decisions on the Viterbi decode is analogous to focusing on the mode of a distribution while stochastic edit distance corresponds to the mean. In classification systems we expect that both forms will lead to nearly identical decisions. The improved costs we learn in the next section will provably reduce stochastic edit distances, but we also expect that in practice they will also improve results for the conventional form.

We formalize the application of FGMs to stochastic edit distance with the following:

Corollary 1 *Let χ be a table of stochastic edit distance probabilities, and (s_1, t_1),*

(s_2, t_2), ..., (s_T, t_T) denote a set of string pairs over finite alphabet Σ, where ℓ bounds the length of any single string. Next let $E_\chi(s,t)$ denote the stochastic edit distance $-\log_2 P((s,t)|\chi)$. Then:

1. *There exists an $O(|s| \cdot |t|)$ time, $O(\min(|s|, |t|))$ space algorithm to compute $E_\chi(s,t)$ – matching the complexity of the conventional dynamic program.*

2. *There exists an $O(\ell^2 T)$ time and space algorithm for producing a revised cost table χ' using Baum-Welch/EM such that:*

$$\sum_{i=1}^{T} E_{\chi'}(s_i, t_i) \leq \sum_{i=1}^{T} E_\chi(s_i, t_i)$$

   where the inequality is strict unless χ is a critical point.

proof: Apply theorem 4. $\square$

The FGM formalism is useful to arrive at a correct solution but because of the extensive parameter tying employed in our construction, the associated computations may be greatly simplified. There is no need to build actual DAGs during the computation. Also, as remarked earlier, the choice model is implicit in χ so that creation of separate choice and observation models is unnecessary when computing $E_\chi(s,t)$. The probability of passing over an edge is just the corresponding entry in χ.

The result is algorithm 7. It requires an $(|s|+1) \times 2$ array of α values. We write $\alpha_{i,j}$ to denote an element, where $1 \leq i \leq |s|+1$ and j is 1 or 2. This array corresponds to the vertices within two adjacent columns of the FGM as depicted in figure 2. The logarithm E_χ of the joint probability of s and t is returned rather than the probability itself, so that the caller need not deal with extended range floating point values. As observed by [HD80], the algorithm's structure resembles that of the standard dynamic program for edit distance.

Algorithm 7

procedure *Evaluate(s, t)*
  for $j = 1, \ldots, |t| + 1$
    $k = 2 - j \bmod 2$
    $\ell = 1 + j \bmod 2$
    for $i = 1, \ldots, |s| + 1$
      if $i > 1 \vee j > 1$
        $\alpha_{i,k} = 0$
      else
        $\alpha_{1,1} = 1$
      if $i > 1$
        $\alpha_{i,k} += \alpha_{(i-1),k} \cdot \chi_{s_{(i-1)},0}$
      if $j > 1$
        $\alpha_{i,k} += \alpha_{i,\ell} \cdot \chi_{0,t_{(j-1)}}$
      if $i > 1 \wedge j > 1$
        $\alpha_{i,k} += \alpha_{(i-1),\ell} \cdot \chi_{s_{(i-1)},t_{(j-1)}}$
  return $\log_2 \alpha_{(|s|+1),k}$

As in the case of evaluation, there is no need to create separate choice and observation models when implementing reestimation. A brief argument establishes this fact. Assume we do have separate models. Then each edge e is of type *left*, *right*, or *joint*, and during Baum-Welch/EM the quantity γ_e will be added to a choice accumulator and also to an accumulator corresponding to the observed alphabet symbol or symbols. So the total γ contribution to each of the three choice accumulators will exactly match the contribution to its corresponding observation model. Hence, if we instead add the γ values directly into an initially zero array $\overline{\chi}$, the total contributions to the left column, top row, and interior will exactly correspond to the choice accumulator contributions above. So after normalization, $\overline{\chi}$ will represent the same model as that resulting from reestimation based on separate models.

Algorithm 8 is the central routine in the stochastic edit distance learning process.

It is called repeatedly with pairs of strings. The current cost table $\chi_{i,j}$ must be initialized before the first call and the improved cost table $\overline{\chi}$ must be set to zeros. As each pair is processed, non-negative values are added to locations within $\overline{\chi}$. After all the pairs have been presented the $\overline{\chi}$ array is simply normalized so that the sum of its entries is one, and then represents the set of improved costs. This entire is repeated to further improve the model. That is, $\overline{\chi}$ is copied to χ and $\overline{\chi}$ is again set to zeros. The sequence of training pairs is then presented again, and so on.

The algorithm uses two work arrays, $\alpha_{i,j}$ and $\beta_{i,j}$, corresponding to the algorithms 1 and 2 respectively. It is assumed that these arrays are large enough to accommodate the training pairs. They need not be initialized by the caller. As in algorithm 7 it is possible to reduce the size either α or β but not both. Doing so results in a constant 2:1 space savings but for clarity we present the algorithm using complete arrays.

Algorithm 8

procedure *LearnCosts(s, t)*

 for $i = 1, \ldots, |s| + 1$

  for $j = 1, \ldots, |t| + 1$

   if $i > 1 \vee j > 1$

    $\alpha_{i,j} = 0$

   else

    $\alpha_{1,1} = 1$

   if $i > 1$

    $\alpha_{i,j} += \alpha_{(i-1),j} \cdot \chi_{s_{(i-1)},0}$

   if $j > 1$

    $\alpha_{i,j} += \alpha_{i,(j-1)} \cdot \chi_{0,t_{(j-1)}}$

   if $i > 1 \wedge j > 1$

    $\alpha_{i,j} += \alpha_{(i-1),(j-1)} \cdot \chi_{s_{(i-1)},t_{(j-1)}}$

 for $i = |s| + 1, \ldots, 1$

  for $j = |t| + 1, \ldots, 1$

   if $i \leq |s| \vee j \leq |t|$

    $\beta_{i,j} = 0$

   else

    $\beta_{(|s|+1),(|t|+1|)} = 1$

   if $i \leq |s|$

    $\beta_{i,j} += \beta_{(i+1),j} \cdot \chi_{s_i,0}$

    $\overline{\chi}_{s_i,0} += (\alpha_{i,j} \cdot \chi_{s_i,0} \cdot \beta_{(i+1),j}) / \alpha_{(|s|+1),(|t|+1)}$

   if $j \leq |t|$

    $\beta_{i,j} += \beta_{i,(j+1)} \cdot \chi_{0,t_j}$

    $\overline{\chi}_{0,t_j} += (\alpha_{i,j} \cdot \chi_{0,t_j} \cdot \beta_{i,(j+1)}) / \alpha_{(|s|+1),(|t|+1)}$

   if $i \leq |s| \wedge j \leq |t|$

    $\beta_{i,j} += \beta_{(i+1),(j+1)} \cdot \chi_{s_i,t_j}$

    $\overline{\chi}_{s_i,t_j} += (\alpha_{i,j} \cdot \chi_{s_i,t_j} \cdot \beta_{(i+1),(j+1)}) / \alpha_{(|s|+1),(|t|+1)}$

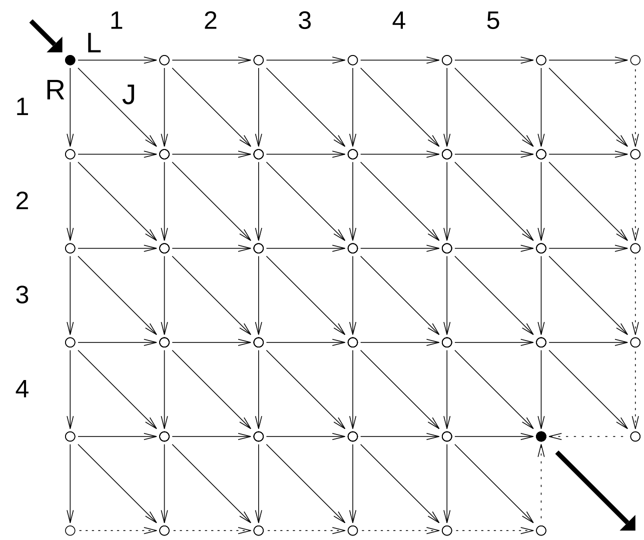

Figure 2: Finite growth model DAG corresponding to computation of the edit distance between strings of length 4 and 5.

In algorithm 8 we remark that one should escape after the α computation in the event of a zero probability observation. Also, the self-check of proposition 3 has a particularly simple interpretation here: if one accumulates all contributions to $\overline{\chi}$, doubling those corresponding to substitutions, the result is exactly $|s| + |t|$.

We now illustrate by example that one has no guarantee of convergence to a global optimum. Let $s = ABB$ and $t = CC$ provide the sole training pair. If the probability of substitutions A, C and B, C are 0.32, and that of left-inserting B is also 0.32, and left-inserting A has probability 0.04, then learning converges to a local optimum in which these operations have probability 1/3, 1/3, 1/3, and 0 respectively. The algorithm has in effect chosen to edit s into t by substituting A, C, then B, C and finally left-inserting A. At convergence the pair's distance is 3.75 bits. But it is clearly better to left-insert A and perform two B, C substitutions. Initializing the edit operations near to this objective leads learning to the desired solution and the pair's distance is reduced to about 2.75 bits. Initializing the four parameters to 0.25 yields a third local maximum giving distance 3.92 bits.

String edit distance is a very simple example of stochastic transduction. Using it to ground our discussion we now return to more general observations.

The costs employed in the edit distance computation depend only on the targets of the edit operations. They are independent of the context in which the edit operation occurs. Yet in most naturally occurring edit processes the likelihood of an edit operation depends strongly on the context in which the operation occurs. Within the FGM framework one may strengthen the model for edit distance so that the cost of an edit operation depends on the context in which that operation is performed. Insertion and deletion costs can depend on one or more positions of trailing context in a single string while substitution costs can depend on a joint context constructed from both strings. This modeling technique should lead to more powerful similarity functions and is applicable to transducers in general.

Context models may also be built by encoding context into the state space. To do this one limits observation models to trivial probability functions that assign probability 1 to a single alphabet member. After passing over an edge one then has certain knowledge of the most recent symbol. Beyond this the graph can be designed so that arrival in a state implies exact knowledge of a trailing finite context of any desired length. In effect the modeling *work* is moved entirely to the choice models. Having arrived at a vertex the choice model predicts the next symbol conditioned on the past. Note that this technique is an example of something that *does not* generalize to continuous observations.

To specify simple stochastic edit distance, we must learn only a single χ table. With contexts, many more free parameters are added to the model. In principle very large context models can be built, but in practice their size is limited by the amount of training data available increasing the need for MDL or MAP guided optimization.

The learning of a 2-way transducer's parameters represents one formalization of what the authors have called the *metric learning* paradigm. The training set may be regarded as positive examples of the *are similar* concept and after learning, the negated logarithm of the FGM's joint evaluation or conditional evaluation is viewed as a measure of *dissimilarity*. The mathematical metric axioms are not in general satisfied so by "metric learning" we really mean *any learned quantification of similarity or dissimilarity*.

This idea applies beyond string settings and for processes more complex than our definition of transduction which specifies that a tuple observation is grown from left to right by concatenation. The string edit distance DAG need not be viewed as arising from 2-way transduction. The salient characteristic that implements metric learning is that some edges observe both strings while others see only one. It is not essential that as one moves from source to sink the string positions considered within each tuple position also move from left to right.

It is then clear that distances analogous to string edit distance may be defined for objects whose natural growth description is more complex than the linear DAG that corresponds to a string. We do not fully develop this direction but now make precise the corresponding DAG structure.

In the case of edit distance each FGM vertex faces three choices. Two choices correspond to single symbol generation, and a third choice selects joint generation. One may view the resulting graph as a product of two linear FGMs, each of which grows a single string from left to right. We now generalize this notion. Given two FGMs F_A, F_B we define the generation DAG of the product FGM denoted $F_A \otimes F_B$ as follows:

1. The vertices of $F_A \otimes F_B$ are pairs (v, w) where v, w are vertices of F_A and F_B respectively.

2. Edge $(v, w) \rightarrow (r, s)$ exists in $F_A \otimes F_B$ iff:

   (a) $v \rightarrow r \in F_A$ and $w = s$. Edges of this type *run* the left FGM only while keeping the right *idle*. For edit distance this corresponds to insertion into the left string, or;

   (b) $w \rightarrow s \in F_B$ and $v = r$ corresponding in edit distance to right string insertion, or;

   (c) $v \rightarrow r \in F_A$ and $w \rightarrow s \in F_B$. Here both machines run and a pair of symbols are generated corresponding to edit distance substitution.

It is readily verified that $F_A \otimes F_B$ has a single source and sink, and that each path corresponds to a permutation of the joint observation tuple so that after specifying the necessary choice and observation models, a well-formed SFGM results.

2.6 New Perspectives on Existing Models

In this section we relate FGMs to three models in widespread use: hidden Markov models, stochastic context free grammars, and normal mixtures. The problem of optimizing the log return of a fixed portfolio is also shown to correspond to a trivial FGM. We focus on reduction of these problems in standard form to FGMs but remark that any of them may be improved by using MDL or MAP guided optimization as described earlier.

2.6.1 Hidden Markov Models

A hidden Markov model (HMM) is an automaton that changes state stochastically after generating an observation. See [Por88, HAJ90] for reviews. Its state transitions are given by a stochastic $n \times n$ matrix M, a vector π gives the probability of starting in each of its n states, an output function b_i is associated with each state i, and there are one or more designated *final* states. As time passes the machine changes states and generates observations which form an output *time series*.

An HMM is essentially a 1-way stochastic transducer restricted so that all transitions *from* a state specify the same observation model and generate a single observation component ($D = 1$). If there are more than one possible starting states then the entire corresponding FGM is preceded by a choice that plays the role of vector π. The edges leading to the sink are weighted one or zero according to membership in the set of final states.

Because output distributions are associated with states not edges in an HMM, every FGM edge leaving a state is associated with the same distribution. For this reason the transducer reduction can be simplified by eliminating the T states whose only purpose is to remember the previous state. So if the HMM states are denoted

$S_1, \ldots, S_n$ then the FGM states are just $\{S_i^t\}$ where the superscript corresponds to time.

This reduction of an HMM to an FGM amounts to the simple *unfolding* in time of the HMM's operation and is therefore done with no loss of efficiency. In some sense it is a simpler representation of the HMM because the temporal order of events has been made explicit.

Reduction in the other direction is problematic. Given a finite alphabet it is not hard to construct an equivalent HMM, i.e. one that imposes the same extrinsic probability distribution. But the canonical such machine may have exponentially more states because an FGM can generate observations in arbitrary order. Moreover, since an FGM can generate observations jointly, the standard HMM formulae for reestimation do not apply. Finally, an FGM that generates joint continuous observations will correspond to an HMM only if these joint densities are trivial, i.e. a product of independent terms. This suggests that FGMs may be more expressive than HMMs.

We now discuss an important variation: *time duration HMMs* [Lev86, RC85] and sketch their rederivation in FGM terms illustrating the utility of parameterized choice models. See [HAJ90] pages 218–221 for a brief review. If in a conventional HMM a state transitions to itself with probability p, then duration in that state is geometrically distributed, i.e. as $p^{t-1}(1-p)$. This functional form restricts the model's ability to match the characteristics of some applications such as phoneme recognition.

In an ordinary HMM, diagonal transition matrix entries give the self-transition probabilities. These will be modeled separately so M is assumed to have a zero diagonal. The reduction of a time duration HMM to an FGM starts with the same state set $\{S_i^t\}$ but requires an additional set $\{V_i^t\}$. Associated with a state S_i^t is a parameterized choice model that assigns a probability to each possible state duration Δ from $1-\infty$. Edges exist from S_i^t to states $V_i^{t+\Delta}$ and along each, Δ observations are generated according to the same distribution. To express this strictly within the FGM framework a linear sequence of Δ edges is necessary with a single observation model attached to each. Associated with $V_i^{t+\Delta}$ is a simple discrete choice model induced by transition matrix M. Edges are then directed to the corresponding $S_j^{t+\Delta}$ state with no attached observation model. Notice that since the diagonal of M is zero there is no edge leading to $S_i^{t+\Delta}$. Even though the choice model associated with each S state has infinite domain, the observation is finite so only a bounded number of edges are included in the FGM.

The simplest such infinite choice model has a parameter for every duration value – a nonparametric approach. Since the training set is finite this reduces to a standard discrete choice model which is easily maximized. Because of limited training data one might prefer a model with fewer parameters. A simple way to reduce parameters is to group them into bins and assume that within a bin probability is uniform. For example bin sizes might be $1, 2, 4, 8, \ldots$. Here duration 1 lies in the first bin, durations $2, 3$ in the second, $4, 5, 6, 7$ in the third, and so on. Parametric models may also be used.

We close our discussion of HMMs with the topic of *nonemitting* states, i.e. states that do not generate any output. These complicate our reduction to FGMs and conventional processing as well since cycles among them may have unbounded duration. Common practice is to preprocess models with such states in order to eliminate self-cycles. Denote by N the set of nonemitting states and by $S - N$ the rest. Now suppose a transition occurs from an ordinary state to a member of N. The next transition may leave N immediately or cycle through its states for some time before doing so. But since no observations are generated we are merely interested in the ultimate departure event. For each state in N it is straightforward to compute the probability of ultimately exiting to one of the states in $S - N$. These values become entries in the transition matrix associated with that state. All entries associated with self-transitions among N are set to zero. This modified machine will impose the same extrinsic distribution as the original but is free of nonemitting cycles and may be reduced to an FGM.

2.6.2 Stochastic Context Free Grammars

A *stochastic context free grammar* (SCFG) is a conventional context free grammar [HU79] with a probability attached to each production such that for every nonterminal

A the probabilities attached to all productions "$A \rightarrow \ldots$" sum to unity. Applied to a sentence of finite length these grammars have an interesting recursive FGM structure. Baum-Welch/EM may then be used to learn the attached probabilities from a corpus of training sentences. The result is essentially the same as the *inside-outside* algorithm described in [Por88].

We assume the grammar G is in Chomsky normal form and use t to denote the input sentence. Viewing G as a stochastic generator of strings each nonterminal A may expand into sentences of many lengths. We write $P(A \rightarrow t[i \ldots j])$ to denote the probability that A will generate the substring of t consisting of its ith through jth positions. In particular the probability of t is $P(S \rightarrow t[1 \ldots L])$ where L denotes the length of t and S is the start symbol. For a terminal b, $P(b \rightarrow t[i \ldots j]) = 1$ if $i = j$ and $t[i] = b$, and zero otherwise. The SCFG is a probability function over all sentences, so that it is a subprobability model for those of any fixed length. For this reason null models (described earlier) are employed in the construction to truncate the normally infinite generation process.

The FGM corresponding to G restricted to length L is a recursive construction starting with the start symbol. We name the top level FGM $S^{1 \ldots L}$ because it computes the probability of S applied to $t[1 \ldots L]$. The construction is the same for each nonterminal A with corresponding productions $A_1, \ldots, A_{N(A)}$ where $N(A)$ denotes the number of productions of the form "$A \rightarrow \ldots$". The FGM $A^{i \ldots j}$ has $N(A)$ edges from source to sink and computes $P(A \rightarrow t[i \ldots j])$. Its single choice model corresponds to the probabilities associated with A in the grammar. Each edge generates observations $i, \ldots, j$ and the attached models are denoted $A_1^{i \ldots j}, \ldots, A_{N(A)}^{i \ldots j}$ where each $A_\ell^{i \ldots j}$ is defined as follows:

1. If production A_ℓ is of the form $A \rightarrow b$ where b is a terminal, then the model's value is $P(b \rightarrow t[i \ldots j])$ as defined above. Within the FGM formalism this is achieved by using a null model when $i \neq j$, and otherwise a discrete model with $P(b) = 1$.

2. If production A_ℓ is of the form $A \rightarrow BC$ and $i = j$ then a null model is used since each of B and C must ultimately generate at least one terminal. If $j > i$

then the attached model consists of an FGM $[BC]^{i..j}$ which encodes all ways in which BC might generate $t[i \ldots j]$. Each of its $j - i$ source-sink paths is distinct and consists of two edges. Along the kth path numbered from zero, the first edge invokes FGM $B^{i...i+k}$ and the second $C^{j-k...j}$. The generation events corresponding to each path are mutually exclusive so a void choice model is employed. That is:

$$P(BC \to t[i \ldots j]) = \sum_{k=0}^{j-i-1} P(B \to t[i \ldots i+k]) \cdot P(C \to t[j-k \ldots j])$$

Each FGM's $B^{i...i+k}$ and $C^{j-k...j}$ generate strictly shorter substrings of t than does $A^{i...j}$ whence the recursion is bounded and ultimately descends to productions of the form $A \to b$ described above. The choice models are tied across occurrences of each FGM $A^{i...j}$ in this recursive hierarchy.

The time and space complexity of the resulting model follows easily from an analysis of its structure. There are $O(L^2)$ FGMs of the form $A^{i...j}$ and $O(L^2)$ of the form $[BC]^{i...j}$. The first kind contains $O(1)$ edges while the second has $O(L)$ edges. The result is $O(L^3)$ time and space complexity. The same counting of FGMs and edges demonstrates that complexity is linear in the grammar's size.

Beyond emulating conventional SCFGs, our FGM construction suggests more complex models. For example, it is possible to learn a probability for each production conditioned on the length of the sentence it generates. Here the single value associated with each production is replaced with a vector of probabilities indexed by sentence length. The vector's length may be set to the maximum length of a sentence in the training corpus. The result is a model which generates only sentences of finite length. More generally each vector position may correspond to a range of lengths, e.g. less than 5, between 6 and 20, 21 and over. If the final range is open-ended then the model generates sentences of unbounded length, as does a SCFG. This new model has more parameters, but given enough training data may outperform an SCFG. To implement it one merely ties the choice models associated with each instance of each FGM $A^{i...j}$ differently. In the construction above they are all tied. Instead we tie the choice model for $A^{i \ldots j}$ to $A^{k \ldots \ell}$ provided $j - i = \ell - k$, i.e. both instances generate output of the same length.

2.6.3 Normal Mixtures

A k-component normal mixture is a probability density function of the form:

$$f(x|\mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k) = \sum_{i=1}^{k} c_i \cdot N_{\mu_i, \Sigma_i}(x)$$

where $\sum_{i=1}^{k} c_i = 1$, $c_i \geq 0$ for $i = 1, \ldots, k$, and $N(\cdot)$ denotes a multivariate normal density. Such a mixture corresponds to a trivial FGM that contains exactly k DAG edges, each from source to sink, and a normal density along each. The FGM's single choice model corresponds to the mixing parameters $c_1, \ldots, c_k$. Unlike transduction, HMMs, and SCFGs, normal mixtures are an example of an FGM which generates an observation tuple of fixed finite dimension. They may be used as the observation model along any edge in a parent FGM. Our earlier reduction of HMMs to FGMs then results in the mixture-density HMMs widely used in speech recognition. To illustrate the variations possible we observe that tying every component with the same index to a single density yields the *semi-continuous* variation [HJ89].

2.6.4 Portfolio Optimization

Our final example consists of a problem that is not strictly speaking a stochastic probability model but nevertheless fits easily into the FGM formalism. Given *stocks* $x_1, \ldots, x_d$ and observations of their returns over many fixed time periods, the objective is to decide how much of each should be held in an optimal investment portfolio. A trivial reduction to FGMs results in a simple iterative algorithm. This problem is considered in [Cov84] where the same algorithm, essentially a rediscovery of Baum-Welch/EM, is described. The *multiplicative update* perspective of [Cov84] is essentially the growth transform of [BPSW70].

We describe this problem using the notation of [Cov84] and reduce it to an FGM. A stock which appreciates by 25% in say one month, is said to have a return of 1.25.

The returns of each stock over time period i forms a vector denoted X^i. We use stochastic vector b to denote our portfolio allocation and the portfolio's return over period i is then $b^t X^i$. This is captured by a simple FGM consisting of a d-way choice corresponding to b with each edge leading directly to the sink. An unnormalized observation model is attached to each edge and assumes the value of the selected stock's return. Given T observations $X^1, \ldots, X^T$, and letting $\mathrm{diag}(b)$ denote the diagonal matrix formed from b, one Baum-Welch/EM iteration is given by:

$$\overline{b} = \sum_{j=1}^{T} \frac{\mathrm{diag}(b) X^j}{b^t X^j}$$

after $\overline{b}$ is normalized to have unity sum. This matches the equation given in [Cov84]. Assuming the X^j occur with equal probability then maximizing the training set's likelihood also maximizes our expected log return from the portfolio.

2.6.5 Other Applications

Bayes nets [Pea88], also known as graphical models [Lau96], can be represented in their simplest forms as FGMs in which vertices correspond to variables, edge weights to conditional probabilities, and vertex *a posteriori* probabilities (γ) to belief. More complex networks can still be represented as FGMs although the reduction is more involved. The relationship between these networks and hidden Markov models is elucidated in [SHJ96].

Exploring the relationship of FGMs to error correcting codes such as those based on trellises, and the recently developed class of *turbo codes* [BGT93], is a subject for future work. We wonder broadly whether our framework's generality might lead to better codes, and in particular whether DAGs that encode nonlinear time orderings might have value. Such a possibility is suggested by [WLK95, Wib96]. The relationship between Turbo codes and Bayes nets is examined in [MMC96] and understanding the relationship of FGMs to this specific outlook is another interesting area for future work.

Acknowledgments

We thank Andrew Appel, Vince Poor, Bob Sedgewick, and Bob Tarjan for their comments on this work, and Joe Kupin for helpful discussions regarding stochastic transduction.

Chapter 3

Learning String Edit Distance Costs

Chapter 2 introduced the application of finite growth models to the problem of learning the parameters of stochastic transducers in rather general form, and considered the case of a *string edit distance* in greater detail. That is, a memoryless single state 2-way transducer.

In this chapter we establish the practical utility of this approach by learning an optimal string edit distance function from a corpus of examples and reporting on the performance of this function in a concrete application: the identification of words in conversational speech given a phonetic transcription and reference dictionary[1]. The function we learn exhibits one third the error rate of Levenshtein distance [Lev66]; the canonical untrained edit distance case. We first implemented the transduction at the heart of our approach in August 1993 for the problem of classifying grey-scale images of handwritten digits.

3.1 Introduction

The speech recognition problem may be viewed as a transduction $S \to T$ from a signal S to text T. The text is typically regarded as a sequence of words from a fixed vocabulary recorded in a lexicon L. This transduction bridges a wide representational

[1]The work described in this chapter was first described in [RY96b] and will appear in [RY97a]

gap and investigators commonly introduce an intermediate phonetic language P and the problem is then viewed as $S \to P \to T$.

Our experiments focus on the second segment $P \to T$. Crossing the entire gap from S to T through the learning of multistage transductors represents an interesting area for future work.

We assume that P is given and seek to identify the correct corresponding sequence of words from L. Moreover, we assume that the word boundaries within P are known, and that L contains one or more valid phonetic spellings for each word in the vocabulary.

Our problem is then to accept a sequence of noisy phonetic spellings taken from P, and output the correct sequence of words from L. This problem has the virtue of being easily stated but retains a strong connection to the original speech recognition task.

The corpus we use is considered to be quite difficult and indeed current systems exhibit an error rate of above 45% on it. This is in contrast to error rates closer to 5% on easier tests. We describe and evaluate several variations on a basic model. The best of these exhibits an error rate of slightly more than 15%, where we remind the reader that this figure applies to the $P \to T$ leg only and assumes knowledge of word boundaries.

The phonetic spellings that form our input are generated by human experts listening to S. For this reason our results might be viewed as an optimistic estimate of what is possible. On the other hand, generation of the output word sequence involves no high order language model; only the probabilities of individual words are considered when forming the output sequence. Such high order models are an important factor in existing speech systems and might therefore reasonably be expected to improve results under our approach. Also, more sophisticated context sensitive transducers are possible within our theoretical framework, and these also promise to improve performance. So despite our simplifying assumptions we submit that our experimental results are of some practical significance to the original problem.

3.2 The Basic Model and Approach

Our generative model begins with the choice of a word w according to a simple probability function on the vocabulary. Given w a particular lexical phonetic spelling ℓ is selected from the alternatives recorded in L – again according to a probability function. Finally, a surface phonetic spelling s is generated from ℓ via stochastic transduction – and in the case of our experiments, using a single state memoryless transducer corresponding to string edit distance. Note that the alphabet used to express lexical phonetic spellings may differ from that used to represent the surface phonetic spellings seen in P. This model is written:

$$p(s, \ell, w) = p(s|\ell)p(\ell|w)p(w)$$

and the recognition problem we face is formalized as:

$$\begin{aligned}
\hat{w} &= \mathrm{argmax}_w \quad p(w|s) \\
&= \mathrm{argmax}_w \quad p(w, s)/p(s) \\
&= \mathrm{argmax}_w \quad p(w, s) \\
&= \mathrm{argmax}_w \quad \sum_{\ell \in L(w)} p(s|\ell)p(\ell|w)p(w)
\end{aligned}$$

where $L(w)$ denotes the set of lexical phonetic spellings that L records for w – and we observe surface phonetic spelling s. The transduction-based marginal $p(s|\ell)$ is properly computed by dividing the standard joint transduction probability $p(s, \ell)$ by the marginal evaluation $p(\ell)$. In our experiments however, the unnormalized joint probability was used instead so that the framework above is only approximated. We will return to this subject at the chapter's conclusion.

The issue is then one of estimating the parameters of this model which were omitted from the notation above for simplicity. We now begin using Φ to denote them. They include the probability function used to choose words, that used to choose lexical phonetic spellings, and the transducer's parameters as well.

We are given a training set of pairs $(w, s(w))$ where w is a vocabulary word and $s(w)$ its surface phonetic transcription. Our learning objective is then to maximize over Φ:

$$\prod_{w \in \text{train}} \sum_{\ell \in L(w)} p(s(w)|\ell, \Phi) p(\ell|w, \Phi) p(w|, \Phi)$$

The test set consists only of surface phonetic forms, and we face the recognition problem described above which recovers a vocabulary word for each of these observed surface forms. This objective is recognized as an instance of the ML mixture density/function parameter estimation problem and we apply the expectation maximization (EM) [DLR77] as discussed in chapter 2.

Our implementation actually models the product $p(\ell|w, \Phi)p(w|\Phi)$ as a single joint probability $p(\ell, w|\Phi)$. The initial parameters for this mixture and all edit distance costs are uniform. A fixed value of 0.1 is added to the mixture's expectation accumulators as a statistical *flattening* precaution – preventing the probability of any lexicon entry from being driven to zero during learning. No effort was made to optimize this constant. Each EM iteration replaces Φ with new values.

The *a posteriori* probabilities $p(\ell|s(w), \Phi)$ are computed during EM for each $\ell \in L(w)$ and these are used to weight the expectation steps taken for the conditional transduction $p(\ell|s(w)|\Phi)$. Here we make the simplifying assumption that this conditional form may effectively be maximized by instead maximizing the joint transducer probability $p(\ell, s(w)|\Phi)$ so that we can directly apply the FGM-based algorithm for learning string edit distance described in the previous chapter. Strictly then the new Φ may not be an improvement but empirically is.

This approach to learning solves an important problem: that of corresponding the observed surface phonetic form $s(w)$ with the hidden lexical phonetic forms $\ell \in L(w)$. Intuitively $s(w)$ is paired with the most likely ℓ and used to train the transducer. Of course this matching is stochastic. Later in this chapter we will see that the simpler approach of merely training all possible pairs with equal emphasis results in poor performance indeed.

3.3 The Switchboard Corpus

The Switchboard corpus consists of over 3 million words of recorded telephone speech conversations [GHM95]. The variability resulting from its spontaneous nature makes it one of the most difficult corpora for speech recognition research. Current recognizers exhibit word error rates of above 45% on it – compared to rates closer to 5% on easier tests based on read speech. Switchboard also includes a lexicon with 71,200 entries using a modified Pronlex alphabet (long form) [LDC95].

Roughly 280,000 of Switchboard's over 3 million words have been phonetically transcribed manually by ICSI using a proprietary alphabet [GHE96].

Both the Switchboard lexicon and ICSI transcription were filtered to make the two compatible. We removed 148 entries from the lexicon that included unusual punctuation ([<!.]). A total of 73,068 transcript deletions were made as follows: 72,257 with no corresponding valid syntactic word, 688 having empty phonetic transcript, 88 with an incomplete transcript, 27 and 8 that included the undocumented symbols `?` and `!` respectively. Note that symbols `?` and `!` are not part of either alphabet (long forms) and are used only in the transcript.

The final lexicon used in our experiments contains 70,952 entries describing 66,284 syntactic words over an alphabet of 42 phonemes. The final transcript used in our experiments has 214,310 entries – 23,955 of which are distinct. These correspond to 9,015 syntactic words over 43 phonemes (32 Pronlex plus a special silence symbol). The transcript was divided 9:1 into 192,879 training and 21,431 test words.

3.4 Experimental Variations

We report on the results of 56 different experiments resulting from an exploration of several different degrees of freedom:

$$4 \text{ lexicons} \times 7 \text{ distance functions} \times 2 \text{ learning-recognition strategies}$$

3.4.1 Lexicons

Our first set E1 of experiments uses the Switchboard lexicon. This lexicon contains many words that are never used in the ICSI transcript and experiment E2 replaces it with a lexicon containing only those 9,621 entries that correspond to words that do occur in the transcript.

Experiments E3 and E4 leave the Switchboard lexicon behind and construct one instead from the ICSI transcript. Experiment E3 uses a lexicon built from the training transcript. It contains 22,140 entries for 8,570 syntactic words. The test transcript includes 512 entries that do not occur in this lexicon so that 2.4% represents an error rate floor for experiment E3. The lexicon for experiment E4 is built using the entire transcript and has 23,995 entries for 9,015 words.

We report the fraction of misclassified samples in the testing transcript, that is the *word error rate*. For each surface phonetic form in the test transcript, one or more syntactic words are postulated by the decision process. The fraction of correctly classified words is then the sum over the test transcript of the ratio of the number of correct words identified to the number of postulated words. The fraction misclassified is then one minus this fraction.

With nothing more said experiment E4 would seem to be uninteresting since its lexicon contains an exact match for each phonetic spelling present within the test transcript. One might therefore expect 100% accuracy from this experiment. But the lexicon contains a large number of homophones, that is different syntactic words that have at least one phonetic spelling in common. This arises from two sources. First, conventional phonetic alphabets have limited expressiveness, especially when applied to short words in rapidly articulated conversational speech. Second, in such speech, words frequently change their phonetic *shape* considerably and in so doing collide with other words. That these phenomena are significant is clearly demonstrated by our results (reported later) for experiment E4.

3.4.2 Distance Functions

Three basic forms of string edit distance are compared in our experiments. The simplest of these is the classic Levenshtein prescription which assigns zero cost to identity substitutions and unity cost to all other operations. Stochastic edit distance is defined in chapter 2 as the negative logarithm of the probability that two strings are simultaneously *grown* using left-insertion, right-insertion, and joint-insertion operations. This form effectively aggregates all transductions between the two strings. Viterbi edit distance instead selects only a single transduction (path through the FGM) of maximal probability. It is then identical to conventional edit distance where the edit costs are the negative logarithms of the corresponding generative insertion probabilities. Note that by definition the Viterbi edit distance can be no less than the stochastic edit distance. Also observe that the Levenshtein form has no stochastic interpretation strictly within our framework since infinitely many identity substitutions is less costly than even a single unity cost operation. The problem derives from the fact that its cost table includes zero entries. Any such table is problematic. But given that all entries are strictly positive, there is always an equivalent stochastic Viterbi form. So in particular the Levenshtein form may be approximated by assigning the zero cost operations some small cost ϵ.

Levenshtein distance is parameter-free but may be viewed as having four parameters: identity substitution, nonidentity substitution, insertion, and deletion – set to 0, 1, 1, and 1 respectively.

Both the Viterbi and stochastic forms of edit distance are parameterized by their table of elementary operation costs. The number of such parameters is then usually much greater than four.

To determine if such fine control over costs is really necessary and justified by the available training data, we implement a *tied* form of both Viterbi and stochastic edit distance that reduces the parameter set to the four given above. Our Viterbi-tied form may then be viewed as a generalized Levenshtein-style prescription. This tying is easily implemented by simply summing over each category of operations during the maximization step.

Thus far we have defined five distance functions: Levenshtein, Viterbi distance (untied), Viterbi distance (tied), stochastic distance (untied), stochastic distance (tied). The final two: Viterbi distance (mixture), and stochastic distance (mixture) are combinations as described and motivated below.

As transducers are simply probability functions and we may form finite mixtures in the usual way. Such combinations are surprisingly effective since the $-\log$ cost of selecting a component is usually very small compared to the total $-\log$ of the mixture's value. For this reason they operate as a stochastic *or gate* of sorts. That is, the probability is high if that of any component is.

One use of mixtures is to combine models of increasing complexity as a strategy to combat overtraining. We implement a particular simple case in which our tied and untied model forms are combined by a uniform fixed mixture.

Another richer use of mixtures uses them to capture possible modality in the process that generates surface phonetic forms from lexical forms. Different speaker types might well exhibit very different transductive characteristics and a stronger overall model will result if they are separately modeled and combined as a mixture. Different transducers might also be trained for very different corpa and then combined in a mixture. There are many other possibilities and the general approach represents an interesting area for future work. In our experiments the mixing coefficients are fixed and uniform, but they might have been optimized using withheld training data [FJM80].

3.4.3 Learning-Recognition Strategies

During training we do not know *a priori* which lexical phonetic form gave rise to the surface form observed. A feature of our *hidden stochastic mixture* approach is that all possibilities are considered, each reinforced in proportion to the model's *a posteriori* estimate that it generated the observation. During recognition, contributions from all possibilities are combined within a proper probabilistic framework.

A simpler *ad hoc* approach reinforces all possibilities equally during learning. During recognition a simple *nearest neighbor* approach is taken. That is, the least distant lexical phonetic form determines the decision.

All experiments were performed for both strategies to support comparison.

3.5 Experimental Results

We begin by presenting results for our *hidden stochastic mixture* approach to the problem – our preferred solution. Results for the *ad hoc* approach are presented last. Table 3.5 summarizes these results. The error rates shown are the best observed during training. That is, as though an oracle told us when to stop performing EM iterations.

	Leven-shtein	Stochastic Distance			Viterbi Distance		
		Tied	Untied	Mixed	Tied	Untied	Mixed
E1	48.04	18.96	15.47	16.11	18.88	**15.44**	16.11
E2	33.00	18.82	15.30	15.98	18.75	**15.26**	15.99
E3	61.87	24.01	**20.35**	21.77	23.98	20.36	21.76
E4	56.35	21.60	**15.98**	18.92	21.59	**15.98**	18.91

Table 1: Word error rate for seven string distance functions in experiments E1-E4.

Notice that the transduction distances exhibit error rates less than one third that of Levenshtein distance. As we anticipated the Viterbi and stochastic edit distance behave similarly. The untied model is superior in all cases.

The graphs that follow give performance throughout training. The initial performance of transduction distance is poor because training starts with a uniform model.

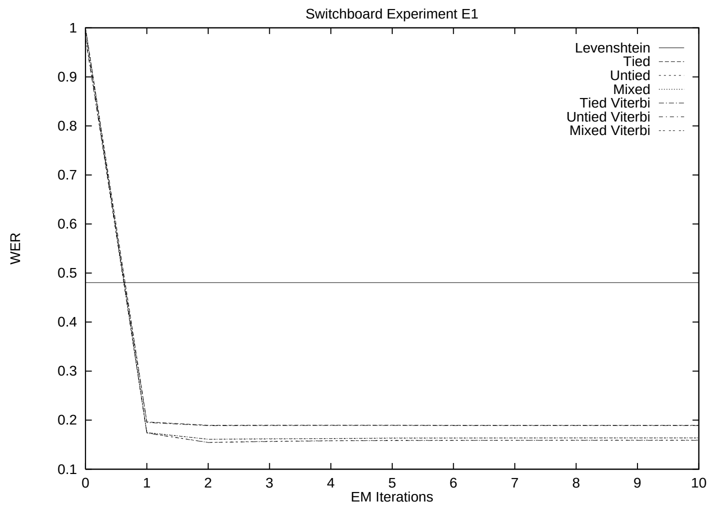

Figure 3: Experiment E1: Word error rate vs. EM iterations for seven models using the full Switchboard lexicon. Improvement is rapid. After only one EM iteration, all transduction distances have less than half the error rate of Levenshtein distance. After two EM iterations, the untied and mixed models have less than one third the Levenshtein error rate.

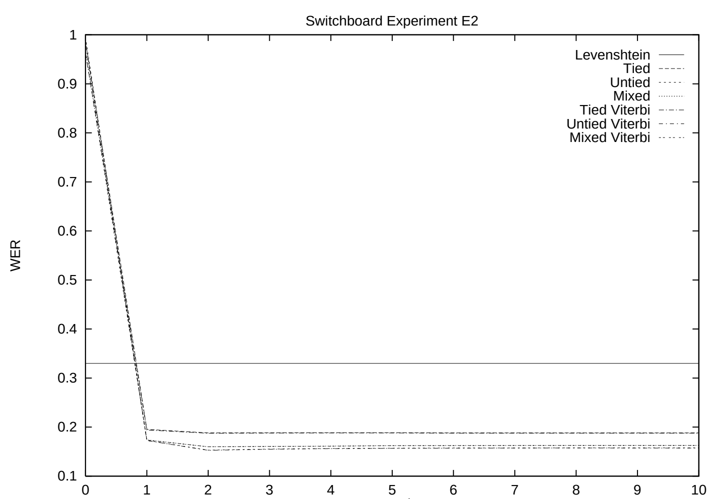

Figure 4: Experiment E2: Word error rate vs. EM iterations for seven models using the subset of the Switchboard lexicon whose words are present in the ICSI transcript. The untied and mixed models have less than half the error rate of Levenshtein distance.

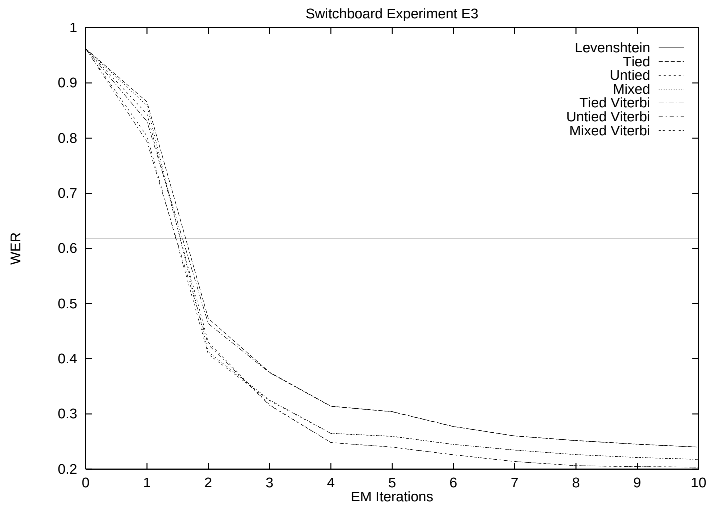

Figure 5: Experiment E3: Word error rate vs. EM iterations for seven models using a lexicon derived from the ICSI training transcript. The untied models have less than one third the error rate of Levenshtein distance. The three lines that are apparent during later EM iterations correspond to the untied, mixed, and tied models, respectively. The performance of all transduction distances is continuing to improve at 10 EM iterations.

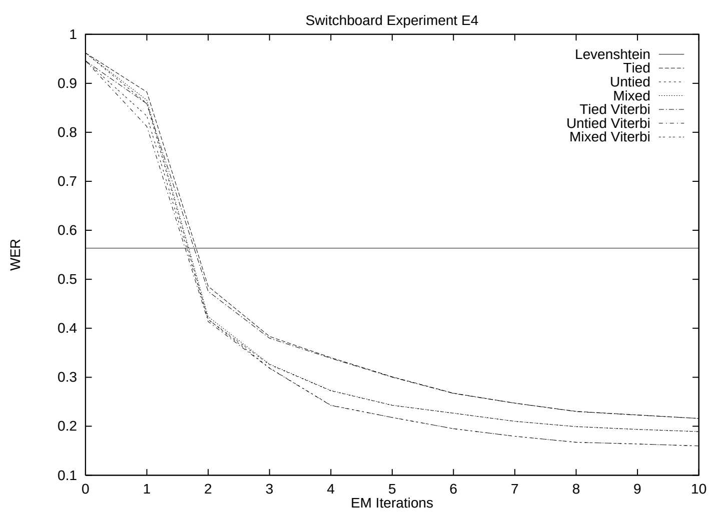

Figure 6: Experiment E4: Word error rate vs. EM iterations for seven models using a lexicon derived from the entire ICSI transcript, including both training and testing portions. The untied and mixed models have less than one third the error rate of Levenshtein distance. The three lines visible during later EM iterations correspond to the untied, mixed, and tied models, respectively. The performance of all transduction distances is continuing to improve at 10 EM iterations.

We now turn to results from our *ad hoc* approach described earlier. It is apparent that trained transduction distances perform poorly relative to simple Levenshtein distance – especially in settings E3 and E4. None of the transduction distances is significantly better than the untrained Levenshtein distance in this approach.

Despite its simplicity this approach is conceptually unsatisfactory because it frequently trains the transducer with unrelated pairs, diluting the effectiveness of learning. Indeed the performance of the trained edit distances is not very different from that of Levenshtein distance. Our results make clear that the added complexity of our preferred *stochastic mixture* approach is not just conceptually superior, but is empirically warranted.

	Leven-	Stochastic Distance			Viterbi Distance		
	shtein	Tied	Untied	Mixed	Tied	Untied	Mixed
E1	48.04	48.39	46.78	46.95	48.41	**46.75**	46.91
E2	33.00	33.51	**31.54**	31.82	33.68	31.55	31.81
E3	61.87	62.80	61.89	62.31	62.89	**61.86**	62.26
E4	**56.35**	**56.35**	56.73	56.36	**56.35**	56.73	**56.35**

Table 2: *Ad hoc approach:* Word error rate for seven string distance functions in experiments E1-E4.

The graphs that follow show performance during training.

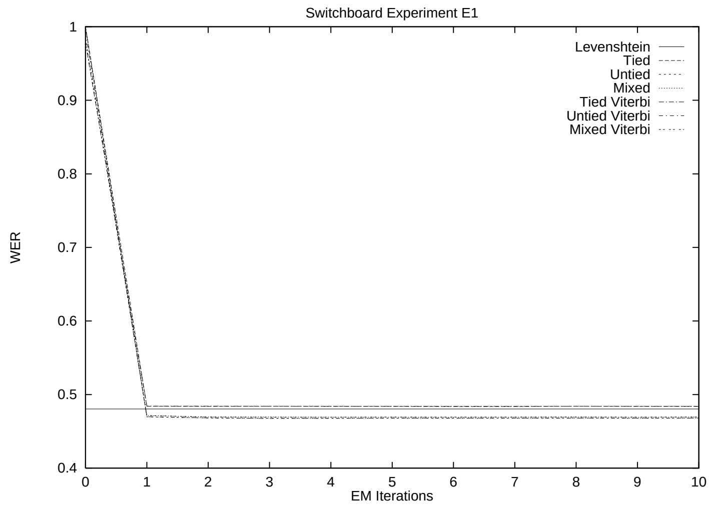

Figure 7: Experiment E1': Word error rate vs. EM iterations for seven models using the full Switchboard lexicon. Both untied and mixed models perform slightly better than untrained Levenshtein distance, while the tied models perform slightly worse.

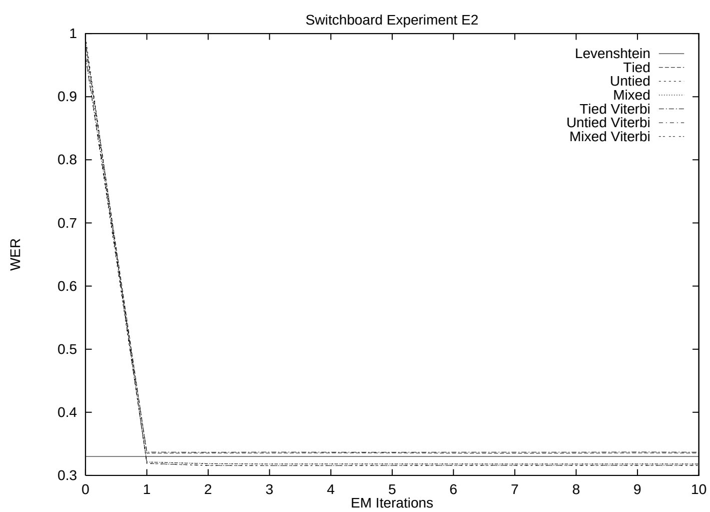

Figure 8: Experiment E2': Word error rate vs. EM iterations for seven models using the subset of the Switchboard lexicon whose words are present in the ICSI corpus. Both untied and mixed models perform slightly better than untrained Levenshtein distance, while the tied models perform slightly worse.

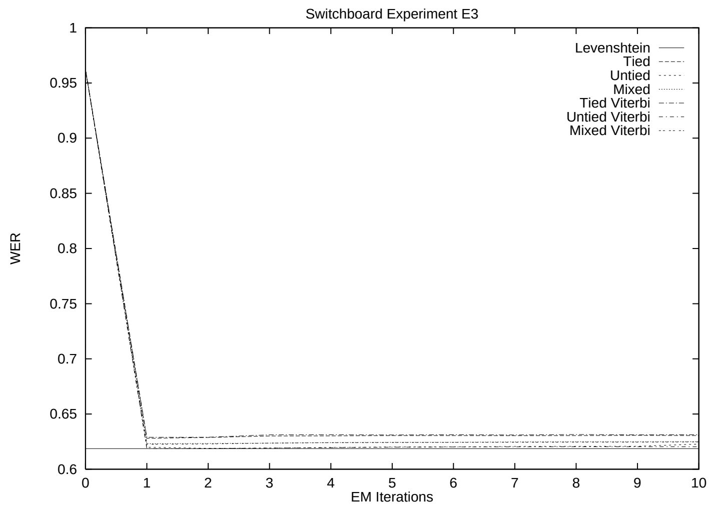

Figure 9: Experiment E3': Word error rate vs. EM iterations for seven models using a lexicon derived from the ICSI training corpus. No model performs better than untrained Levenshtein distance.

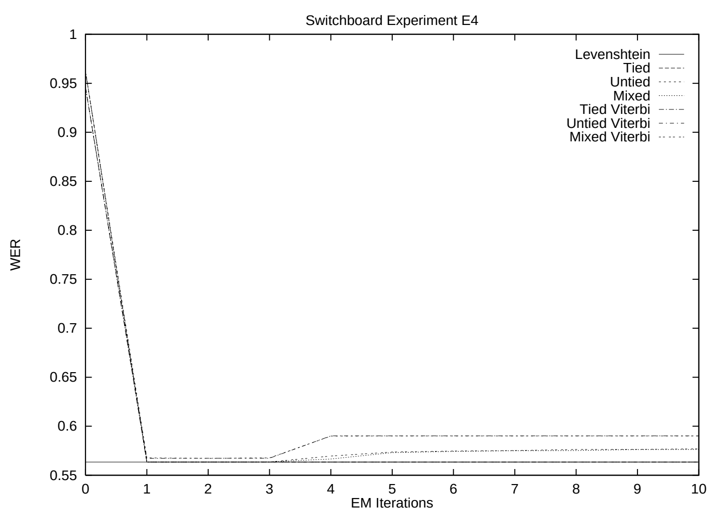

Figure 10: Experiment E4': Word error rate vs. EM iterations for seven models using a lexicon derived from the entire ICSI corpus, including both training and testing portions.No model performs better than untrained Levenshtein distance, although the tied models equal its performance. Additional training decreases performance for the untied and mixed models.

3.6 Future Work

An immediate opportunity exists to improve performance by adding a higher order language model to the system. Mixtures of untied transducers also may help given additional training data.

But our positive results for experiment E3 suggest perhaps the most interesting next step: replace the manual transcript labeling step, with an automatic process – even if the phonemes it recognizes are somewhat noisy and do not correspond perfectly with standard categories. Then enhance the transductive model to include memory, that context sensitivity.

Even this is just one step towards a yet more interesting objective: demonstrating that one may bridge the entire $S \to T$ gap within the framework of stochastic transduction, learning if necessary one or more hidden languages analogous to the artificial phonetic language invented by human linguists.

3.7 Postscript

In section 3.2 we made the simplifying assumption that $p(s|\ell)$ is optimized by instead optimizing the joint transduction probability $p(s, \ell)$. Moreover we remarked that our experiments did not in fact exactly implement the framework described.

This section clarifies these issues by providing an alternative formulation for which no such assumption is necessary. Its implementation and evaluation is, however, left to future work.

The model derivation of section 3.2 may be altered so that the joint probability $p(s, \ell, w)$ is factored instead as $p(s, \ell)p(w|s, \ell) = p(s, \ell)p(w|\ell)$ resulting in the revised model and decision rule:

$$\begin{aligned} \hat{w} &= \text{argmax}_w \quad p(w|s) \\ &= \text{argmax}_w \quad p(w, s)/p(s) \\ &= \text{argmax}_w \quad p(w, s) \end{aligned}$$

$$= \operatorname{argmax}_w \sum_{\ell \in L(w)} p(s, \ell, w)$$
$$= \operatorname{argmax}_w \sum_{\ell \in L(w)} p(s, \ell) p(w|\ell)$$

The virtue of this alteration is that the joint transduction probability $p(s, \ell)$ is now used directly.

Another issue surrounds this transduction probability. As we've implemented it, it does not directly correspond to a generative model for finite strings. But as discussed in section 2.5.1 of chapter 2, it is possible to compute honest probabilities that are conditioned on string lengths. Combining such a model with one for the lengths leads to the generative model required to neaten the conceptual framework of this chapter.

Chapter 4

A Modeling Assembly Language

We have seen that the DAG-based FGM perspective can be used to express many models. This chapter describes certain aspects of the design of a model *assembly language* `MODL` based on the FGM framework – and represents an extended abstract describing our implementation efforts in progress. In our view, hidden Markov models, stochastic context free grammars, Bayes nets, stochastic transducers, mixture models, and many others are best thought of and perhaps implemented as high level languages with which one can succinctly specify members of a particular model class. The computations ultimately performed may all be expressed as FGM operations suggesting that one should first *compile* these high-level specifications into FGM form, where they can be executed by a single computational engine. This approach sweeps many technical details such as extended numerical range requirements and the intricacies of recursively specified models, down to the assembly language level. Another benefit of our approach is that alternative optimization criteria such as MDL are conveniently implemented without major impact on the high level design.

4.1 General Architecture

The kernel of the `MODL` system is ANSI C-based and includes a subroutine-level programmer's API. Parameter passing is particularly simple so that interfaces to other languages are straightforward.

Because of the subroutine-level API's simplicity, an interpreted ASCII scripting language is easily wrapped around it to further simplify its use and broaden the set of potential users. Such an ASCII model specification might then be generated by any language, or perhaps by specialized graphical model design tools – and represents a highly portable model description.

Additional APIs might be developed for popular engineering/mathematics support programs such as *Mathematica* and *Matlab*[1].

The system is object oriented, and even in the subroutine API, objects are named using character strings. Objects can be written or read from a file, and a portable ASCII representation is always used.

The components of a complex recursive FGM that includes observation models that are themselves FGMs, may be specified in an arbitrary order. At specification stage only names are recorded for each object referred to. Once the set of required objects has been specified, the top-level FGM is *assembled* into a working model. At this time all references are checked with respect to existence and type, and references by name are converted to internal pointers. The resulting object is then a functioning model to which data may be presented.

Additional run-time checks exist to help ensure model correctness. The error reporting philosophy is that errors should be described as thoroughly as possible, but no attempt is made to recover, i.e. the program terminates. This is accomplished using a uniform approach to error reporting that identifies several levels of context to aid in diagnosis. In the case of the scripting language, this context begins with the input line number, includes the software API subroutine involved, and may include more detailed internal information.

Several software development steps are taken to help ensure the quality and portability of the resulting system. First, it is based at the lowest level on memory allocation primitives and other modules of *libpa* – the "library of practical abstractions" [RY97b].

[1]Mathematica is a product of Wolfram Research, and Matlab is a product of the Math Works Inc.

Because of this, run-time analyzers such as *Purify*[2] are particular effective identifiers of common implementation problems. At a higher level, the subroutine library is built on top of a single object infrastructure developed for `MODL` . The library is compiled and tested on a wide variety of systems including Sun Solaris, Linux i586, DEC Unix[3], and if possible on SGI Unix and Microsoft Windows NT. A library test program and test scripts are included with the planned distribution. In addition, a quick system self-check is provided at the subroutine API level to give users additional confidence that `MODL` is operating properly on their system. Finally, two versions of the library and scripting language interpreter are made available. The first has a great deal of "asserts" enabled in the `MODL` system and libpa libraries as well. The second disables them resulting in a considerable performance increase.

The system may be extended over time to add new primitive model types. These must be implemented in C on top of the system's object infrastructure and subject to other interface requirements.

The rest of this chapter discusses the structure of the library and its API. The scripting language remains to be designed but we anticipate that it will consist of a straightforward encapsulization of the subroutine API.

4.2 The Object System

Objects are created within a specified *memory set* (MSET). They may be destroyed individually, or for convenience as part of the destruction of an entire MSET. The caller may create a hierarchy of MSETs starting from a single root. The destruction of any element destroys all children as well. The MSET approach is taken to simplify the generation of programs free of memory leaks. An alternative considered was the implementation of a specialized garbage collector made possible through the use of intelligent pointers.

Objects have an ASCII name within a single global name space. Some objects support a copy operation and most support file read and write operations that allow

[2] Purify is a product of Pure Software Inc.
[3] Unix is a registered trademark of AT&T

the object to be saved in a portable ASCII form.

Observation vectors are not objects within the system, but rather are simply arrays of `double`; each element of which has one of three attributes: ordinary, unknown, or out-of-bounds. A component of known value is of type "ordinary". Components labeled "unknown" cause FGM evaluation to attempt to marginalize with respect to them. It is also possible to infer the value these positions with the result written back into the corresponding positions. Hidden Markov and many other models correspond to a DAG with depth dependent on the length of the observed time series. Using `MODL` one creates a DAG deep enough to represent the longest expected series, and then uses the "out-of-bounds" attribute to mark positions beyond the end of the series under consideration. This is a general mechanism that applies to other variable length models as well. Discrete observations are represented using integer values. Run time range, value, and dimensionality checking is performed by the observation models and selection functions that deal with observations.

In addition to libpa, the system's utility substrate includes error reporting routines, common support for input and output of basic data types, and a data structure hashing facility used for self-checking. It also includes several as of yet unreleased libpa modules providing linear algebraic, normal density related, and dictionary functions.

There are only seven object types in the system:

- FGM Specification
- Observation Model
- Observation Argument
- Choice Model
- Choice Argument
- Selection Function
- Selection Argument

4.2.1 FGM Specifications

This object type is best thought of as an FGM in symbolic form. That is, the references to things like observation models attached to its edges, consists of object names. The entities referred to need not yet exist when the specification is made.

The creator returns an empty specification that includes an empty DAG. A procedure is provided to add new vertices to the DAG. Vertices are identified with natural numbers. Zero designates the source. The enumeration need not be dense, but space is linear in the maximum value used. When a vertex is created a choice model is identified along with an argument (see discussion of arguments below).

Once nodes i and j exist, a procedure may be called to connect them using a directed edge. The corresponding choice model index must be specified along with a selection function and argument, and observation function and argument. An edge name is also assigned for use in Viterbi decoding.

The object supports topological sorting of its vertices at which time any cycles or certain other structural problems are detected. The user never invokes this sort however. It is called by the creator (assembler) of an observation model of type FGM. File I/O is supported for FGM specifications, but not for their assembled form. That is, reassembly is required each time an FGM is read from a file.

4.2.2 Observation Models and Arguments

The observation model object class is in some sense the heart of the system. Presented with an observation, members evaluate its probability[4] or support Baum-Welch/EM iteration. Other operations are described below.

The initial release of `MODL` supports three types of observation models: simple discrete probability functions, multivariate normal densities, and FGMs. A creator is provided for each. The FGM creator accepts as input an FGM specification, and produces an FGM observation model object. In this *assembly* process, an entirely new

[4]Strictly we mean the value of a nonnegative functional, since FGMs need not be viewed as stochastic models. Nevertheless, since so many applications are probabalistic, we will use the term "probability."

data structure is created from the specification. All symbolic references become simple pointers. An important value added by this step has to do with recursive FGMs. Such a system is organized into form that can support computation in time linear in the overall edge count. This amounts to properly ordering all required computations down to the primitive model level. Equivalent invocations of observation models are detected and only performed once. Here "equivalent" refers to the combination of selection and observation. What is considered is the net selection, not the sequence of steps. So the result of several layers of selection by projection can be identical to a single selection. This capability is important when implementing models such as stochastic context free grammars.

In the initial `MODL` release the observation argument object holds a boolean value that selects between ordinary model instances, and those required to implement MDL reestimation (see chapter 2). When the argument is true, the observation model assumes a value based on the observation. When false, its value represents a penalty term corresponding to the current value of the model's parameters. These penalty terms form in the simplest case a linear chain from the DAG's root and represent multiplication of the standard observation probability by the appropriate penalty term.

A Viterbi decode procedure may be called to locate a maximum weight (most likely) source-sink path through the DAG. A sequence of edge names is returned. An inference call is provided that replaces all unknown observation values with their most likely value given the known observation components. Evaluation attempts to marginalize with respect to unknown components. As a result one can easily compute all conditional probabilities as well. Because of the FGM framework's generality, these calls may not make sense in all settings, and for others may not return an exact answer.

During the evaluation α-pass, encountering an out-of-bounds observation adds a temporary zero weight edge directly to the sink. The β-pass then uses this list in its processing. In this way only the model edges relevant to the available observation data are processed.

The parameters of an observation model need not be reestimated during a maximization step, and a special mutator call provides this option.

4.2.3 Choice Models and Arguments

A choice model object is associated with each vertex in an FGM and gives the probability of choosing each outgoing edge. A choice model is essentially an observation model in which the observation consists of an edge index. Only the simple choice model corresponding to a discrete observation model is provided in the initial release of `MODL` . In this release the choice model argument object operates as for observation models to implement MDL.

4.2.4 Selection Functions and Arguments

A selection function object represents a family of selection functions parameterized by a given selection function argument object. The initial release of `MODL` supports a single family: coordinate-wise projectors. The argument specifies which observation tuple positions are to be included, and in what order.

Selection function objects support argument composition. For projectors this combines two projections to yield an argument object representing their combination. Comparison of argument objects is also supported. In this way the detection of equivalent selection functions is accomplished in a highly general way.

Chapter 5

Emphasis Reparameterization

This chapter[1] illuminates certain fundamental aspects of the nature of normal (Gaussian) mixtures. Thinking of each mixture component as a *class*, we focus on the corresponding *a posteriori* class probability functions. It is shown that the relationship between these functions and the mixture's parameters, is highly degenerate – and that the precise nature of this degeneracy leads to somewhat unusual and counter-intuitive behavior. Even complete knowledge of a mixture's *a posteriori* class behavior, reveals essentially nothing of its absolute nature, i.e. mean locations and covariance norms. Consequently a mixture whose means are located in a small ball anywhere in space, can *project* arbitrary class structure everywhere in space.

The well-known expectation maximization (EM) algorithm for Maximum Likelihood (ML) optimization may be thought of as a reparameterization of the problem in which the search takes place over the space of sample point weights. Motivated by EM we characterize the expressive power of similar reparameterizations, where the objective is instead to maximize the *a posteriori* likelihood of a labeled training set. This is relevant to, and a generalization of a common heuristic in machine learning in which one increases the weight of a mistake in order to improve classification accuracy. We prove that EM-style reparameterization is not capable of expressing arbitrary *a posteriori* behavior, and is therefore incapable of expressing some solutions.

[1] This chapter first appeared as a technical report [RY96c]

However a slightly different reparameterization is presented which is almost always fully expressive – a fact proven by exploiting the degeneracy described above.

5.1 Introduction

A *normal mixture* is a finite stochastic combination of multivariate normal (Gaussian) densities. That is, a probability density function p on $\mathbb{R}^d$ of the form:

$$p(x) = \sum_{i=1}^{k} m_i N_{\Sigma_i,\mu_i}(x)$$

where $m_1, \ldots, m_k \geq 0$ with $\sum_{i=1}^{k} = 1$, and $N_{\Sigma_i,\mu_i}(x)$ denotes the normal density with mean μ_i and covariance Σ_i. Each constituent normal density is referred to as a *component* of the mixture, and $m_1, \ldots, m_k$ are the *mixing coefficients*.

Normal mixtures have proven useful in several areas including pattern recognition [DH73] and speech recognition [RJLS85, Bro87, HAJ90] – along with vector quantization and many others. Each component of the mixture is thought of as corresponding to some *class* denoted $\omega_1, \ldots, \omega_k$, and the *a posteriori* class probabilities $p(\omega_1|x), \ldots, p(\omega_k|x)$ are used to effect classification.

This chapter considers certain aspects of the relationship between the *two faces* of a normal mixture, i.e. the mixture itself versus the *a posteriori* class functions it induces. Section 5.2 shows that this relationship is not one-to-one and exposes the considerable degeneracy wherein many distinct normal mixtures induce the same class functions.

As a positive result of this degeneracy one can search the entire space of class functions without considering all possible mixtures. This is relevant to problems whose solution depends only on the mixture's second face, i.e. its induced class functions. An example of such a problem is that of maximizing the *a posteriori* class probability of a labeled training set. Here the objective is to predict the correct labels, not model the observation vectors themselves.

Section 5.3 shows that such search problems may almost always be carried out using reparameterizations we refer to as *emphasis* methods that are motivated by the simple and intuitive expectation maximization (EM) algorithm [BE67, DLR77, RW84] for unsupervised maximum likelihood (ML) parameter estimation. The reparameterized problem cannot express an arbitrary mixture but nevertheless, because of degeneracy, can express a mixture that induces optimal class functions with respect to the prediction of training labels. To clarify these problems and their relationship to the notion of emphasis we begin with review.

Given an unlabeled training set the EM algorithm iteratively improves (in the ML sense) any mixture – unless it is already locally optimal. The improved means and covariances are given by a simple weighted averaging computation. Given data values $s_1, \ldots, s_n$ and a normal mixture M, one begins by computing $p(\omega_i|s_j, M), \forall 1 \leq i \leq k, 1 \leq j \leq n$. Conceptually the result is a $k \times n$ table of nonnegative values. The rows of this table are then normalized so that their sum is one. Each row then corresponds to a convex combination of the samples. The entries along a row are thought of as *weights* attributed to the sample. Each improved mean vector μ_i is merely the corresponding weighted average (convex combination) of the sample vectors. Each improved covariance is the sample convex combination of outer products $(s_j - \mu_i)(s_j - \mu_i)^t$. The improved mixing parameters are obtained by normalizing the vector consisting of the table row weights prior to their normalization. This process is repeated in an elegant cycle, i.e. each table induces a mixture that gives rise to a new table, and so on.

We view this table as reparameterizing the problem of searching for an ML mixture. The point is that one might instead have used a direct parameterization consisting of the individual means, covariance matrices, and mixing parameters – and employed standard constrained optimization methods from numerical analysis. We loosely say that such a table driven scheme is an *emphasis reparameterization.*

Only a locally optimal solution is found through EM but it is important to realize that the global optimum is a stationary point of the iteration. This means that it can be expressed via emphasis, i.e. that there exists a table which induces the globally optimal mixture. This is interesting because not all mixtures can be induced from the table. In particular, the induced means must lie within the convex hull of the sample set, and the covariances are similarly constrained.

The optimization problem corresponding to the other, *a posteriori* face of the mixture seeks to maximize the *a posteriori* likelihood of a labeled training set. That is, to find a model M which maximizes:

$$\prod_{i=1}^{n} p(\omega(s_i)|s_i, M)$$

where $\omega(s_i)$ is the class label associated with sample s_i. This is sometimes called the *maximum mutual information* (MMI) criterion in the speech recognition literature [Bro87]. We remark that more general forms of this problem may be stated in which the labels are themselves probabilities. Sections 5.3 and 5.4 consider the question of whether emphasis-like reparameterizations may be used to attack the more general form of this problem.

This is relevant to a common heuristic in machine learning in which one somehow increases the weight of (emphasizes) a mistake in order to improve classification accuracy. It is natural to wonder whether such approaches are even capable of expressing the solution. In the case of normal mixtures this chapter illuminates the issue considerably. By theorem 6 one can almost always exactly match the *a posteriori* behavior of an arbitrary mixture by an emphasis method that slightly modifies the EM approach. Section 5.4 exposes the limitations of EM-style emphasis and shows that it is not in this sense universal.

These results depend on a detailed understanding of the degenerate relationship between a normal mixture and its induced class functions. A convenient visualization of these *a posteriori* class functions, focuses on *decision boundaries*, i.e. the surfaces along which classification is ambiguous.

Imagery like ours in figure 11, and pages 28–31 of [DH73], suggest an intuitive relationship between mixture component locations, and the resulting *a posteriori* class structure and decision surfaces. One imagines each mean to be asserting ownership over some volume of space surrounding it. This view is however far from the truth and the *a posteriori* class structures arising from normal mixtures are far more interesting than these simple examples suggest.

Theorem 5 reveals that the *a posteriori* class behavior of normal mixtures is far

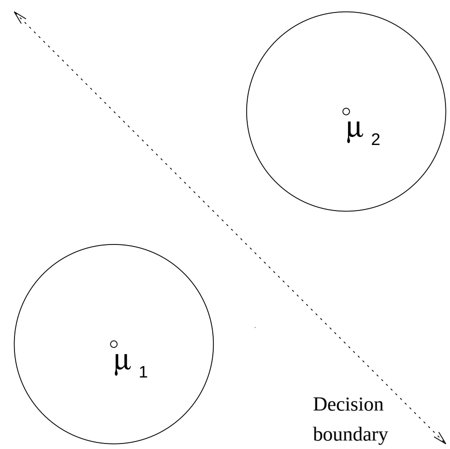

Figure 11: A even mixture of two normal densities in $\mathbb{R}^2$, each with identity covariance. The two classes are separated by a linear decision boundary.

stranger and counter-intuitive than is generally understood. It shows that knowledge of a mixture's *a posteriori* class structure (which may be thought of as decision surfaces), tells us essentially nothing about the absolute location of the mixture's means – or the absolute nature of its covariances. One can easily imagine a normal mixture, and picture its corresponding class structure. Now if one is instead given only this class structure, the theorem says that infinitely many normal mixtures are consistent with it – and in particular that the means of such a mixture can be located within an arbitrarily small ball, located anywhere in space.

Despite the popularity of normal mixtures for classification problems, the precise nature of this highly degenerate relationship between class functions and mixtures does not seem to have been considered in the literature. Part of the reason may be that the simple view in which means dominate some portion of the space around them is exactly the case when all covariances have identical determinant, and uniform mixing coefficients are used. The most common example of this setting consists of nearest-neighbor Euclidean clustering which corresponds is the identity covariance case. In practice, if the determinants are somewhat comparable, and the mixing coefficients nearly uniform, the simple view is almost always valid. But in many cases, such as when estimating mixture parameters, no such constraint is imposed, and the means, covariance matrices, and mixing coefficients are free to assume any values. Here the full range of strange behavior may be expected.

5.2 *A Posteriori* Degeneracy

Let $s_1, \ldots, s_n \in \mathbb{R}^d$ be a set of observations. Any set of positive weights $\gamma_1, \ldots, \gamma_n$, gives rise in a natural way to a normal density. That is, the normal density having mean:

$$\mu = \frac{1}{\sum_{i=1}^n \gamma_i} \sum_{i=1}^n \gamma_i s_i \tag{9}$$

and covariance:

$$\Sigma = \frac{1}{\sum_{i=1}^n \gamma_i} \sum_{i=1}^n \gamma_i (s_i - \mu)(s_i - \mu)^t \tag{10}$$

which may be thought of as the weighted maximum likelihood (ML) model corresponding to the observations. Given k sets of weights, as many normal densities arise. Adding k additional positive *mixing* weights $w_1, \ldots, w_k$ serves to specify a mixture of these k densities, with the probability of the ith component given by $w_i / \sum_{i=1}^k w_i$. Thus a total of $nk + k$ weights induce a normal mixture, and we loosely refer to them as *emphasis parameters*.

The well known expectation maximization (EM) method may be viewed as a process which accepts as input a normal mixture, and outputs an emphasis parameter set – from which an improved normal mixture arises. This cycle continues as the algorithm climbs towards a local maximum. The global maximum is a stationary point of this iteration.

From our viewpoint, what is interesting here is that it is possible to search for an optimal normal mixture, by searching over the space of values for the $nk + k$ emphasis parameters – rather than using the canonical parameterization consisting of the unknown mean vectors, covariance matrices, and mixing coefficients.

Now the emphasis parameterization above cannot express an arbitrary mixture. This is because each induced mean is a convex combination of the observations, and must therefore lie within their convex hull. An immediate corollary to our EM comments above is then that the means of an optimal mixture lie within the convex hull of the observation set. Similarly, not all covariance matrices can be generated by emphasis, and the covariances of an optimal mixture are similarly constrained.

Expectation maximization strives to maximize the probability of the observation set. If instead the function to be optimized depends only on the *a posteriori* behavior of the mixture, it is natural to ask:

> When can emphasis of an observation set induce a mixture which is class equivalent to an arbitrary one?

Obviously the observation set must satisfy certain orthodoxy conditions such as being of sufficient size and possessing in some sense enough independence. But beyond these issues the fact that many mean vectors and covariance matrices are not expressible by emphasis would seem to present a much larger obstacle.

In this section we show that the the relationship between a normal mixture and its class functions is highly degenerate – so much so that constrained expressiveness of emphasis does immediately rule out an affirmative answer to our question. Examples and discussion follow the main mathematical development. The section ends with a technical convergence-rate result that is needed to establish the next sections' reparameterization theorem.

Definition 5 *A finite mixture is a probability function on a measure space $\mathcal{X}$, arising from k conditional probability functions (components) as follows:*

$$p(x) = \sum_{i=1}^{k} p(x, \omega_i) = \sum_{i=1}^{k} p(x|\omega_i)p(\omega_i)$$

where the constants $\{p(\omega_i)\}$ are referred to as the mixing coefficients*. Any such mixture induces k* a posteriori *class functions:*

$$p(\omega_i|x) = \frac{p(x|\omega_i)p(\omega_i)}{p(x)}$$

Two finite mixtures are class-equivalent if they induce the same class functions.

The class functions are not independent since they are constrained by: $\sum_k p(\omega_i|x) = 1$. We begin with a simple proposition which allows us in what follows, to focus on ratios of conditional probabilities, and ignore constant factors. This will be important since the nonconstant portion of the ratio of two normal densities is a rather simple object.

Proposition 4 *Let $p(x)$ be a k-component mixture, and $p'(x|\omega_1), \ldots, p'(x|\omega_k)$ be a collection of strictly positive conditional probability functions. If for some j, there exist constants $C_{i,j}$ such that $\forall i \neq j$ and $x \in \mathcal{X}$:*

$$\frac{p(x|\omega_i)}{p(x|\omega_j)} = C_{i,j} \cdot \frac{p'(x|\omega_i)}{p'(x|\omega_j)} \tag{11}$$

then there exist mixing coefficients $p'(\omega_1), \ldots, p'(\omega_k)$ such that the resulting mixture $p'(x)$ is class-equivalent to $p(x)$.

proof: We begin by giving a formula for mixing coefficients $p'(\omega_1), \ldots, p'(\omega_1)$, such that:

$$\underbrace{C_{i,j} \frac{p(\omega_i)}{p(\omega_j)}}_{k_{i,j}} = \frac{p'(\omega_i)}{p'(\omega_j)} \tag{12}$$

Relaxing for the moment the restriction $\sum_{\ell=1}^{k} p(\omega_\ell) = 1$, observe that if $p(\omega_j)$ were 1, then setting $p(\omega_i) = k_{i,j}, i \neq j$, would satisfy the equation. Normalization then yields a solution which does sum to 1:

$$\begin{aligned} p'(\omega_j) &= \frac{1}{1 + \sum_{\ell \neq j} k_{\ell,j}} \\ p'(\omega_i) &= \frac{k_{i,j}}{1 + \sum_{\ell \neq j} k_{\ell,j}} \quad i \neq j \end{aligned}$$

Multiplying each side of Eq. 12 by the corresponding side of Eq. 11 yields:

$$\frac{p(x|\omega_i)p(\omega_i)}{p(x|\omega_j)p(\omega_j)} = \frac{p'(x|\omega_i)p'(\omega_i)}{p'(x|\omega_j)p'(\omega_j)} \quad \forall i \neq j, x \in \mathcal{X} \tag{13}$$

The approach above works so long as $p(\omega_j) \neq 0$. If it is zero, then p may be treated as a $k-1$ component mixture, and the proposition follows by induction. Till now j has been fixed and Eq. 13 is established for $i \neq j$. This equation is however easily extended to any pair i, ℓ of indices. Let f_i denote $p(x|\omega_i)p(\omega_i)$ and f' denote $p'(x|\omega_i)p'(\omega_i)$. Then:

$$\frac{f_i}{f_\ell} = \frac{f_i}{f_j} \cdot \frac{f_j}{f_\ell} = \frac{f'_i}{f'_j} \cdot \frac{f'_j}{f'_\ell} = \frac{f'_i}{f'_\ell}$$

Now we may write:

$$p(\omega_i|x) = \frac{f_i}{\sum_{\ell=1}^k f_\ell} = \frac{1}{1 + \sum_{\ell \neq i} f_\ell / f_i}$$

and a corresponding expression for $p'(\omega_i|x)$ in terms of $\{f'_\ell\}$. We have already seen that all ratios $f_i/f_\ell = f'_i/f'_\ell$, so $\sum_{\ell \neq i} f_\ell/f_i = \sum_{\ell \neq i} f'_\ell/f'_i$ whence $p(\omega_i|x) = p'(\omega_i|x)$ and we are done. $\square$

We now specialize our discussion to mixtures whose components are normal densities.

Definition 6 *A d-dimensional k-component normal mixture, is a finite mixture of multivariate normal densities:*

$$N_{\mu,\Sigma}(x) \triangleq \frac{1}{(2\pi)^{d/2}\Sigma^{1/2}} \cdot e^{-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)} \tag{14}$$

The mixture's parameters are then $\Phi = \{\Sigma_1, \ldots, \Sigma_k, \mu_1, \ldots, \mu_k, p(\omega_1), \ldots, p(\omega_k)\}$.

The following theorem exposes the large degeneracy in the relationship between normal mixtures, and their induced class functions.

Theorem 5 *Let p be a d-dimensional normal mixture with k components. For any $x \in \mathbb{R}^d$ and $\epsilon > 0$, there exists a d-dimensional k-component normal mixture p', such that:*

1. $\|\mu' - x\| < \epsilon$

2. $\|\Sigma'\|_2 < \epsilon$

3. p' *and* p *are class-equivalent*

where $\|\Sigma'\|_2$ *refers to the Frobenius/Euclidean matrix norm.*

proof: Begin by selecting some distinguished component j. By proposition 4 we have only to produce $\Sigma'_1, \ldots, \Sigma'_k$ and $\mu'_1, \ldots, \mu'_k$ such that the ratio conditions of the proposition are satisfied. Since constant factors do not matter, we ignore several portions of the definition of a normal density Eq. 14. Namely, the leading constant and the constant portion of the exponent once multiplied out. The result is that:

$$N_{\mu,\Sigma}(x) \propto e^{-\frac{1}{2}(x^t\Sigma^{-1}x - 2\mu^t\Sigma^{-1}x)}$$

The proportional ratio condition of proposition 4 then leads immediately to the following necessary and sufficient conditions:

$$\left.\begin{array}{rcl} \Sigma_i^{-1} - \Sigma_j^{-1} & = & \Sigma'^{-1}_i - \Sigma'^{-1}_j \\ \mu_i^t\Sigma_i^{-1} - \mu_j^t\Sigma_j^{-1} & = & \mu'^t_i\Sigma'^{-1}_i - \mu'^t_j\Sigma'^{-1}_j \end{array}\right\} \ \forall i \neq j \tag{15}$$

We set $\mu'_j = x$ and begin by sketching the rest of the proof. The constraints are satisfied by choosing Σ'^{-1}_j so that each of its eigenvalues is *large*. Each resulting Σ'^{-1}_i must then be positive definite and will also have large eigenvalues. Both Σ'_j and Σ'_i then have small norm, satisfying the requirements of the theorem. In this process, it is only Σ'^{-1}_j we are free to choose. Each choice determines the Σ'_i, and μ'_i (since we have already set $\mu'_j = x$). In the limit, as we choose Σ'^{-1}_j with increasingly large eigenvalues, $\|\mu'_i - \mu'_j\| = \|\mu'_i - x\|$ approaches zero whence we can satisfy the theorem's other condition.

Denote by $\overline{\lambda}(A)$ the largest eigenvalue of positive definite matrix A, and by $\underline{\lambda}(A)$ the smallest. For matrices A, B, it then follows easily from the Cauchy-Schwartz inequality that $\underline{\lambda}(A+B) \geq \underline{\lambda}(A) - \overline{\lambda}(B)$, by writing $\|[(A+B)+(-B)]u\| \leq \|(A+B)u\| + \|-Bu\|$ where u denotes any unit length vector. Now the first constraint in Eq. 15 may be written:

$$\Sigma'^{-1}_i = \Sigma'^{-1}_j + (\Sigma^{-1}_i - \Sigma^{-1}_j) \tag{16}$$

and we then have:

$$\underline{\lambda}(\Sigma'^{-1}_i) \geq \underline{\lambda}(\Sigma'^{-1}_j) - \overline{\lambda}(\Sigma^{-1}_i - \Sigma^{-1}_j)$$

The parenthesized term is constant since we are given both Σ^{-1}_i and Σ^{-1}_j, and by subadditivity of symmetric matrix spectral radii, does not exceed their sum $\overline{\lambda}(\Sigma^{-1}_i) + \overline{\lambda}(\Sigma^{-1}_j)$. We can easily choose Σ'^{-1}_j such that $\underline{\lambda}(\Sigma'^{-1}_j)$ is arbitrarily large, e.g. $c \cdot I$ where c is a large positive constant. It follows then that $\underline{\lambda}(\Sigma'^{-1}_i)$ will also be arbitrarily large, ensuring that it is positive definite.

Next recall that the column norms of a matrix A cannot exceed its spectral radius (operate on each member of the canonical basis). In our positive definite setting each is then bounded above by $\overline{\lambda}(A)$, so that $\|A\|_2 \leq \sqrt{d}\overline{\lambda}(A)$. Choosing the smallest eigenvalue of Σ'^{-1}_j and Σ'^{-1}_i to be arbitrarily large, forces $\overline{\lambda}(\Sigma'_j)$ and $\overline{\lambda}(\Sigma'_i)$ to be arbitrarily small – satisfying the theorem's first condition.

We set μ'_j to be equal to x from the statement of the theorem, and the second constraint in Eq. 15 may be rearranged to yield:

$$\mu'_i = (\Sigma'_i \Sigma'^{-1}_j)x + [\Sigma'_i(\Sigma^{-1}_i \mu_i + \Sigma^{-1}_j \mu_j)] \tag{17}$$

By our earlier discussion, we may force Σ'_i to be arbitrarily close to zero – forcing the bracketed term in Eq. 17 to approach zero as well. We next show that the first parenthesized term $\Sigma'_i \Sigma'^{-1}_j$ tends to the identity matrix I as $\underline{\lambda}(\Sigma'^{-1}_j) \to \infty$. This then demonstrates that $\mu'_i \to x$ satisfying the theorem's second condition, and finishes the proof.

It is easy to see that $\Sigma'_i \Sigma'^{-1}_j \to I$ if and only if $(\Sigma'_i \Sigma'^{-1}_j)^{-1} = \Sigma'_j \Sigma'^{-1}_i \to I$. Using Eq. 16 this becomes:

$$I + \Sigma'_j(\Sigma^{-1}_i - \Sigma^{-1}_j) \to I$$

which is clearly the case since $\Sigma'_j \to 0$. $\square$

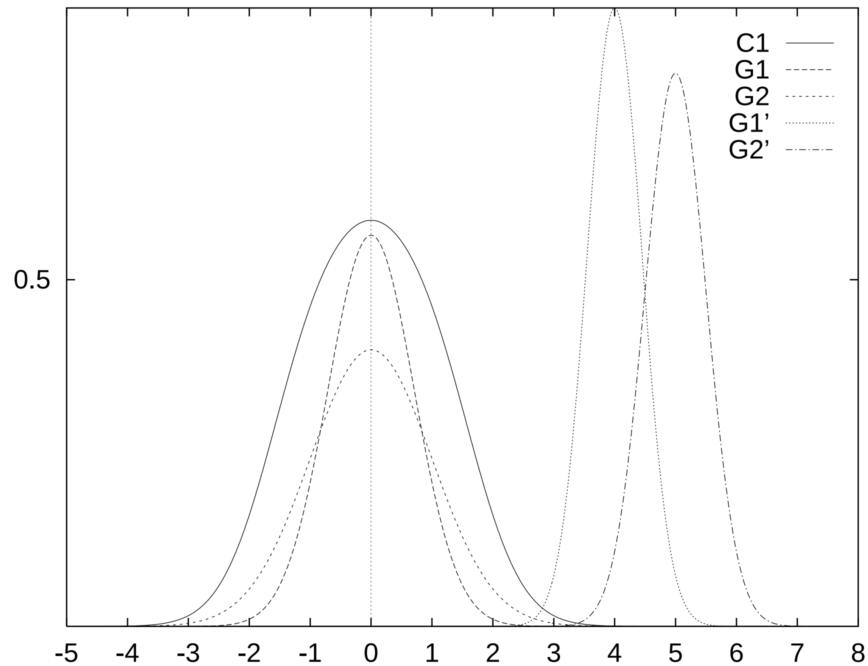

Figure 12: Two ways to induce a single class function. An illustration of the degeneracy in the relationship between normal mixtures, and their induced class functions. Zero mean normal densities G1 and G2 when mixed evenly induce C1, the *a posteriori* probability of G1. Normal densities G1' and G2' located in the right portion of the graph do not have zero means, but may be mixed (in a far from even way) so that C1 results.

To recapitulate, any location for ${\mu'}_j$, and sufficiently large (in the $\underline{\lambda}$ sense) ${\Sigma'}_j^{-1}$, will give rise using the proof's constructions, to values for the other means, covariances, and mixture coefficients, such that a class-equivalent mixture results. The proof goes on to establish that in the limit, everything is confined as required within an ϵ-neighborhood.

Figure 12 provides a simple one dimensional example of the normal mixture to class function degeneracy. The left pair of Gaussians G1,G2 both have zero mean. The taller one, G1 corresponds to the first class and has variance 1/2, while G2 has variance 1 and corresponds to the second class. When mixed evenly, the *a posteriori* probability C1 results. Notice that C1 assumes value 1/2 where G1 crosses G2. The right pair of Gaussians G1',G2' have means at 4 and 5, and variances of 1/5 and 1/4 respectively. An even mixture of them would result in a class function very different from C1. But if very little weight ($\approx 5.74234 \times 10^{-5}$) is placed on G1', its mode drops under the graph of G2', and surprisingly, the induced class function is exactly C1. The parameters of this new mixture follow immediately from the calculations within the proof of theorem 5.

Figure 13 depicts another two class problem, this time in two dimensions. Here various elements of each class are arranged so that they may be perfectly separated by a circular decision boundary. It is clear that we can create such a boundary by modeling each class with a normal density centered at the origin. We may evenly mix these components and set their covariance matrices to be multiples of the identity matrix; chosen so that the two density's intersection is the desired circle. This is in some sense the canonical solution but infinitely many others are possible. To begin with, it is not necessary that both covariance matrices be multiples of the identity. It suffices that their difference be such a multiple. It is not necessary that their means be located at the origin. Given a choice of covariance matrices, and the location of one mean, a location for the other may be computed which gives rise to the same boundary. Moreover, by choosing the covariances to be nearly zero, corresponding to highly peaked densities, the two means may be located within an arbitrary ϵ-neighborhood anywhere in $\mathbb{R}^2$. This illustrates the degeneracy by focusing only on the decision boundary, but theorem 5 shows that the family of class functions may be matched everywhere. The situation in figure 13 was first contrived in an attempt to disprove our *emphasis reparameterization theorem*, the subject to which we next turn. The data points are arranged along only part of the circle so that no convex combination of them, can possibly express the origin. That is, the origin is outside of the convex hull of the dataset. At that time we thought that this would prevent such combinations from leading to a mixture capable of inducing the desired circular decision boundary. The revelation of theorem 5 is that the component means of such a mixture can be located anywhere.

We have seen that all the ${\Sigma'}_i$ converge to ${\Sigma'}_j$ as the latter tends to zero. At the same time, the ${\mu'}_i$ approach ${\mu'}_j$. The rate of convergence is our next topic. In particular

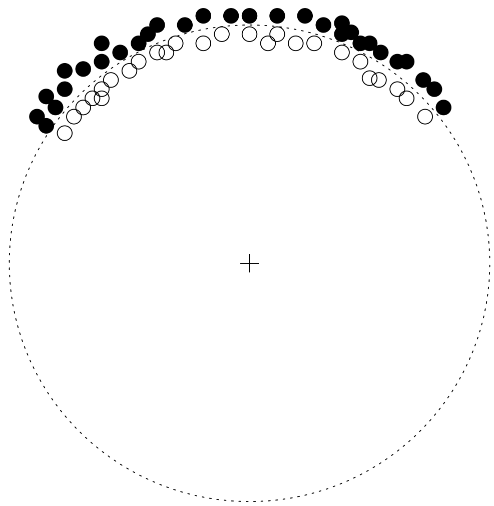

Figure 13: If data points of two colors are imagined to hug the inside and outside of a circular decision boundary, one might think that a Gaussian mixture model capable of separating the classes, would necessarily have component means at the center – but this is not the case.

we will show that $|\Sigma'_i - \Sigma'_j| \rightarrow 0$ quadratically as $\Sigma'_j \rightarrow 0$. This is formalized in the following proposition which is needed to establish the main result of the next section. We remark that the convergence of each μ'_i to μ'_j is only linear.

Proposition 5 $\|\Sigma'_i - \Sigma'_j\| = O(\overline{\lambda}^2(\Sigma'_j))$

proof: The difference $\Sigma_i^{-1} - \Sigma_j^{-1}$ is constant and is denoted by C. Then:

$$\Sigma'_i - \Sigma'_j = (\Sigma'^{-1}_j + C)^{-1} - (\Sigma'^{-1}_j)^{-1}$$

Denoting Σ'^{-1}_j as P for clarity this becomes:

$$\begin{aligned}(P+C)^{-1} - P^{-1} &= [(I + CP^{-1})P]^{-1} - P^{-1} \\ &= P^{-1}[I - (I + CP^{-1})^{-1}]\end{aligned} \tag{18}$$

Now clearly $I + CP^{-1} \rightarrow I$ with $\overline{\lambda}(P^{-1})$ and the eigenvalues of $I + CP^{-1}$ approach unity at this rate. So $(I + CP^{-1})^{-1} \rightarrow I$ as well – also with $\overline{\lambda}(P^{-1})$. Hence $\|I - (I +$

$CP^{-1})^{-1}\| = O(\overline{\lambda}(P^{-1}))$ in Eq. 18. But $\|P^{-1}\| = O(\overline{\lambda}(P^{-1}))$ as well, so $\|(P+C)^{-1} - P^{-1}\| = O(\overline{\lambda}^2(P^{-1}))$. $\square$

5.3 Emphasis Reparameterization

The question "Can emphasis induce a mixture class-equivalent to an arbitrary one?" posed at the beginning of the previous section is reasonable since from theorem 5 we know that the means of such a mixture can lie within an arbitrarily small ball located anywhere in space. So confinement to the convex hull of the samples is not necessarily a restriction. Likewise the covariances can be chosen to have arbitrarily small Frobenius norm.

The main result of this section is a qualified positive answer to this question. It states that a modified form of emphasis can, for almost all sufficiently large observation sets, induce a mixture which is class equivalent to an arbitrary one.

Modifications are needed because even though the means and covariances may be chosen with a great deal of freedom, they are still highly constrained by the conditions of Eq. 15 – and it is sometimes impossible to generate a mixture which has satisfies these constraints by the form of emphasis employed by EM. The next section expands on the limits of EM-style emphasis.

The required modifications consist of a single change to Eq. 10 which becomes:

$$\Sigma = \sum_{i=1}^{n} \gamma_i (s_i - \mu)(s_i - \mu)^t \tag{19}$$

where the leading normalization has been removed. This allows the *scale* of the covariance to be adjusted without affecting the mean.

Before stating and proving our reparameterization result, a particular matrix is defined whose nonsingularity is a condition of the theorem.

Condition 1 *Denote by S_i the column vector formed by concatenating $s_i s_i^t$ and s_i, and let n (the number of samples) equal $d(d+3)/2$. Next form square matrix S by combining these columns. For example, if $d = 2$ we have:*

$$S \triangleq \begin{pmatrix} s_{1,1}s_{1,2} & s_{2,1}s_{2,2} & s_{3,1}s_{3,2} & s_{4,1}s_{4,2} & s_{5,1}s_{5,2} \\ s_{1,1}^2 & s_{2,1}^2 & s_{3,1}^2 & s_{4,1}^2 & s_{5,1}^2 \\ s_{1,2}^2 & s_{2,2}^2 & s_{3,2}^2 & s_{4,2}^2 & s_{5,2}^2 \\ s_{1,1} & s_{2,1} & s_{3,1} & s_{4,1} & s_{5,1} \\ s_{1,2} & s_{2,2} & s_{3,2} & s_{4,2} & s_{5,2} \end{pmatrix}$$

Our condition is that S be nonsingular.

It is not hard to show that this condition is almost always satisfied, a fact formalized by:

Proposition 6 *Regarding a sample of size $d(d+3)/2$ as a single random variable of dimension $d^2(d+3)/2$, the set of such vectors which give rise to singular S matrices, has measure zero.*

proof: Since the matrix consists of monomials of distinct structure, no non-zero linear combination (over the ring of polynomials) of its columns or rows vanishes. So its determinant, as a polynomial, is not identically zero. But the roots of a non-zero polynomial form a set of zero measure. $\square$

Next we simplify our setting, adjust notation, and establish one more proposition before turning to the theorem. Without loss of generality, we may assume that our samples have mean zero. To see this, first let t denote some target mean, where we seek stochastic γ values such that $\sum \gamma_i s_i = t$. We may equivalently solve $\sum \gamma_i s'_i = t'$, where $s'_i \triangleq s_i - E[S]$ and $t' = t - E[S]$, since $\sum \gamma_i s'_i = (\sum \gamma_i s_i) - E[S]$. The problem of expressing a target covariance is unchanged by coordinate translation, since $(s_i - t) = (s'_i - t')$. Notice that this is true even if the γ are not stochastic. So in what follows, we will drop the *prime* notation and simply assume that $E[\{s_i\}] = 0$.

The mixture we generate will have all of its means located near zero, the mean of the sample set. The distinguished mixture component, identified with index j in the previous section, is located exactly at the origin. The location of the other mean vectors, are thought of as small displacements δ from the origin. We then write $s_i(\delta)$ to mean $s_i - \delta$, and the S matrix is then parameterized by δ in the obvious way, and denoted $S(\delta)$. Note that $S(0)$ is just S as originally defined.

Proposition 7 *Given nonsingular S, $\exists c > 0$ such that $\forall \delta$ satisfying $\|\delta\| < c$, $S(\delta)$ is nonsingular.*

proof: Immediate from the continuity of the δ parameterization of S. $\square$

Each value of δ corresponds to a choice of mean vector, and gives rise to a linear mapping $S(\delta)$. That is, for each choice of δ, a linear map arises. The proposition tells us that so long as our mixture's means are kept within distance c of the origin, all associated linear mappings are invertible.

The utility of $S(\delta)$ is that it allows us to simultaneously express the objectives of obtaining by emphasis a specified mean vector, and covariance matrix. If Γ is a nonnegative emphasis vector, δ_t is the targeted mean vector, and Σ_t the desired covariance matrix, then we must solve the system $S(\delta_t)\Gamma = \Sigma_t : 0$. Here Σ_t is regarded as a vector which is concatenated (denoted ":") with the d-dimensional zero vector.

The Γ above is nonnegative, but not necessarily stochastic. This may seem to be inconsistent with our definition of the generation of a mean vector by emphasis, because of the normalization it includes. But it is not since $\sum \gamma_i(s_i - \delta) = 0$ is equivalent to $(1/\sum \gamma_i) \sum \gamma_i = \delta$, which is exactly our definition.

Theorem 6 *Given any d-dimensional normal mixture M with k components, and $s_1, \ldots, s_n \in \mathbb{R}^d$ such that $n \geq d(d+3)/2$ with some subset of size $d(d+3)/2$ satisfying condition 1, then there exists a $k \times n$ table of nonnegative values $\{\gamma_{i,j}\}$, and nonnegative values $m_1, \ldots, m_{k-1}$ with $\sum m_k \leq 1$, such that the normal mixture M' generated as described below, is class-equivalent to M.*

1. *The mixing parameters of M' are $m_1, \ldots, m_{k-1}, 1 - \sum_{j=1}^{k-1} m_j$.*

2. *Each mean μ_i' within M' is given by:*

$$\mu_i = \frac{1}{\sum_{j=1}^n \gamma_{i,j}} \sum_{j=1}^n \gamma_{i,j} s_j$$

3. Each covariance Σ_i' within M' is given by:

$$\Sigma_i = \sum_{j=1}^n \gamma_{i,j}(s_j - \mu_i)(s_j - \mu_i)^t$$

Also, the number of parameters may be reduced to $(k-1)d(d+3)/2 + d$, matching the complexity of the canonical parameterization, by replacing the top γ-table row, by a single new parameter α.

proof: Choose some subset of the $\{s_i\}$ of size exactly $d(d+3)/2$ that satisfies condition 1. We disregard the other elements entirely, and therefore simply assume that $n = d(d+3)/2$. As argued earlier, we may also assume without loss of generality that the $\{s_i\}$ have mean zero. Then the distinguished mixture component from theorem 5, with parameters (μ'_j, Σ'_j), is chosen as follows. Its mean μ'_j is the origin, and its covariance Σ'_j is proportional to the average of the element self-outer-products, i.e.:

$$\Sigma'_j = \alpha \frac{1}{n} \sum_{i=1}^n s_i s_i^t$$

This may be thought of as choosing the first row of the γ table to be $1/n$, and introducing a new scale parameter α. The table's top row elements are no longer parameters, so the total becomes $(k-1)d(d+3)/2$ table parameters, plus $d-1$ mixing parameters, plus α – matching the count in the statement of the theorem. As described in the proof of theorem5, the choice of μ'_j is arbitrary, but Σ'_j may need to be scaled down to preserve the positive nature of all matrices. The number of resulting parameters in a direct parameterization (i.e. consisting of means, covariances, and mixing parameters), then matches our count.

As $\alpha \to 0$ we know that the $\mu'_i \to \mu'_j = 0$. Our first step is to choose α sufficiently small, so that the largest $\|\mu'_i\|$ is smaller than the c of proposition 7. Next, α is possibly reduced further until each covariance matrix Σ'_i arising from Eq. 7 is positive.

Each μ' corresponds to a δ value in our definition of $S(\delta)$. The reference mean, located at the origin, corresponds to $\delta = 0$, and by our construction, arises from weighting the s_ℓ by non-negative γ values – in this case uniform ones. Associated with each ${\mu'}_i$ there is also a ${\Sigma'}_i$. Recall from our earlier discussion, that our focus is on the equation $S(\mu_i)\Gamma = \Sigma_i : 0$. Since $S(\delta)$ is non-singular for each of the ${\mu'}_i$, we know that some vector of γ values satisfies this equation. We have only to show that solutions consisting only of nonnegative values can be found.

We will say than a point is *representable* under some $S(\delta)$, if its inverse image under this mapping consists only of non-negative values. There exists a representable open neighborhood about $\alpha({\Sigma'}_j : 0)$ under $S(0)$, since the inverse image of this point[2] is the constant vector α/n, and consists of strictly positive values.

Within our bounds on δ, it uniform-continuously[3] parameterizes $S(\delta)$. Hence there exist values c' and ϵ, such that for all δ with $\|\delta\| \leq c'$, the open ball about $\alpha({\Sigma'}_j : 0)$, with radius ϵ, is representable under $S(\delta)$. To avoid introducing another symbol, let ϵ now denote the maximum such radius.[4]

Next notice that the space of representable points is convex and includes the origin. So representability of $\alpha({\Sigma'}_j : 0)$ implies representability for all α values. Now if necessary, reduce α further so that all the μ' are within c' of the origin.

Now let P denote the subspace of nonnegative Γ vectors. Let:

$$T \triangleq \bigcap_{\|\delta\| \leq c'} S(\delta) P$$

Subspace T is the portion of the range, representable under *any* $S(\delta)$ such that $\delta \leq c'$.

About $\alpha({\Sigma'}_j : 0)$ we have seen that there is a representable ball of radius ϵ which touches the boundary of T. We now denote this radius by $\epsilon(\alpha)$ since we are about to consider its behavior as $\alpha \to 0$. By our earlier comments, T includes $\alpha({\Sigma'}_j : 0)$, and all proportional vectors, in particular as $\alpha \to 0$.

---

[2]The inverse image is a unique point because $S(\delta)$ is invertible.

[3]and in fact nearly linearly for small δ.

[4]A maximum exists because entries in S corresponding to covariance diagonal, are nonnegative, preventing many negative values from being representable.

The geometric observation key to our proof is that $\epsilon(\alpha)$ can shrink only as fast as α itself, since this value represents the shortest distance to some point set, in this case T^c. We imagine T to be a tunnel, leading to the origin while constricting, with the boundary of T^c forming the tunnel's wall.

Focus now on some ${\mu'}_i$. While there is a corresponding representable ball about $\alpha({\Sigma'}_j : 0)$ of radius $\epsilon(\alpha)$, there is no guarantee that ${\Sigma'}_i$ is within it. As $\alpha \to 0$, we have seen that the radius of this ball shrinks linearly with α. But the distance of ${\Sigma'}_i$ from ${\Sigma'}_j$ by proposition 5 shrinks quadraticly with α, whence eventually, i.e. for small enough α, ${\Sigma'}_i$ becomes representable. Then α is reduced as necessary for for each component of the mixture, so that afterwards, every $\Sigma' : 0$ is representable. $\square$

Other emphasis theorems are possible. In particular it is much easier to establish a similar result in which the weights may be either positive or negative because then the only requirement is that $S(\delta)$ be nonsingular.

5.4 The Limitations of EM-style Emphasis

In the previous section we saw that using a particular form of emphasis reparameterization, the *a posteriori* behavior of any mixture could be matched. We say that such a sample set and emphasis method is *universal*. This section is by contrast essentially negative, and begins with an example demonstrating that even in one dimension, EM-style emphasis is not universal.

Our example is in $\mathbb{R}^1$, and includes two observation points located at 0 and 1. The leading normalization factors in Eq. 10,9 amount to enforcing convex combination, so there is only one degree of freedom given our two element observation set. If γ denotes the weight of the first point, then the second has weight $1-\gamma$. The induced mean is just γ and the induced variance $\gamma(1-\gamma)$. Our objective is to generate a two element mixture which is class equivalent to an arbitrary one, so two emphasis parameters γ_1 and γ_2 are needed. The two constant differences to be matched (from Eq. 15) are denoted Δ_Σ and Δ_μ. That these are independent and unconstrained characteristics an equivalence class follows from the observation that we may, without loss of generality, set one of the target means to zero. The constraints then become:

$$\begin{aligned}\frac{1}{\gamma_1(1-\gamma_1)} - \frac{1}{\gamma_2(1-\gamma_2)} &= \Delta_\Sigma \\ \frac{1}{1-\gamma_1} - \frac{1}{1-\gamma_2} &= \Delta_\mu\end{aligned}$$

Choose $\Delta_\mu > 0$. From our second constraint we have $\gamma_1 = 1 - 1/[\Delta_m u + 1/(1-\gamma_2)]$ from which it follows that $\gamma_1 = \gamma_2$ only when they both are 1. But in this case $\Delta_\mu = 0$, contradicting our choice. So $\gamma_1 \neq \gamma_2$. Now when $\gamma_2 = 0$, $\gamma_1 > 0$, so here $\gamma_2 < \gamma_1$. Because they are never equal, and their relationship is continuous, this inequality must hold for any pair satisfying the constraints. But it is easily verified that $\Delta_\Sigma - \Delta_\mu = 1/\gamma_1 - 1/\gamma_2$ whence this quantity is negative. So no pair γ_1, γ_2 exists which satisfies the constraints given say $\Delta_\mu = 1$ and $\Delta_\Sigma = 2$.

We remark that if instead, one is interested in matching the *a posteriori* class behavior of a mixture at only the given sample points, then the example's argument does not apply. Indeed it may be verified that this less stringent requirement can be satisfied. This is not entirely surprising since given exactly $d(d+3)/2$ sample points, one has three free parameters, and must match only two. This is interesting, and we conjecture that it is true more generally for $d > 1$ and $k > 2$ – so long as $n = d(d+3)/2$. In any event, as a result of our arguments at the end of this section, it cannot be true for arbitrarily large n. Since we are interested in reparameterizing problems with arbitrarily many samples, we will not consider the matter further in this chapter.

The example above applies to dimension 1 only, and a particular set of sample points. Moreover, its analytical approach does not generalize easily. We now show that in the general case EM-style emphasis is not universal.

Without loss of generality we will assume ${\mu'}_j = 0$ – since otherwise, the problem, and any solution thereto, can be translated appropriately. We may also assume the sample points are distinct. Eq. 15 becomes:

$$\Delta_{\mu_i} = {\Sigma'}_i^{-1} {\mu'}_i \tag{20}$$

where $\Delta_{\mu_i} \triangleq \Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j$. Our focus is on the simple rearrangement of Eq. 20:

$$\Sigma'_i \Delta_{\mu_i} = \mu'_i \tag{21}$$

Independent of the $\Sigma'_1, \ldots, \Sigma'_k$ we are free to set $\mu'_1, \ldots, \mu'_k$ so that the Δ_{μ_i} have arbitrary values. In particular, we will set them so that $\Delta_{\mu_i} = (CC \ldots C)^t$ where $C > 1$ is a constant.

Now focus on target mixtures such that $\Delta_{\Sigma_i} \triangleq \Sigma_i^{-1} - \Sigma_j^{-1}$ has value αI where $\alpha \to \infty$. Then $\Sigma'^{-1}_i = \alpha I + \Sigma'^{-1}_j$. So independent of the choice of particular $\Sigma_1, \ldots, \Sigma_k$, and Σ'_j, each Σ'^{-1}_i, where $i \neq j$, will approach αI. There inverses therefore approach the diagonal matrix $1/\alpha I$.

As $\alpha \to \infty$, we adjust means as necessary so as to maintain $\Delta_{\mu_i} = (CC \ldots C)^t$. Then from Eq. 21 it is apparent that μ'_i is very nearly a scaled up copy of the diagonal of Σ'_i. Since the off diagonal elements approach zero, it must eventually be the case that $\|\Sigma'_i\| < \|\mu'_i\|$. Also, since the sample set is finite, it must eventually be that both are smaller than say 1/100th of the smallest distance between sample points. When this happens, the distance from μ'_i to the nearest sample will be at least $\|\mu'_i\|$.

Now the covariance Σ'_i is a convex combination of matrices of the form $(s_\ell - \mu'_i)(s_\ell - \mu'_i)^t$ – and the diagonals are nonnegative. The norm of Σ'_i is at least equal to the norm of its diagonal, but the diagonal is just the distance from the sample point to μ'_i. From this it follows that $\|\Sigma'_i\| \geq \|\mu'_i\|$ which presents a contradiction, whence EM-style emphasis is not universal. Our arguments above establish:

Theorem 7 *In the setting of theorem 6, altered after the fashion of EM so that each covariance Σ'_i is given by:*

$$\Sigma_i = \frac{1}{\sum_{j=1}^n \gamma_{i,j}} \sum_{j=1}^n \gamma_{i,j}(s_j - \mu_i)(s_j - \mu_i)^t$$

there exist target mixtures which may not be generated by emphasis.

We have seen that no number of sample points make EM-style emphasis universal. But given enough points, we can certainly identify a particular class equivalence-class, since all functions involved are analytic with a finite number of parameters.

So given enough sample points, a reparameterization *must* be universal in order to match a target distribution at the sample points only. Now we are interested in reparameterizations which apply given an unlimited number of points. Therefore, unlike the modified emphasis reparameterization introduced in the previous section, EM-style reparameterization is simply not expressive enough to safely reparameterize *a posteriori* optimization problems.

5.5 Concluding Remarks

The *a posteriori* degeneracy we clarify in section 5.2 for normal mixtures must be added to the list of remarkable characteristics of Gaussians. We have not considered analogues of theorem 5 for other densities.

The reparameterization of section 5.3 is of mathematical interest, and lends some credibility to approaches which attempt to maximize *a posteriori* probability by adjusting weights on the available samples – perhaps according to some error measure. An examination of reparameterization as a primary optimization technique, represents an interesting area for future work. We must however caution that while our proofs are constructive, we have not considered the matter of numerical sensitivity. Indeed, if one attempts to emulate a mixture using means far displaced from their natural locations, the mixing parameters become quite small. In these cases floating point underflow is a virtual certainty.

One should not interpret section 5.4 to say that EM-style emphasis will not work in practice for many problems. It merely exposes its theoretical limitations.

Acknowledgments

We thank Leonid Gurvits for several helpful discussions – and in particular for pointing out the simple factorization in proof of proposition 5, and simplifying our proof of proposition 6. We also thank Joe Kilian for several helpful discussions which contributed to our understanding of degeneracy and class equivalence.

Chapter 6

Conditional Normal Mixtures

Discrete context models are well established and successful tools with which to model or compress natural language text. These models predict the next letter, conditioned on the last few letters seen.

When the data are continuous, as in speech recognition or image processing, the situation is more complicated because one cannot rely upon exact context matches. As a result, context is typically either ignored, resulting in weak models, or it is incorporated in a way that is in some sense technically unsatisfactory, i.e. the contexts overlap so that the future is in effect used to predict the past.

This chapter discusses the learning of context models for continuous problem domains within a strict probabilistic framework. The probability of a time series, or of a static object such as an image, is expressed as a product of conditional probabilities. Each term predicts a never-before-seen part of the observation conditioned on earlier-seen portions. Models of this form are referred to in the literature as *causal*.

6.1 Introduction

Our focus is on such models where each term arises by conditionalizing a mixture of primitive models, e.g. a conditionalized mixture of normal densities. We show that reestimation of such a model's parameters may be reduced to two simpler tasks. In the case of normal densities one of these is trivial, and the other may be approached the emphasis parameterization introduced in the previous chapter.

Now in the case of normal mixtures one might instead use a conventional gradient descent approach and a standard parameterization. Our contribution is the identification of a particularly simple, generic, and in some sense natural alternative.

In the case of a time series (e.g. a speech signal) the objective is usually to learn the parameters of a single conditional model that is then applied repeatedly to evaluate the probability of a series. For static objects such as images of handwritten digits, one might instead associate a different conditional model with each pixel position. The image's probability is then the product of the predictions made by these distinct models.

The original motivation for this work was in fact the learning of improved causal models for images – in particular of handwritten symbols. The developments of this chapter may form the basis for future experiments in this direction.

Given a collection of pairs $\{(c_i, x_i)\}$ where $x_i \in \mathbb{R}^n$ and $c_i \in \mathbb{R}^M$, our objective is to maximize over Φ:

$$\prod_i p(x_i|c_i, \Phi)$$

We begin by observing that if the model is a single multivariate normal density, then this problem may be solved by instead optimizing the joint probability:

$$\prod_i p(x_i, c_i|\Phi)$$

and conditionalizing the resulting normal density – and it is well known that a single normal density is optimized (in the ML sense) by merely choosing the sample mean and covariance. The observation follows from the closure properties of multivariate normal densities under multiplication and conditionalization – and specifically because the product of a conditional normal density $p(x|c)$ and an ordinary normal density $p(c)$, is an ordinary normal density $p(x, c)$.

So the case of a single normal density is in a sense uninteresting. We remark that linear predictive speech coding (LPC) may be viewed as such a model where $n = 1$.

In general these simple conditional forms arising from a single normal density form a prediction x based on a linear transformation of the conditioning information c. As such their expressive power is limited. In particular they cannot deal with *modality*. For example, the population of c values may separate into two obvious clusters, each leading to very different predictions for x. A simple linear predictor cannot cope with this situation and will instead make a single blended prediction.

By moving from single densities to mixture densities, nonlinear predictions result and modality can be addressed.

A general mixture may be written

$$p(x, c) = \sum_{i=1}^{k} p(x, c|\omega_i)p(\omega_i)$$

to describe the joint density (x, c). Provided that each component $p(x, c|\omega_i)$ has an associated conditional form $p(x|c, \omega_i)$ we may then write:

$$p(x|c) = \sum_{i=1}^{k} p(x|c, \omega_i)p(\omega_i|c) \tag{22}$$

Such conditionalized mixture forms can capture modality because their mixing coefficients $p(\omega_i|c)$ are not constant. That is, they depend on the context c and the mixture stochastically selects a component based on the context.

If each component is a normal density with fixed parameters, the IMM technique for adaptive estimation and tracking results [BSL93, BBS88]. From this perspective, our interest is in recovering the model's parameters from observations.

For a single normal density we have seen that an optimal conditional model may be formed by first forming an optimal joint model, and then conditionalizing – and that the optimal joint model is specified by the sample mean and covariance.

Building an optimal joint model for a mixture of normal densities is a non-trivial and heavily studied problem. The most popular approach is Expectation Maximization (EM) which is discussed in earlier chapters and is an iterative technique for climbing to a locally optimal model.

But the situation is worse yet for conditionalized mixture forms because it turns out that conditionalizing an optimal joint mixture model, need not result in an optimal conditional model.

If the underlying joint density is exactly a normal mixture of k components, then the optimal conditional density is of course just the conditionalized form the joint density (by direct calculation). But given a finite sample, and especially one that is explained poorly by a normal mixture of k components, the optimal conditional model will in general be different.

To see this we offer the following argument sketch: suppose that x and c are independent with x explained perfectly by a k element mixture. Then $p(x|c) = p(x)$ Now consider equation 22. The $p(\omega_i|c)$ terms are constant in an optimal model for $p(x)$ but instead are conditioned on c in the equation. By modeling the joint density (x, c) it will seldom be the case that these terms are constant whence the resulting model is suboptimal. That is, in the case of independence, the context should be ignored.

The next section further reveals the role of these *a posteriori* class functions in the overall optimization. An example serves to illustrate that the ML estimate does not necessarily optimize *a posteriori* class probabilities. Consider figure 14(a) which depicts four points along the x-axis, two w_1, w_2 labeled white and two b_1, b_2 black. These colors correspond to two classes ω_1 and ω_2. The functions sketched above them represent the ML normal density associated with each class. Notice the mean of each is centered at the midpoint between each pair of points, and that the variance is considerable. If one computes, for example, the *a posteriori* probability $p(\omega_1|w_2)$ based on the two normal densities shown, the result will certainly exceed 1/2 but clearly falls well short of unity. If the densities are replaced with the more peaked forms of the figure's part (b), all *a posteriori* probabilities are in closer correspondence to the point colors. This is true despite the fact that their means have drifted away from the original midpoint locations. This illustrates that ML optimization of a labeled dataset need not optimize *a posteriori* class probabilities.

Figure 14 also illustrates the general concept of emphasis introduced in the previous chapter since the densities of part (b) may be generated by placing more weight on points w_1 and b_2 and computing the weighted mean and variance. As the weight on these outer points is increased the densities approach a pair of vertical pulses (also depicted) and the *a posteriori* probabilities approach the ideal values of zero or one in correspondence with each point's color.

6.2 Conditional Discrete Mixtures

This section demonstrates how one may separate the parameter-learning task for conditional mixtures, into two simpler problems. This is accomplished using the mathematical machinery of EM combined with the introduction of new parameters.

Let x denote the random variable being predicted/modeled, and c a random variable which conditions the prediction. To simplify our notation we will generally use the single function p to denote both probability densities and probabilities; as well as related marginals and conditional forms. In all cases it should be possible to distinguish different functions by context. We begin by defining the central problem of this chapter:

Definition 7 *An instance of the* <u>*Minimize Relative Conditional Entropy*</u> *(MRCE) problem consists of two sets of values* $\mathcal{X} = \{x_1, \ldots, x_{n_x}\}$ *and* $\mathcal{C} = \{c_1, \ldots, c_{n_c}\}$*, a probability function* $m(x, c)$*, and a parameterized conditional probability density* $p(x|c, \Phi)$*. The problem consists of finding a revised parameter estimate* $\overline{\Phi}$ *such that* $H_m(\mathcal{X}|\mathcal{C}, \overline{\Phi}) < H_m(\mathcal{X}|\mathcal{C}, \Phi)$*, or announcing that a local minimum has been reached. That is, maximize with respect to* $\overline{\Phi}$*, the log-likelihood:*

$$L(\overline{\Phi}) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_c} m(x_i, c_j) \log p(x_i|c_j, \overline{\Phi})$$

where the values x_i *and* c_j *are members of some (possibly finite) measure spaces.*

We remark that $m(x, c)$ may be any non-negative function since it may be normalized to form a probability function without affecting the problem at hand. The

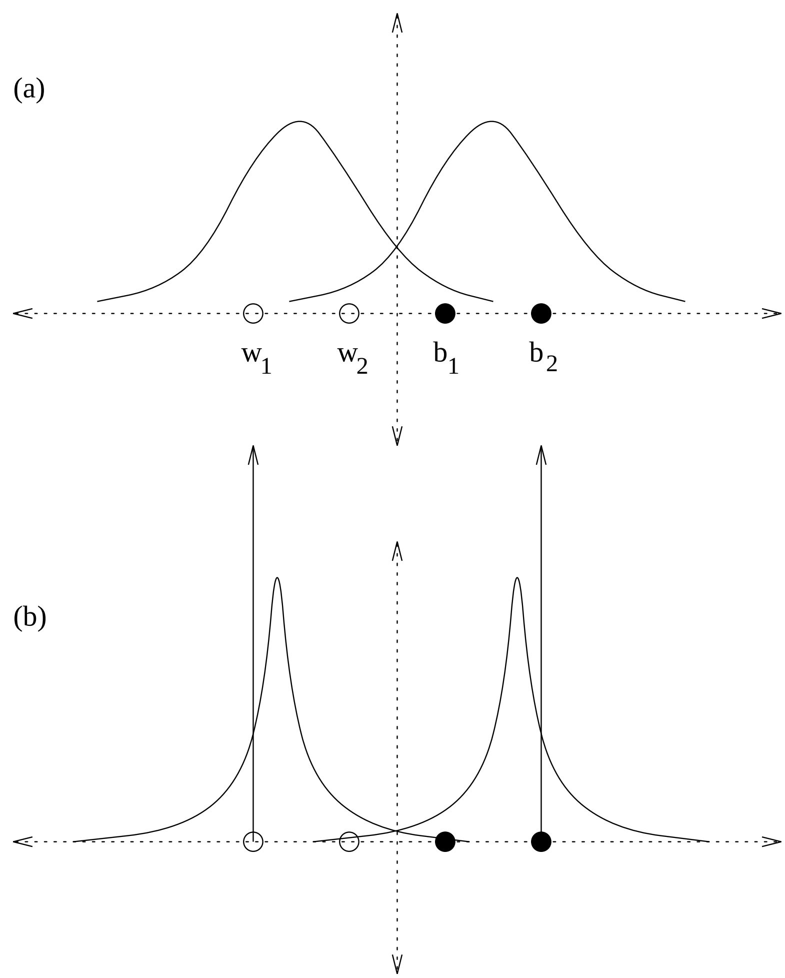

Figure 14: The maximum-likelihood normal densities shown in part (a) arising from the four labeled points, do not lead to optimal *a posteriori* classification of these points. The more peaked densities of part (b) are better.

MRCE problem as one step of an iterative approach since it is not generally possible to solve directly for even a local maximum of $L(\overline{\Phi})$.

Our interest is in parameterized conditional densities $p(x|c, \Phi)$ which arise from *finite mixture densities* of the form:

$$p(x, c|\Phi) = \sum_{k=1}^{m} p(x, c, \omega_k|\Phi)$$

where ω is a hidden discrete random variable which corresponds to selection of a component of the mixture. We then write:

$$p(x, c|\Phi) = \sum_{k=1}^{m} p(x|c, \omega_k, \Phi^1)p(c, \omega_k|\Phi^2)$$

where it assumed that $\Phi = \Phi^1 \cup \Phi^2$ with $\Phi^1 \cap \Phi^2 = \emptyset$. Were this not the case, we could make it so by duplicating any common parameters. The resulting parameterized density includes the original pdf as a special case. So except for the matter of added model complexity, our assumption that $\Phi_1 \cap \Phi_2 = \emptyset$ is made without harm[1]. The generation of a pair (x, c) may be thought of as a two stage process. First, a pair (c, ω) is chosen according to Φ^2 Second, x is generated according to Φ^1 and conditioned on the earlier choice of (c, ω). Now $p(x, c|\Phi) = p(x|c, \Phi)p(c|\Phi)$ from which we easily derive the conditional prediction formula:

$$\begin{aligned} p(x|c, \Phi) &= \frac{1}{p(c|\Phi)} p(x, c|\Phi) \\ &= \frac{1}{p(c|\Phi)} \sum_{k=1}^{m} p(x|c, \omega_k, \Phi^1)p(c, \omega_k|\Phi^2) \\ &= \sum_{k=1}^{m} p(x|c, \omega_k, \Phi^1)p(\omega_k|c, \Phi^2) \end{aligned}$$

Thus the conditional form required by the MRCE problem arises naturally from the joint mixture density. Next, $p(c, \omega_k|\Phi^2) = p(c|\omega_k, \Phi^2)p(\omega_k|\Phi^2)$, and as before we

[1]That is, for a given optimization problem, the global optimum over the enlarged parameter set, is at least as good as the global optimum for the original problem

can without weakening the model, assume that Φ^2 is made up of the m marginal values $p(\omega_k|\Phi^2)$ which we will sometimes denote as just $p(\omega_k)$, and m independently parameterized densities $p_k(c|\omega_k, \Phi_k^2)$. We will omit the subscript on p since it is clear from the conditioning on Φ_k^2 what density we are referring to. We remark that the same result may be reached by starting with the more common definition of a mixture density:

$$p(x, c|\Phi) = \sum_{k=1}^{m} \alpha_k p_k(x, c|\phi_k)$$

where $\sum_{k=1}^{m} \alpha_k = 1$, the α_i are nonnegative, and each p_k is a parameterized probability density function with $\Phi \triangleq (\alpha_1, \ldots, \alpha_m, \phi_1, \ldots, \phi_m)$. We prefer our presentation however since it seems more natural given the structure of later derivations.

Definition 8 *An instance of the* <u>*Mixture-MRCE*</u> *problem is an instance of MRCE with the additional assumption that the parameterized probability density* $p(x|c, \Phi)$ *arises from a finite mixture density.*

Now suppose that $\mathcal{O} = \{(x_k, c_k)\}_{k=1,\ldots,n}$ is an independent sample of unlabeled observations where $\mathcal{X}$ denotes $\{x_k\}$ and $\mathcal{C}$ denotes $\{c_k\}$. Our approach in this chapter is that of maximum-likelihood estimation[2], and we are therefore interested in finding a value for Φ which maximizes $p(\mathcal{X}|\mathcal{C}, \Phi)$, i.e. $\prod_{i=1}^{n} p(x_i|c_i, \Phi)$. Such a value need not exist in general due to degenerate cases such as zero variance distributions, but we will not consider this complicating detail here. Since we can equally well maximize the log-likelihood, we have the following:

Observation 1 *The problem of finding an improved parameter for conditional mixture applied to a finite set of observations* $\mathcal{O} = \{(x_k, c_k)\}_{k=1,\ldots,n}$, *is a special case of the Mixture-MRCE problem in which* $m(x_i, c_j) = 1$ *when* $i = j$ *and* 0 *otherwise.*

[2]Maximum-likelihood estimation is sometimes understood to include the assumption that the distribution from which the sample is drawn is a member of the parameterized family under consideration. We are modelers and assume that nature can be only approximately described by our densities. We therefore do not imagine that there is a true value Φ^* which must be estimated. We simply seek the best *fit* of our model, to the observed data.

The definitions above and much of the development that follows may be generalized to the case where $\mathcal{X}$ and or $\mathcal{C}$ are measure spaces; in which case the summations become integrals. We will however present the discrete case since it corresponds to our original problem of parameter estimation given a finite sample.

We now show that the mixture-MRCE problem may be approached by reducing it to an instance of itself with particular structure, and multiple independent ordinary (nonmixture) MRCE problems. The ordinary MRCE problems are simpler because each concerns only a single component of the original mixture and excludes entirely the mixing coefficients $p(\omega_k)$. The mixture-MRCE subproblem is simpler because it excludes the x_i of the original problem, i.e. it concerns only the model $p(c, \omega_k|\Phi^2)$.

Theorem 8 *Given a parameter value Φ for an instance of mixture-MRCE, the problem of finding an improved parameter value $\overline{\Phi}$ may be reduced to another instance of the mixture-MRCE – and a set of independent instances of simple (nonmixture) MRCE, where each is confined to a single component of the original mixture. By this we mean that any parameter values which improve these subproblems, may be combined to form $\overline{\Phi}$.*

Proof: Using the same steps used derive the well-known EM algorithm, we have:

$$\begin{aligned}
Q(\overline{\Phi}|\Phi) &= \sum_{i=1}^{n_x}\sum_{j=1}^{n_c}\sum_{k=1}^{m} m(x_i, c_j)p(\omega_k|x_i, c_j, \Phi)\log p(x_i, \omega_k|c_j, \overline{\Phi}) \\
&= \sum_{i=1}^{n_x}\sum_{j=1}^{n_c}\sum_{k=1}^{m} m(x_i, c_j)p(\omega_k|x_i, c_j, \Phi)\log p(\omega_k|c_j, \overline{\Phi^2}) && (23) \\
&+ \sum_{i=1}^{n_x}\sum_{j=1}^{n_c}\sum_{k=1}^{m} m(x_i, c_j)p(\omega_k|x_i, c_j, \Phi)\log p(x_i|\omega_k, c_j, \overline{\Phi^1}) && (24)
\end{aligned}$$

Reordering the summations in Eq-23 and grouping yields:

$$\sum_{k=1}^{m}\sum_{j=1}^{n_c}\left[\sum_{i=1}^{n_x} m(x_i, c_j)p(\omega_k|x_i, c_j, \Phi)\right]\log p(\omega_k|c_j, \overline{\Phi^2})$$

which after normalization of the bracketed term is easily recognized as an instance of the mixture-MRCE problem. The second term Eq-24 making up Q may be similarly transformed to yield:

$$\sum_{k=1}^{m} \left[\sum_{i=1}^{n_x} \sum_{j=1}^{n_c} (m(x_i, c_j) p(\omega_k | x_i, c_j, \Phi)) \log p(x_i | c_j, \omega_k, \overline{\Phi^1}) \right]$$

which is just m independent instances of the simple MRCE problem. Finally, observe that Eq-6.2 and Eq-6.2 are independent of one another since the first is an optimization problem over $\overline{\Phi^1}$ and the second over $\overline{\Phi^2}$ – completing our proof. $\Box$

Notice that if $\mathcal{C}$ contains a single value, the complexities introduced by the conditional form of the problem vanish and one is left with the standard EM setting in which the mixture-MRCE subproblem is trivially maximized. Also, the conditional estimation problem becomes a plain estimation problem.

6.3 Application to Normal Densities

For normal densities the conditional estimation problems of Eq-24 are easily solved since the optimal conditional normal density is obtained by finding the optimal joint density and then conditionalizing.

The MRCE problem of Eq-23 is however much more difficult. It, or for that matter the entire problem, might be addressed by gradient descent where the covariance matrices have been reparameterized to ensure positive definiteness. The purpose of this chapter is to describe an alternative approach.

Using the emphasis reparameterization results of the previous chapter, we may (except for degenerate cases) express *a posteriori* class behavior by reweighting the available samples.

So in particular Eq-23 may be approached in this way. That is, the $\{c_i\}$ may be reweighted until the term is maximized. This amounts to a reparameterization of the problem in terms of these weights rather than in terms of means, covariance matrices, and mixing coefficients.

The fact that for normal densities one may pass easily from such a weight set, to an ML traditional parameter set, is essential to the approach. That it is capable of expressing the solution is also a special characteristic of normal densities (see previous chapter).

But in practice one might wish to use this reparameterization (without proof of expressiveness) for other densities for which an ML estimate of a weighted sample set is readily available. Its virtue is that the particular density may then be viewed as a data type referred to by a single optimization algorithm.

A particularly simple numerical prescription for optimization is that of Powell (see [Act70, Bre73, PFTV88]). Speed of convergence is traded for simplicity in that no gradient computation is required. Emphasis reparameterization does however introduce one wrinkle. The number of parameters can easily exceed the degrees of freedom in the underlying problem. In the case of normal mixtures, the number of natural parameters (i.e. mean vector values and covariance matrix entries) may be far less than the number of training samples available. In this case we propose to modify Powell's method to use an initial set of random directions with size matching the number of native parameters.

The result is a simple and highly general approach to estimating the parameters of conditional mixtures. That the approach can express the globally optimal parameter set has only been established for normal mixtures, but we expect that approach may nevertheless work well in other settings.

6.4 Further discussion of the formulation of the MRCE problem

The conditional entropy minimization problem can express the traditional problem of supervised learning given boolean $m_{k,i}$ with $\sum_{i=1}^{m} m_{k,i} = 1$. That is, each datapoint has an attached label. The MRCE problem is then to minimize the bits required to express the labels, given the data. Our definition of MRCE places no restriction, other than non-negativity, on the $m_{k,i}$. In this section we motivate the form of our definition by showing that these weights have a natural stochastic interpretation.

Imagine a random process where each trial consists of drawing a label from $\omega_1, \ldots, \omega_m$ for each of the points $c_1, \ldots, c_n$. The outcome of each trial is represented by boolean random variables $\delta_{k,i}$ defined to be 1 if c_k is observed to have label ω_i, and 0 otherwise. No independence assumptions are made. The trials might be time dependent; and within a trial, the labels might be dependent in some complex way. Each trial produces a labeled dataset with likelihood:

$$\prod_{k=1}^{n} \prod_{i=1}^{m} p(\omega_i | c_k, \overline{\Phi^{1,2}})^{\delta_{k,i}}$$

Denoting by $L(\overline{\Phi^{1,2}})$ the logarithm of this likelihood, a straightforward consequence of the linearity of expectations is then that:

$$\begin{aligned} E(L(\overline{\Phi^{1,2}})) &= E\left(\sum_{k=1}^{n} \sum_{i=1}^{m} \delta_{k,i} \log p(\omega_i | c_k, \overline{\Phi^{1,2}})\right) \\ &= \sum_{k=1}^{n} \sum_{i=1}^{m} \log p(\omega_i | c_k, \overline{\Phi^{1,2}}) E(\delta_{k,i}) \end{aligned}$$

where the expectation is over trials of the random labeling process. The key feature of this expression is that the expectation of each labeled dataset's log likelihood, is independent of the exact nature of the labeling process; depending only on the individual expectations $E(\delta_{k,i})$. The significance of this independence, is that we can optimize $E(L(\overline{\Phi^{1,2}}))$ with respect to $\overline{\Phi^{1,2}}$, where the only knowledge we have regarding the labeling process is the set of terms $\{E(\delta_{k,i})\}$. Now $\sum_{i=1}^{m} \delta_{k,i} = 1$ since only a single label is drawn for each point. So $\sum_{i=1}^{m} E(\delta_{k,i}) = 1$ as well. These $\delta_{k,i}$ terms are therefore the probability of observing label ω_i given point c_k and correspond to the term $p(\omega_i | c_k, x_k, \Phi)$ in Eq-23. Because of the argument above, we think of the weights as *stochastic labels* attached to each point.

In the setting above, each trial can be thought of as drawing a sample of size n consisting of exactly $c_1, \ldots, c_n$ each time. This is just one example of a process which draws sample of size n, from the $\{c_k\}$ with equal probability $1/n$. An argument essentially the same as that above can then be made for a process in which each trial draws a single data point from $c_1, \ldots, c_n$ and then a single label for that point. By repeated trials, the sample of size n is built. The result is that given only the probability of drawing each c_k, and the $p(w_i|c_k)$, we can optimize our expected log likelihood for a series of n samples, independent of the exact nature of the process.

Now an equivalent problem results if our matrix of weights $M \triangleq \{m_{k,i}\}$ is scaled by a constant so that the sum of its entries is 1, so we will assume that this is the case. It can then be thought of a joint probability density on k, i so that entry may be expressed as a product $p(i|k)p(k)$. More formally, M may be expressed as a diagonal matrix with trace 1 representing $p(k)$, times a row stochastic matrix representing $p(i|k)$.

In summary, an instance of the MRCE problem may be thought of as consisting of $c_1, \ldots, c_n$ and the stochastic knowledge in the two matrices above. The significance of its solution is that it represents a minimum not just over parameter values, but over all processes consistent with this knowledge. If our objective were to maximize expected likelihood rather than expected log likelihood, this would not be true. Thus beyond the obvious mathematical convenience of working with log likelihood, there is a deeper benefit to doing so. Everything we've said is true in a far more general sense for arbitrary density $f(x, y)$ and sets $\mathcal{X}$ and $\mathcal{Y}$, where we focus on a series of T samples and have the likelihood:

$$\prod_{t=1}^{T} \prod_{x \in \mathcal{X}} \prod_{y \in \mathcal{Y}} f(x, y)^{\delta_{x,y}}$$

but we will not repeat the argument above at this slightly more abstract level.

Instead we turn to a concrete example loosely motivated by the problem of portfolio optimization as framed in [Cov84] and discussed in chapter 2, in order to illustrate the mathematical point above.

Suppose that one may draw vectors from $\mathbb{R}^{365}$ at random according to some hidden distribution A, and that the one thing known about this distribution is that only two values 2 and 0.5 are ever drawn in each component position, and that they occur with equal probability in each position. That is, all single variable marginals are known.

Now focus on the product of a vector's components. Our expectation of the $\log_2$ of this product's value is zero because the product becomes a sum of logs, and because of linearity of expectations.

Of significance is that the expectation is zero, independent of the nature of A. By contrast, our expectation of the product's value (without taking the logarithm) is highly distribution dependent. For example, if the distribution generates a vector of 2's half of the time and a vector of 0.5's the rest of the time, the resulting expectation is $(2^{365} + 2^{-365})/2$ and is immense.

The fascinating observation regarding optimizing log-likelihood as opposed to likelihood, is that on the one hand the optimization is over a huge space of hidden distributions, but on the other hand the optimization ignores information in those distributions that can significantly affect the expected likelihood.

6.5 Possible Applications

We conclude by identifying and briefly discussing a number of potential applications for conditional normal mixture models (CNM).

As mentioned earlier, our starting point was handwriting recognition and in particular the problem of offline digit classification. The space of digit images is modeled as a mixture of models, one for each digit. We suggest that each digit's model itself consist of a mixture of models intended to capture the sometimes very different styles used to represent the same numeral. Each digit style model would then consist of a product of CNMs, one for each pixel position, with conditioning on some subset of the earlier-seen pixels. The use of CNMs in this way promises to better model the variation that exists within a single style. The final model may be used to implement a traditional Bayesian classifier. The introduction of discriminative training into this framework is an interesting area for future work.

The same approach might be used in machine vision to build highly general statistical models for particular objects based on a set of training images. A classifier may then build as for handwritten digits. Applied to images CNMs might also find application in image restoration or perhaps in the area of digital water-marking.

Our development in chapter 2 showed that Baum-Welch/EM may be used to estimate the parameters of non-causal models as well. In many applications, including some of those mentioned above, a non-causal model may suffice. For images this corresponds to the modeling of a pixel using a context that entirely surrounds it. Baum-Welch/EM is certainly easier to implement than the approach of this chapter and may therefore represent the technique of choice when implementing non-causal CNM models.

Finally we point out that while this chapter has focused on ML estimation, techniques similar to those of chapter 2 might be developed to implement alternative estimation criteria.

Bibliography

[Act70] Forman S. Acton. *Numerical Methods that Work*. Harper and Row, 1970.

[Bau72] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[BBS88] H. A. P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Trans. Automatic Control*, pages 780–783, August 1988.

[BE67] L. E. Baum and J. E. Eagon. An inequality with application to statistical estimation for probabalistic functions of a Markov process and to models for ecology. *Bull. AMS*, 73:360–363, 1967.

[Ber79] J. Berstel. *Transductions and Context-Free Languages*. Teubner, Stuttgart, 1979.

[BF96] Y. Bengio and P. Frasconi. Input/output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5), 1996.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo codes (1). In *Proc. ICC'93*, pages 1064–1070, Geneva, Switzerland, 1993.

[BPSW70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math Stat.*, 41:164–171, 1970.

[Bre73] Richard P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, 1973.

[Bro87] Peter F. Brown. *Acoustic-phonetic modeling problem in automatic speech recognition.* PhD thesis, Carnegie-Mellon University, 1987.

[BSL93] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software*, chapter 11, pages 461–465. Artech House, Boston, London, 1993.

[CJ93] Rama Chellappa and Anil Jain, editors. *Markov Random Fields: Theory and Application.* Academic Press, 1993.

[Cov84] Thomas M. Cover. An algorithm for maximizing expected log investment return. *IEEE Transactions on Information Theory*, 1984.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley, 1991.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc., 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Ser. B (methodological)*, 39:1–38, 1977.

[FJM80] F F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, volume Amsterdam, May 21–23, pages 381–397. North Holland, 1980.

[GHE96] S. Greenberg, J. Hollenbach, and D. Ellis. Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus. In *Proc. ICSLP*, Philadelphia, October 1996.

[GHM95] J. Godfrey, E. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *Proc. IEEE ICASSP*, pages 517–520, 1995.

[HAJ90] X. D. Huang, Y. Ariki, and M. A. Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[HD80] Patrick A. V. Hall and Geoff R. Dowling. Approximate string matching. *Computing Surveys*, 12(4):381–402, December 1980.

[HJ89] X. D. Huang and M. A. Jack. Unified modelling of vector quantization and hidden markov models using semi-continuous hidden markov models. In *Proc. ICASSP*, pages 639–642, 1989.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Lau96] Steffen L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[LDC95] COMLEX pronouncing lexicon, version 0.2. Linguistic Data Consortium LDC95L3, July 1995.

[Lev66] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady 10*, 10:707–710, 1966.

[Lev86] S. E. Levinson. Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech and Language*, 1:29–45, 1986.

[MMC96] Robert J. McEliece, D. J. C. MacKay, and J.F. Cheng. Turbo decoding as an instance of pearl's 'belief propagation' algorithm. *Submitted to IEEE Journal on Selected Areas in Communication*, 1996.

[OGV93] José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5), May 1993.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[PFTV88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C – The Art of Scientific Computing*, chapter 10. Cambridge University Press, 1988.

[Por88] A. B. Poritz. Hidden Markov models: a guided tour. In *Proc. ICASSP-88*, pages 7–13, 1988.

[RC85] M. J. Russell and A. E. Cook. Experimental evaluation of duration modelling techniques for automatic speech recognition. In *Proc. ICASSP-85*, pages 5–8, 1985.

[RJLS85] L. R. Rabiner, B. H. Juang, S. E. Levinson, and M. M. Sondhi. Recognition of isolated digits using hidden markov models with continuous mixture densities. *AT&T Technical Journal*, 1985.

[RW84] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26:195–239, 1984.

[RY94] Eric Sven Ristad and Peter N. Yianilos. Probability value library. Technical report, Princeton University, Department of Computer Science, 1994.

[RY96a] Eric Sven Ristad and Peter N. Yianilos. Finite growth models. Technical Report 533-96, Princeton University, Department of Computer Science, 1996.

[RY96b] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. Technical report, Princeton University, Department of Computer Science, 1996.

[RY96c] Eric Sven Ristad and Peter N. Yianilos. On the strange *a posteriori* degeneracy of normal mixtures, and related reparameterization theorems.

Technical report, Princeton University, Department of Computer Science, 1996.

[RY97a] Eric S. Ristad and Peter N. Yianilos. Learning string edit distance. In *(to appear) The Fourteenth International Conference on Machine Learning (ICML'97)*, 1997.

[RY97b] Eric Sven Ristad and Peter N. Yianilos. Library of practical abstractions, release 1.2. ftp://ftp.cs.princeton.edu/pub/packages/libpa, March 1997.

[SHJ96] Padhraic Smith, David Heckerman, and Michael I. Jordan. Probabilistic independence networks for hidden probability models. Technical Report MSR-TR-93-03, Microsoft Research, 1996.

[SK83] D. Sankoff and J. B. Kruskal. *Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983, 1983.

[Wib96] Niclas Wiberg. *Codes and Decoding on General Graphs.* PhD thesis, Linköping Studies in Science and Technology, Sweden, 1996. No. 440.

[WLK95] N. Wiberg, H. Loeliger, and R. Kotter. Codes and iterative decoding on general graphs. *European Transactions on Telecommunications*, 6(5):513–526, October 1995.