GIBBS-MARKOV MODELS

John D. Lafferty
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217

Abstract

In this paper we present a framework for building probabilistic automata parameterized by context-dependent probabilities. Gibbs distributions are used to model state transitions and output generation, and parameter estimation is carried out using an EM algorithm where the M-step uses a generalized iterative scaling procedure. We discuss relations with certain classes of stochastic feedforward neural networks, a geometric interpretation for parameter estimation, and a simple example of a statistical language model constructed using this methodology.

1. Introduction

Standard statistical approaches to speech and language processing problems use hidden Markov models, or more general probabilistic automata such as stochastic context-free grammars, taking advantage of their well-understood properties and efficient training algorithms. But such models are limited in their ability to incorporate contextual information and long-distance dependencies. Because of the Markov assumption, all predictive information must be encoded into the states. We present a generalization of such models that addresses some of these shortcomings. The models that we describe are related to certain classes of stochastic feedfoward neural networks. They are also closely related to the recent use of exponential models and maximum entropy methods in a variety of natural language processing problems [5, 11, 16]. This paper presents one approach to extending these techniques to incomplete data problems for such applications.

Our basic approach is to consider networks for which there is an *embedded* Markov model corresponding to either a finite-state or context-free grammar. Our training procedure is an EM algorithm where in the M-step we use a generalized iterative scaling algorithm for constructing maximum entropy distributions. If the connectivity of the network is sufficiently sparse, both the E-step and M-step can be carried out efficiently. For certain classes of language models the M-step can be efficiently implemented using a method that we call the *cluster expansion* due to its resemblance to series expansion methods in statistical physics. The presence of an embedded Markov model enables further improvements in the efficiency of the training algorithm.

Section 2 of this paper gives a brief overview of the strengths and weaknesses of the hidden Markov model approach, and reviews the relevant characteristics of stochastic neural networks. Section 3 presents the notion of a Gibbs-Markov model. Section 4 discusses the training problem for Gibbs-Markov models, showing that it can be expressed in terms of maximizing an auxiliary function, and Section 5 discusses a geometric interpretation for this algorithm. Section 6 outlines a simple example of how these techniques can be applied to building statistical language models for speech recognition and understanding tasks.

2. Stochastic Networks and Grammars

One of the most common approaches to constructing statistical models of language for applications such as speech and language processing is to design and train a hidden Markov model (HMM) or stochastic context-free grammar [2, 3, 10]. This methodology is also currently being applied in various problems in computational biology [14, 20]. HMM's provide a simple framework for assigning probabilities to the output of an arbitrary finite-state machine. The basic appeal is that the automaton, whether discovered automatically or encoded man-

Research supported in part by NSF and ARPA under grant IRI-9314969 and the ATR Interpreting Telecommunications Research Laboratories.

ually, represents important aspects of the semantics of the problem, and the statistical model then automatically trains the operation of the machine to fit the task. Formally, an HMM $\mathcal{M}$ is determined by a finite set of states $\mathcal{S} = \{s_i\}$, a finite output alphabet $\mathcal{O} = \{o_i\}$, a state transition matrix $A_{ij} = P(S_t = s_i \,|\, S_{t-1} = s_j)$, an output generation matrix $B_{ij} = P(O_t = o_i \,|\, S_t = s_j)$, and an initial state distribution vector $\pi_i = P(S_0 = s_i)$. The Markov property translates into the algebraic statement

$$P(o_1 o_2 \cdots o_t \,|\, \mathcal{M}) = 1 \cdot (T(o_1)T(o_2) \cdots T(o_t)\,\pi)$$

where $T(o_k)$ is the matrix of probabilities $T(o_k)_{ij} = A_{ij}B_{ki}$.

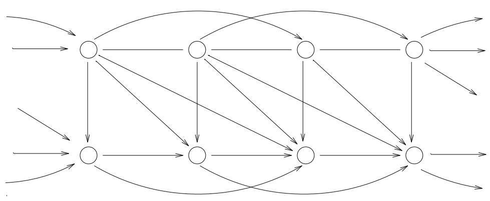
Two of the main strengths of HMM's are that there are efficient algorithms for maximum likelihood training (the Baum-Welch algorithm, a special case of the EM algorithm) and for finding the most probable state sequence for a given output string (the Viterbi algorithm). As an indication of the simplicity of HMM's, it is known that the problem of determining when two HMM's are probabilistically equivalent can be decided in polynomial time [4]. A thorough discussion of the mechanics of training and decoding HMM's, as well as a discussion of HMM/neural network hybrids related to Gibbs-Markov models is given in [18].

HMM's and stochastic grammars are fundamentally limited, however, in their modeling capabilities. The Markov assumption, which enables the E-step to be carried out efficiently, forces all predictive information to be encoded into the states. This can be a severe limitation when the objective of the model is to achieve low entropy (high compression) rates; this is the case when used for modeling language in speech recognition. Put differently, recall that a finite-state automaton may be expressed in terms of phrase structure rules of the form $A \to aB$ and $A \to a$, and that a pushdown automaton may be expressed in terms of phrase structure rules of the form $A \to BC$ and $A \to a$, where A, B and C are nonterminals, representing states, and a is a terminal, representing an arbitrary output symbol. An HMM or stochastic grammar assigns a probability to each derivation by making an independence assumption. In particular, the Markov property can be expressed by the assumption that the probability of a derivation $S \to \alpha_1 \to \cdots \to \alpha_n = o_1 o_2 \cdots o_N$ that results from the application of a sequence of rules $A_i \to \beta_i$ in the grammar can be expressed as

$$\begin{aligned} &P(S \to \alpha_1 \to \cdots \to \alpha_n = o_1 \cdots o_N) \\ &\quad = \prod_{i=1}^{n} P(A_i \to \beta_i \,|\, A_i \textit{ is leftmost nonterminal}) \end{aligned}$$

Thus, the probability of a nonterminal rewriting is independent of the context in which it appears. As a consequence, stochastic grammars cannot model lexical and long-distance dependencies. While the independence assumption is convenient for encoding the grammar, it is often a gross approximation when extended to the probabilistic model. As one example of an attempt to relax the Markov assumption, Black *et al.* [6] used CART to estimate context-dependent rewrite probabilities. In this paper we present another approach to parameterizing and estimating context-dependent probabilities.

Certain classes of stochastic neural networks offer advantages over hidden Markov models because of their inherent ability to model contextual dependencies. For example, *sigmoid belief networks*, a special case of more general graphical models [17], are feed-forward networks with pairwise interactions and binary values taken by the hidden and visible units. Such a network might be represented schematically as follows:



The upper row of states represents units which are *hidden* or unobserved in the training samples, and the bottom row of states represents units which are *observed*, corresponding to the output symbols in an HMM. The states are ordered, and state S_i can be connected to state S_j only if $i < j$. The conditional probability that state S_j has value 1 is given by

$$p(S_j = 1 \,|\, S_1 \cdots S_{j-1}) = \frac{e^{\sum_{i<j} \lambda_{ij} S_i}}{1 + e^{\sum_{i<j} \lambda_{ij} S_i}} = \sigma(\sum_{i<j} \lambda_{ij} S_i)$$

where $\sigma(x) = 1/(1 + e^{-x})$. Training the weights λ_{ij} can be carried out using the EM algorithm. However, when the network is highly connected, the E-step is intractable since it involves summing over an exponentially large number of hidden configurations. To surmount this difficulty, both Gibbs sampling [19] and mean field approximation techniques [21] have been proposed for approximating the expectations that must be computed.

For language processing problems, Gibbs sampling is typically not a feasible alternative. The values that the hidden and visible states can assume is given by vocabularies that are typically on the order of 10,000-100,000 entries in size. Even for character-based models, Gibbs sampling can be computationally intensive [11]. Furthermore, the mean field approximation is not appropriate for such models because the number of states is so large. Such approximations may be more suitable to modeling nucleotide sequences in computational biology.

3. Gibbs-Markov Models

We use the term *Gibbs-Markov model* (GMM) to describe a statistical model for which there is an underlying automaton, as for a hidden Markov model or stochastic grammar, but where the state transition and output symbol generation probabilities are given by Gibbs distributions. The *features* (or sufficient statistics) of the Gibbs distributions allow the probabilities to be context-dependent. The simplest possible GMM for a given automaton is simply an HMM, and each GMM has an *embedded* HMM in the sense that by holding the context-dependent parameters fixed, the model is equivalent to an HMM. (Since the form and basic algorithms for hidden Markov models and stochastic context-free grammars are essentially the same, we will usually not make a distinction between them and will use the acronym HMM to refer to both.)

More formally, a GMM $\mathcal{G}$ for an underlying finite-state machine is determined by a finite set of states $\mathcal{S} = \{s_i\}$, a finite output alphabet $\mathcal{O} = \{o_i\}$, a set of *state transition features*

$$f_\alpha : \mathcal{S} \times (\mathcal{S} \times \mathcal{O})^n \to \{0,1\},$$

together with associated *state transition parameters* λ_α, and a set of *output generation features*

$$g_\gamma : \mathcal{O} \times \mathcal{S} \times (\mathcal{S} \times \mathcal{O})^n \to \{0,1\},$$

together with associated *output generation parameters* λ_γ.

We let S_t and O_t be random variables representing the state and output symbol at time t. The *history at time* t, denoted $\mathcal{H}_t$, is the vector-valued random variable (equivalently σ-algebra)

$$\mathcal{H}_t = (S_0, O_0, \ldots, S_{t-1}, O_{t-1}) \in (\mathcal{S} \times \mathcal{O})^t .$$

The probability distribution of S_t given the history at time t is given by

$$P(S_t \mid \mathcal{H}_t) = \frac{1}{Z(\mathcal{H}_t)} e^{\lambda \cdot f(S_t, \mathcal{H}_t)}.$$

The probability distribution of O_t given the state S_t and the history at time t is given by

$$P(O_t \mid S_t, \mathcal{H}_t) = \frac{1}{Z(S_t, \mathcal{H}_t)} e^{\lambda \cdot g(O_t, S_t, \mathcal{H}_t)}.$$

Here $\lambda \cdot f$ and $\lambda \cdot g$ denote the sums $\sum_\alpha \lambda_\alpha f_\alpha$ and $\sum_\gamma \lambda_\gamma g_\gamma$, taken over the appropriate index sets. Some of the transition and generation probabilities may be constrained to equal zero, in accordance with the underlying grammar.

To express an HMM in these terms, the indices α are parametrized by state bigrams (s_i, s_j) and the state transition features are taken to be

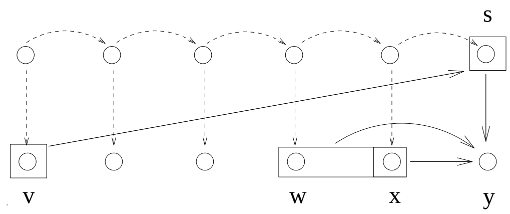
$$f_{s_i,s_j}(S_t, \mathcal{H}_t) = \langle S_t = s_i \rangle \langle S_{t-1} = s_j \rangle ,$$

where $\langle \mathcal{P} \rangle = 1$ if the predicate $\mathcal{P}$ is true and $\langle \mathcal{P} \rangle = 0$ if $\mathcal{P}$ is false. The indices γ are parameterized by state-output bigrams (o_i, s_j) and the output generation features are taken to be

$$g_{o_i,s_j}(O_t, S_t, \mathcal{H}_t) = \langle O_t = o_i \rangle \langle S_t = s_j \rangle .$$

In this case we obtain an HMM with, in terms of our earlier notation, $A_{ij} = e^{\lambda_{s_i,s_j}}$ and $B_{ij} = e^{\lambda_{o_i,s_j}}$ (for an arbitrary initial state distribution). We refer to this as the *embedded HMM*. With any additional features, the model is by definition non-Markovian.

To illustrate the properties of such models when applied to language, consider the GMM suggested in the following figure.



In this figure the top layer of states is hidden, and the bottom layer of states is observed. The dashed arcs indicate the transitions for the embedded HMM; there are features f_{s_i,s_j} and g_{o_i,s_j} as above. The boxed states and solid lines indicate additional features. In addition to predicting output y in the presence of the state s, y can also be predicted from the immediately preceding output symbols, w and x. This is accomplished with *bigram* and *trigram* features of the form

$$\begin{aligned} g_{x,y}(O_t, S_t, \mathcal{H}_t) &= \langle O_t = y \rangle \langle O_{t-1} = x \rangle \\ g_{w,x,y}(O_t, S_t, \mathcal{H}_t) &= \langle O_t = y \rangle \langle O_{t-1} = x \rangle \langle O_{t-2} = w \rangle. \end{aligned}$$

The longer distance connection between v and the state s represents a feature of the form

$$g_{v,s,y}(O_t, S_t, \mathcal{H}_t) = \\ \langle O_t = y \rangle \langle S_{t-1} = s \rangle \langle v \textit{ appears in last 100 words} \rangle.$$

If the state S_t encodes information such as the "part-of-speech" of the word being generated, then such a feature allows a more plausible notion of "trigger words" compared to the use of long-distance features in [16], by allowing words to be triggered only in the appropriate grammatical contexts.

An advantage of such a model is that the embedded HMM can be trained first, and then used as the starting point for the full EM training. If $\tilde{p}(x)$ is the incomplete density, recall that the standard EM procedure obtains a lower bound on the change in log-likelihood as

$$\sum_x \tilde{p}(x) \log \frac{p_{\lambda'}(x)}{p_\lambda(x)} \geq \sum_x \tilde{p}(x) \sum_h p_\lambda(h \,|\, x) \log \frac{p_{\lambda'}(x,h)}{p_\lambda(x,h)} \equiv A(\lambda', \lambda)\,,$$

a simple consequence of Jensen's inequality. Thus, the auxiliary function $A(\lambda', \lambda)$ differs from the change in log-likelihood by the average Kullback-Leibler divergence between the conditional distributions $p_\lambda(h \,|\, x)$ and $p_{\lambda'}(h \,|\, x)$; that is,

$$\sum_x \tilde{p}(x) \log \frac{p_{\lambda'}(x)}{p_\lambda(x)} - A(\lambda', \lambda) = \sum_x \tilde{p}(x) \sum_h p_\lambda(h \,|\, x) \log \frac{p_\lambda(x,h)}{p_{\lambda'}(x,h)}.$$

For models such as the one illustrated above, we can anticipate that this expected divergence is small since the embedded HMM differs from the full model only in terms of its output generation probabilities. Thus, since maximizing the auxiliary function is an attempt to achieve the largest possible increase in log-likelihood while keeping the posterior state distributions as similar as possible, the embedded model can provide an excellent approximation for estimating the context-dependent model.

While the auxiliary function $A(\lambda', \lambda)$ given above is easily maximized for a hidden Markov model, this is not so for a general Gibbs-Markov model. In the following section we explain how additional lower bounds can be obtained to yield practical auxiliary functions.

4. Auxiliary Functions and Iterative Scaling

The M-step in the EM training for HMM's can typically be carried out efficiently. In fact, for non-parametric HMM's the auxiliary function $A(\phi', \phi)$ is convex in ϕ_α, and the equations $(\partial/\partial\phi'_\alpha)A(\phi', \phi) = 0$ can be solved in closed form. Once the hidden data is exposed, maximum likelihood estimation amounts to simple counting. The situation is much more interesting and challenging for a general GMM.

We will derive an auxiliary function that can be efficiently maximized by obtaining an additional lower bound on the change in log-likelihood. Let $p_\lambda(x,h) = Z_\lambda^{-1} e^{\lambda \cdot f(x,h)}$. (In this section we make no distinction between features f_α and g_γ.) Then

$$\begin{aligned}\sum_x \tilde{p}(x) \log \frac{p_{\lambda'}(x)}{p_\lambda(x)} &\geq \sum_x \tilde{p}(x) \sum_h p_\lambda(h \,|\, x) \log \frac{p_{\lambda'}(x,h)}{p_\lambda(x,h)} \\ &\geq \sum_x \tilde{p}(x) \sum_h p_\lambda(h \,|\, x) \left((\lambda' - \lambda) \cdot f(x,h)\right) - \frac{Z_{\lambda'}}{Z_\lambda}.\end{aligned}$$

The second inequality is obtained using the fact that $\log(x) \leq x - 1$. To simplify notation, let $q[\,f\,]$ denote the expectation of f with respect to q, let $\tilde{p}_\lambda(x,h) = \tilde{p}(x) p_\lambda(h \,|\, x)$, and $\Delta\lambda = \lambda' - \lambda$. The above inequality can now be expressed as

$$\sum_x \tilde{p}(x) \log \frac{p_{\lambda'}(x)}{p_\lambda(x)} \geq \tilde{p}_\lambda[\,\Delta\lambda \cdot f\,] - p_\lambda[\,e^{\Delta\lambda \cdot f}\,]$$

since the ratio of partition functions $Z_{\lambda'}/Z_\lambda$ can be written as an expectation. This does not yet leave us with a satisfactory auxiliary function because if we take partial derivatives and set them to zero, we obtain a system of *coupled* equations, and simply replace one implicit maximization problem with another.

We require an additional piece of notation in order to write the decoupled system. Let $f^\sharp(x,h)$ be defined by $f^\sharp(x,h) = \sum_\alpha f_\alpha(x,h)$ and let $f(\alpha \,|\, x,h) = f_\alpha(x,h)/f^\sharp(x,h)$. Since

$$e^{\sum_\alpha t_\alpha x_\alpha} \leq \sum_\alpha t_\alpha \, e^{x_\alpha}$$

whenever $t_\alpha \geq 0$ and $\sum_\alpha t_\alpha = 1$, we have that

$$\begin{aligned}\sum_x \tilde{p}(x) \log \frac{p_{\lambda'}(x)}{p_\lambda(x)} &\geq \tilde{p}_\lambda[\,\Delta\lambda \cdot f\,] - \sum_\alpha \sum_{x,h} p_\lambda(x,h) f(\alpha \,|\, x,h)\, e^{\Delta\lambda f^\sharp(x,h)} \\ &\equiv \mathcal{A}(\lambda', \lambda)\,.\end{aligned}$$

Taking derivatives, we obtain the decoupled system

$$\frac{\partial \mathcal{A}(\lambda', \lambda)}{\partial \lambda_\alpha} = \tilde{p}_\lambda[\,f_\alpha\,] - p_\lambda[\,f_\alpha \, e^{\Delta\lambda_\alpha f^\sharp}\,] = 0\,.$$

This is a family of polynomials in $\Delta\phi_\alpha = e^{\Delta\lambda_\alpha}$ that can be efficiently solved (in parallel) using Newton's algorithm.

The resulting algorithm for iteratively updating the parameters λ_α is a generalization of the Darroch-Ratcliff iterative scaling procedure [9]; it is discussed in detail in [11]. It is an improvement over the Darroch-Ratcliff algorithm for problems where a large number of parameters need to be estimated because it does not require that the features f_α sum to a constant. The algorithm is often easier to implement and can yield faster convergence rates than the Darroch-Ratcliff algorithm (see Section 6). Our approach to deriving an auxiliary function simplifies the training procedure. Because the auxiliary function can be maximized directly, there is little distinction between the E-steps and M-steps.

5. A Geometric Interpretation

In this section we give a geometric interpretation of the parameter estimation algorithm just outlined. In particular, we adopt Csiszár's formulation [7] of iterative scaling, using the techniques of alternating projection and divergence geometry, to the situation where the features are not normalized.

Let $\Omega = (\mathcal{O} \times \mathcal{S})^n$ be the complete data event space, and let $\mathcal{I} = \{\alpha\}$ be the set of indices for the features $\{f_\alpha\}$. The first step is to extend from probabilities on Ω to probabilities on $\Omega \times \mathcal{I}$, defining a map $f\!\uparrow : \mathrm{Prob}(\Omega) \longrightarrow \mathrm{Prob}(\Omega \times \mathcal{I})$ by

$$(f\!\uparrow\! p)(\omega, \alpha) = p(\omega) f(\alpha \,|\, \omega)\,.$$

The map $f\!\uparrow$ is an "isometry" with respect to the Kullback-Leibler divergence:

$$D(f\!\uparrow\! p \,\|\, f\!\uparrow\! q) = D(p \,\|\, q)\,.$$

If $\mathcal{P}$ is a closed, convex set of probabilities we define a projection $\pi_{\mathcal{P}}$ by

$$\pi_{\mathcal{P}}(q) = \arg\min_{p \in \mathcal{P}} D(p \,\|\, q)\,.$$

Let $\mathcal{L}(f)$ be the linear family of probabilities p on Ω that satisfy $p[\,f_\alpha\,] = \tilde{p}_\lambda[\,f_\alpha\,]$ for all α. While $\tilde{p}_\lambda$ is arbitrary, it is to be thought of as the complete data distribution obtained in an E-step. The goal is to find the member of $\mathcal{L}(f)$ that is closest, with respect to the Kullback-Leibler divergence, to $\tilde{p}_\lambda$. Finally, let $\mathcal{L}^{\uparrow}(f)$ be the linear family of probabilities of the form $p = f\!\uparrow\! q$ that satisfy $\sum_\omega p(\omega, \alpha) f^{\sharp}(\omega) = \tilde{p}_\lambda[\,f_\alpha\,]$.

With these definitions, it is easy to see that

$$\pi_{\mathcal{L}^{\uparrow}(f)} \circ f\!\uparrow = f\!\uparrow \circ\, \pi_{\mathcal{L}(f)}\,.$$

Furthermore, the linear family $\mathcal{L}^{\uparrow}(f)$ can be expressed as the intersection $\mathcal{L}^{\uparrow}(f) = \mathcal{L}_1 \cap \mathcal{L}_2$, where

$$\begin{aligned}
\mathcal{L}_1 &= \{p \mid \sum_\omega p(\omega, \alpha)\, f^{\sharp}(\omega) = \tilde{p}_\lambda[\,f_\alpha\,],\ \textit{for each}\ \alpha\} \\
\mathcal{L}_2 &= \{p \mid p(\omega, \alpha) = p(\omega)\, f(\alpha \,|\, \omega)\} = f\!\uparrow \mathrm{Prob}(\Omega)\,.
\end{aligned}$$

The key fact is that

$$\pi_{\mathcal{L}^{\uparrow}(f)}(f\!\uparrow\! p) = f\!\uparrow \left(\pi_{\mathcal{L}(f)} p\right) = \lim_{n \to \infty} (\pi_2 \circ \pi_1)^n (f\!\uparrow\! p)$$

where $\pi_i = \pi_{\mathcal{L}_i}$. This is a consequence of general results of Csiszár.

The reason this represents progress is that the projections π_1 and π_2 can be calculated explicitly. In fact, projection π_1 is

$$(\pi_1\, q)(\omega, \alpha) = \frac{1}{Z_\phi} q(\omega, \alpha)\, \phi_\alpha^{f^{\sharp}(\omega)}$$

where Z_ϕ is a normalizing constant and $\phi_\alpha \geq 0$ is chosen to satisfy the equation

$$\frac{1}{Z_\phi} \sum_{\omega \in \Omega} q(\omega, \alpha)\, f^{\sharp}(\omega) \phi_\alpha^{f^{\sharp}(\omega)} = \tilde{p}_\lambda[\,f_\alpha\,]\,.$$

Note that because of the normalization term Z_ϕ, the equations above are coupled; we will later remove this coupling. The projection π_2 is given by

$$(\pi_2\, q)(\omega, \alpha) = f\!\uparrow \left(\frac{1}{Z} q(\omega) \prod_\theta \left[\frac{q(\theta \,|\, \omega)}{f(\theta \,|\, \omega)} \right]^{f(\theta \,|\, \omega)} \right)$$

where Z is again a normalizing constant.

Given these formulæ, it is straightforward to calculate that the composition $\pi_2 \circ \pi_1$, evaluated on $f\!\uparrow\!\mathrm{Prob}(\Omega)$, is given by $\pi_2 \circ \pi_1 (f\!\uparrow\! p) = f\!\uparrow\! q$ where

$$q(\omega) = \frac{1}{Z_\phi} p(\omega) \prod_\alpha \phi_\alpha^{f_\alpha(\omega)}$$

with the parameters $\phi_\alpha \geq 0$ uniquely determined as the roots of the coupled equations

$$\frac{1}{Z_\phi} \sum_\omega p(\omega) f_\alpha(\omega) \phi_\alpha^{f^{\sharp}(\omega)} = \tilde{p}_\lambda[\,f_\alpha\,]\,.$$

From the above analysis it can be seen that the normalization $1/Z_\phi$, though necessary to make the projection π_1 a probability distribution, does not play an essential role. Were it not for this constraint, the coupled equations above would in fact be uncoupled, and could each be solved independently of the others. It turns out that by relaxing the constraint that $\pi_1 q$ is a probability distribution, we can still obtain convergence by using the *extended Kullback-Leibler divergence*

$$\widetilde{D}(m \,\|\, n) = \sum_{x \in X} \left[m(x) \log \frac{m(x)}{n(x)} - m(x) + n(x) \right]$$

for positive measures m and n. The extended Kullback-Leibler divergence $\widetilde{D}$ enjoys many of the same properties as the standard divergence; in particular, $\widetilde{D}(m \,\|\, n) \geq 0$ with equality if and only if $m = n$, and $\widetilde{D}(m \,\|\, n)$ is convex in m and n. Without the constraint that $\pi_1\, q$ is normalized, it is calculated as

$$(\pi_1\, q)(\omega, \alpha) = q(\omega, \alpha)\, \phi_\alpha^{f^\sharp(\omega)}$$

where $\phi_\alpha \geq 0$ is chosen to satisfy the equation

$$\sum_{\omega \in \Omega} q(\omega, \alpha)\, f_\alpha(\omega) \phi_\alpha^{f^\sharp(\omega)} = \tilde{p}_\lambda[\, f_\alpha \,]\,.$$

This rederives the improved iterative scaling algorithm in geometric terms.

The EM algorithm can in general be viewed in geometric terms [1, 8]. When estimating within a (closed) family Q of probabilities, given an incomplete-data density $\tilde{p}$, the E-step is given by the projection

$$\pi_E(q) = \arg\min_{p \in \mathcal{P}} D(p \,\|\, q)$$

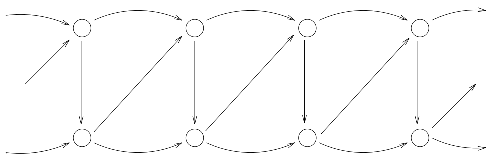
where $\mathcal{P}$ is the convex subset of all probabilities whose incomplete-data marginal agrees with $\tilde{p}$. The M-step is equivalent to the projection

$$\pi_M(p) = \arg\min_{q \in Q} D(p \,\|\, q)\,.$$

As we have seen, when Q is the exponential family specified by linear constraints, the projection π_M can itself be calculated by alternating projection onto two intersecting linear families. This provides an intuitive, geometric framework for parameter estimation which is complementary to the simpler auxiliary function formulation given in the previous section.

6. An Example Language Model

As a simple yet realistic example of a language model constructed as a GMM, let us consider a model in which a word is predicted from both the current state and the previous word, and in which a state is predicted from the previous state as well as the previous word. Schematically, this is shown in the figure below.



We use x and y to denote arbitrary words, and s and t to denote states. We also adopt a notation for which $\langle s \sim t \rangle$ is true when $s \to t$ is a valid state transition and $\langle x \sim y \rangle$ is true when (x, y) is a valid bigram for the finite-state grammar. Including "bias" terms into the model, the Gibbs distributions for the state transition and output generation components of the model are given as

$$\begin{aligned} p_\lambda(y \,|\, x, s) &= \frac{1}{Z_\lambda(x, s)} e^{\lambda_{x,y} \langle x \sim y \rangle + \lambda_{s,y} \langle s \sim y \rangle + \lambda_y} \\ p_\lambda(s \,|\, x, t) &= \frac{1}{Z_\lambda(x, t)} e^{\lambda_{t,s} \langle t \sim s \rangle + \lambda_{x,s} \langle x \sim s \rangle + \lambda_s}\,. \end{aligned}$$

When the state s encodes the "part-of-speech" of the word it generates, the embedded HMM for this model is the familiar class-bigram model that is commonly used for tagging text. Class-based n-gram models such as this do not achieve perplexities as low as those for word-based n-gram models, since all of the predictive information is channeled through the state. But the approach that uses Gibbs-Markov modeling to predict words with context-dependent parameters in addition to class-based parameters will attain compression rates at least as high as those of their n-gram cousins, at the same time that it tags text with high accuracy. In addition, a GMM parameterized as above (and trained as below) is more *smooth* than a standard HMM. This allows the model to make better use of its training data, making the reservation of held-out data for deleted interpolation smoothing unnecessary. The computational burden of parameter estimation is, however, significantly increased.

Suppose that $\{X\}$ is a set of sentences comprising the corpus available for training the model. We denote by $\tilde{p}(X)$ the empirical distribution of these sentences, and write $X = x_1 x_2 \cdots x_{N(X)}$. Let $\phi_\alpha = e^{\lambda_\alpha}$. The iterative scaling algorithm chooses factors $\Delta\phi_\alpha$ and updates the parameters by setting $\phi_\alpha \mapsto \phi_\alpha \Delta\phi_\alpha$. Let $\phi_{x,y}$ be the parameter for the word bigram (x, y) and let $\phi_{t,s}$ be the parameter for the state transition $t \to s$. The parameters $\phi_{x,y}$ are estimated only for those pairs (x, y) that appear as bigrams in the corpus. Similarly, we estimate parameters $\phi_{s,t}$ and $\phi_{s,x}$ only for those states s, t and those words x that are allowed by the finite-state grammar.

According to the algorithm described in the previous two sections, the scaling terms $\Delta\phi$ are calculated in the following manner. The bigram parameter update $\Delta\phi_{x,y}$ is chosen as the unique positive solution of the polynomial

$$\sum_s \tilde{p}_\lambda(x,s)\, p(y\,|\,x,s)\, \Delta\phi_{x,y}^{f^\sharp(y,x,s)} = \tilde{p}_\lambda[\,\langle x \sim y\rangle\,]$$

where the probability $\tilde{p}_\lambda(x,s)$ of the history (x,s) is given by

$$\sum_X \frac{\tilde{p}(X)}{p_\lambda(X)} \sum_{t,i:x_{i-1}=x} \alpha_{i-1}(t)\, p_\lambda(s\,|\,x,t)\, p_\lambda(x_i\,|\,x,s)\, \beta_i(s)$$

and $\tilde{p}_\lambda[\,\langle x \sim y\rangle\,] = \tilde{p}(x,y)$ is simply the bigram probability as it appears in the training data. The quantities $\alpha_i(s)$ and $\beta_i(s)$ are the usual forward and backward probabilities as they are calculated in the Baum-Welch algorithm.

In the same way, the state transition parameter update $\Delta\phi_{t,s}$ is chosen to solve the equation

$$\sum_x \tilde{p}_\lambda(x,t)\, p(s\,|\,x,t)\, \Delta\phi_{t,s}^{f^\sharp(s,x,t)} = \tilde{p}_\lambda[\,\langle t \sim s\rangle\,]$$

where now the probability $\tilde{p}_\lambda(x,t)$ of the history (x,t) is calculated as

$$\sum_X \frac{\tilde{p}(X)}{p_\lambda(X)} \sum_{i:x_i=x} \alpha_i(t)\, \beta_i(t)$$

and $\tilde{p}_\lambda[\,\langle t \sim s\rangle\,]$ is given by

$$\sum_X \frac{\tilde{p}(X)}{p_\lambda(X)} \sum_i \alpha_{i-1}(t)\, p_\lambda(s\,|\,x_{i-1},t)\, p_\lambda(x_i\,|\,x_{i-1},s)\, \beta_i(s).$$

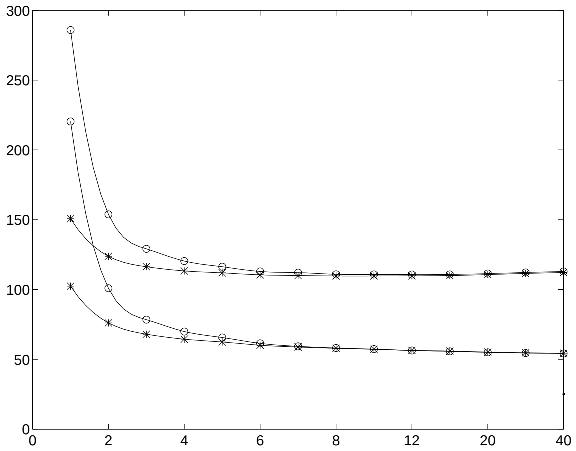
Note that by an abuse of notation we have written $\tilde{p}_\lambda(x,s)$ and $\tilde{p}_\lambda(x,t)$ to denote probabilities of histories where states s and t correspond to different relative indices. The updates for the parameters $\phi_{s,y}$, $\phi_{x,s}$, ϕ_s, and ϕ_x are computed in a similar fashion.

An advantage of the formalism is that the embedded HMM can be trained first, using the familiar Baum-Welch approach. Then this model can be used to train the context-dependent parameters of the full model. The the equations for this model can be efficiently computed and solved using a technique that we call the *cluster expansion*, borrowing the name of a closely related method for calculating partition functions in statistical mechanics. This is described in detail in [15]. While the technique does nothing to address the inherent computational difficulty of carrying out the E-step for a highly connected network, it can be a very useful tool for efficient estimation of a large class of models that arise in applications.

We have built a language model of this form on the *Switchboard* corpus [12] for use in a speech recognition system. This corpus includes approximately three million words of text, transcribed from more than 150 hours of speech collected from telephone conversations restricted to 70 different topics. In addition to the parameters described above, we included topic-dependent unigram parameters corresponding to features

$$f_{t,w}(O_t, S_t, \mathcal{H}_t) = \langle O_t = w\rangle\, \langle \textit{the current topic is } t\rangle\,.$$

The resulting model achieves significantly lower perplexities on test data, between 7% and 18% depending on the topic. The graphs below show training and test set perplexities as a function of iteration for a model of this form trained on the Switchboard corpus for a vocabulary of approximately 5000 words.



The lower pair of curves compares the generalized iterative scaling algorithm derived in Sections 4 and 5 against the Darroch-Ratcliff algorithm, and demonstrates that the improved algorithm converges more quickly. This speed-up in the convergence rate is also demonstrated on the test data, as shown on the upper pair of graphs. These graphs also demonstrate the smoothness properties of GMM's. While a standard HMM would become sharply overtrained as the model converges on test data, the hierarchical structure of the GMM results in only a slight overfitting.

Acknowledgement

This paper owes much to Stephen and Vincent Della Pietra; thanks for many stimulating conversations on divergence geometry and iterative scaling over the past few years. Thanks are also due to Bernhard Suhm for constructing Switchboard models and running experiments.

References

[1] S. Amari, *Information geometry of the EM and em algorithms for neural networks*, preprint, Department of Mathematical Engineering and Information Physics, University of Tokyo, 1995.

[2] L. Bahl, F. Jelinek, and R. Mercer. *A maximum likelihood approach to continuous speech recognition*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-5(2):179–190, March 1983.

[3] J. Baker, *Trainable grammars for speech recognition*, in Proceedings of the Spring Conference of the Acoustical Society of America, 547–550, Boston, MA, 1979.

[4] V. Balasubramanian, *Equivalence and reduction of hidden Markov models*, Master's thesis, Department of Electrical Engineering and Computer Science, MIT, issued as AI Technical Report 1370, 1993.

[5] A. Berger, S. Della Pietra and V. Della Pietra, *A maximum entropy approach to natural language processing*, Computational Linguistics, to appear.

[6] E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos, *Towards history-based grammars: Using richer models for probabilistic parsing*, Proceedings of the DARPA Speech and Natural Language Workshop, Arden House, Morgan Kaufman Publishers, 1992.

[7] I. Csiszár, *A geometric interpretation of Darroch and Ratcliff's generalized iterative scaling*, The Annals of Statistics, **17**, No. 3, 1409–1413, 1989.

[8] I. Csiszár and G. Tusnády, *Information geometry and alternating minimization procedures*, Statistics & Decisions, Supplement Issue, **1**, 205–237, 1984.

[9] J. Darroch and D. Ratcliff, *Generalized iterative scaling for log-linear models*, Ann. Math. Statist. **43**, 1470–1480, 1972.

[10] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, L. Ures, *Inference and estimation of a long-range trigram model*, Proceedings of the Second International Colloquium on Grammatical Inference and Applications, Lecture Notes in Artificial Intelligence, **862**, Springer-Verlag, 78-92, 1994.

[11] S. Della Pietra, V. Della Pietra and J. Lafferty, *Inducing features of random fields*, Carnegie Mellon technical report CMU-CS-95-144, 1995.

[12] J. Godfrey, E. Holliman, and J. McDaniel, Switchboard: Telephone speech corpus for research development, *Proc. ICASSP-92*, I-517-520, 1992.

[13] F. Jelinek, J. Lafferty, and R. Mercer, *Basic methods of probabilistic context-free grammars*, Speech Recognition and Understanding: Recent Advances, Trends, and Applications, P. Laface and R. De Mori, editors, Springer Verlag, Series F: Computer and Systems Sciences, Volume 75, 1992.

[14] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler, *Hidden Markov models in computational biology: Applications to protein modeling*, Journal of Molecular Biology, **235**:1501–1531, February 1994.

[15] J. Lafferty and B. Suhm, *Cluster expansions and iterative scaling of maximum entropy language models*, in proceedings of the Fifteenth International Workshop on Maximum Entropy and Bayesian Methods, Kluwer Academic Publishers, 1995, to appear.

[16] R. Lau, R. Rosenfeld, and S. Roukos, *Adaptive language modeling using the maximum entropy principle*, Proceedings of the ARPA Human Language Technology Workshop, Morgan Kaufman Publishers, 108–113, 1993.

[17] S. Lauritzen and D. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, J. Roy. Stat. Soc. B **50**: 157–224, 1988.

[18] A. Nádas and R. Mercer, *Hidden Markov models and some connections with artificial neural networks*, in *Mathematical Perspectives on Neural Networks*, P. Smolensky, M. Mozer, and D. Rumelhart, ed., Erlbaum, 1994.

[19] R. Neal, *Connectionist learning of belief networks,* Artificial Intelligence **56**, 71–113, 1992.

[20] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. Underwood and D. Haussler, *Stochastic context-free grammars for tRNA modeling*, Nucleaic Acids Research, **22**(23):5112–5120, 1994.

[21] L. Saul, T. Jaakkola, and M. Jordan, *Mean field theory for sigmoid belief networks*, Computational Cognitive Science Technical Report 9501, MIT, 1995.