Incremental Estimation of Discrete Hidden Markov Models Based on a New Backward Procedure

German Florez-Larrahondo, Susan Bridges and Eric A. Hansen
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
{gf24,bridges,hansen}@cse.msstate.edu

Abstract

We address the problem of learning discrete hidden Markov models from very long sequences of observations. Incremental versions of the Baum-Welch algorithm that approximate the β-values used in the backward procedure are commonly used for this problem, since their memory complexity is independent of the sequence length. We introduce an improved incremental Baum-Welch algorithm with a new backward procedure that approximates the β-values based on a one-step lookahead in the training sequence. We justify the new approach analytically, and report empirical results that show it converges faster than previous incremental algorithms.

Introduction

Hidden Markov models (HMMs) have been successfully applied to modeling tasks in speech recognition, pattern recognition, biological sequence analysis, and other real-world applications. Although many variations of HMMs have been proposed, including coupled Markov models, factorial HMMs, input-output HMMs, and Markov weighted transducers, theoretical and empirical results have shown that traditional HMMs are capable of representing complex probability distributions given enough hidden states and sufficiently rich observations (Bengio 1999).

The algorithms described in this paper estimate discrete-valued stationary signals where the output of the source is categorical. Research on discrete observations is important because "...when the observations are categorical in nature, and the observations are quantitative but fairly small, it is necessary to use models which respect the discrete nature of the data" (MacDonald & Zucchini 1997, p.3). Examples of categorical variables include operating system commands typed by a user on a console, computer security audit events, network requests, online transactions, and DNA bases, among many others.

In many domains, the traditional Baum-Welch learning algorithm is difficult to apply because the length T of the training sequences is very large (or possibly infinite), and the number of states N in the model is relatively small (Warrender, Forrest, & Pearlmutter 1999; Lane 2000; Qiao *et al.* 2002; Florez *et al.* 2005a). Because the time and space complexity of Baum-Welch is $O(N^2T)$, learning in such domains is difficult because of the large values of T. For example, to train HMMs for intrusion detection, Lane (2000, p.57) used sequences of commands generated by UNIX users, where the sequence length varied from "just over 15,000 tokens to well over 100,000". Florez et al. (2005a) collected library function calls from scientific parallel programs for similar training of HMMs, and the average number of calls for an implementation of the LU-Factorization method was more than 20,000, and for a benchmarking application was more than 800,000.

In addition, these models and learning algorithms are generally embedded in complex applications (such as intrusion and fault detection), where impact on the performance of a production system due to training the model needs to be kept as small as possible (Florez *et al.* 2005b; Warrender, Forrest, & Pearlmutter 1999). This motivates research on new learning algorithms that can handle lengthy discrete data streams, but have reduced memory requirements compared to the traditional Baum-Welch algorithm.

Incremental learning algorithms can be used to solve such problems, since they can speed the convergence of the learning process as well as reduce its memory requirements (Gotoh & Silverman 1996; Neal & Hinton 1999). Incremental Baum-Welch algorithms use the *forward-backward* procedure to re-estimate the parameters of the model as soon as new data examples are available. Although the α-values can be computed normally before the end of a training sequence, the β-values cannot. A simple but elegant solution to the incremental learning problem is proposed by Stenger et al. (2001), in which all the β-values are assumed to be 1.0. In this paper, we demonstrate that the convergence behavior of incremental Baum-Welch algorithms can be improved when the β-values are instead approximated by a function that looks ahead one observation in the sequence. Additional improvements are also described.

HMMs are a special case of Bayesian networks, which use *local structure* to reduce the factor N^2 in the complexity of parameter estimation and probabilistic inference (Friedman & Goldszmidt 1999). Although we do not consider factored models in this paper, our work can be viewed as complementary since it reduces the space and time complexity of the traditional Baum-Welch algorithm when the sequence length T is large.

Background

We begin with a brief review of stationary discrete first order hidden Markov models, the Baum-Welch algorithm, and incremental versions of the Baum-Welch algorithm. For a more complete description of HMMs, refer to the work of Rabiner (1989) and MacDonald and Zucchini (1997).

Discrete Hidden Markov Models (HMMs)

Consider a system with N states, indexed $i, j, k, ...$, where each state represents an observable event. Let the state at time t be q_t. Employing a notation similar to Rabiner's, a hidden Markov model λ can be described as a doubly stochastic process with the following elements:

- *N*, the number of states and *M*, the number of distinct observation symbols per state (the alphabet size).
- **A**, the state transition probability distribution with elements a_{ij}, for $1 < i, j < N$.
- **B**, the observation symbol probability distribution with elements $b_j(k)$, for $1 < j < N$ and $1 < k < M$.
- π, the initial state distribution with elements π_i, for $1 < i < N$.

We assume that no previous knowledge of the topology of the model or the meaning of the hidden states is given, and therefore, the task of the learning algorithm is to estimate the parameters of random, ergodic (fully-connected) models.

Offline Estimation of HMMs: The Baum-Welch Algorithm (BW)

The Baum-Welch algorithm (known henceforth as BW) learns the transition and symbol probabilities of an HMM by maximizing $\mathcal{Q}(\lambda, \lambda') = \sum_Q P(O, Q|\lambda) log\,[P(O, Q|\lambda')]$, where λ is the current set of parameters of the model, λ' is the set of reestimated parameters, Q is the set of all possible state sequences and O is the sequence of observations to be learned (Rabiner 1989). Iterative maximization of this function has been proved to lead to an increase in likelihood, *i.e.,* $P(O|\lambda') \geq P(O|\lambda)$ (Rabiner 1989; MacDonald & Zucchini 1997).

The reestimation formulas for BW can be obtained analytically by maximizing $\mathcal{Q}(\lambda, \lambda')$ via Lagrange multipliers, assuming the stochastic constraints of the HMM parameters. A key result from the maximization is the estimation of the probability of being in state i at time t and state j at time $t+1$, given the model λ and the sequence of observations, $\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O, \lambda)$. Defining $\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = i|\lambda)$, and $\beta_{t+1}(i) = P(O_{t+1} O_{t+2} ... O_T|q_t = i, \lambda)$, $\xi_t(i,j)$ can be written as:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (1)$$

The well known *forward* and *backward* procedures find exact values for $\alpha_t(i)$ and $\beta_t(i)$ in $O(N^2 T)$ time. Note that another widely used estimator, the probability of being in state i at time *t* given the model and observation sequence, $\gamma_t(i)$, can be computed based on (1):

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)} \qquad (2)$$

The traditional Baum-Welch algorithm then updates **A**, **B** and π as functions of $\xi_t(i,j)$ and $\gamma_t(i)$. A detailed analysis of BW indicates that its space complexity is $O(N(N+M+TN))$ and its time complexity is $O(N(1+T(M+N))$. But since in most practical applications $T \gg M$, it is widely accepted that the space and time complexity for the traditional batch learning of an HMM is $O(N^2 T)$.

Many variations of the Baum-Welch algorithm have been introduced. Of particular interest is the work of Binder et al. (1997), motivated by the problem of *monitoring* long sequence of observations, in which the space complexity of BW is reduced to $O(N^2 log T)$, at the expense of $O(N^2 T log T)$ time complexity.

Incremental Estimation of HMMs

Incremental learning algorithms for HMMs are an active field of research in the signal processing and control system communities. Such algorithms generally show faster convergence than the standard batch training algorithm (Gotoh & Silverman 1996). Note that theoretical justification for incremental versions of the EM algorithm are given by Neal and Hinton (1999).

An approach to incremental learning adopted by many researchers is to estimate the HMM parameters as soon as new data examples are available using the Baum-Welch algorithm, with the constraints that the α-values can be computed normally, but the β-values cannot because they are associated with the probability of the partial observation from the current event to the end (Stiller & Radons 1999; Stenger *et al.* 2001; Koenig & Simmons 1996). Following this idea, Stiller and Radons (1999) present an incremental estimation algorithm in which the HMM's transition probabilities are not computed directly, but instead, auxiliary variables containing "lifted" parameters are computed, assuming inhomogeneous Markov chains.

A simpler scheme was proposed by Koening and Simmons (1996) for learning of models for robot navigation. They approximate the α-values and β-values using a "sliding window" of training data, reducing the memory requirements of the Baum-Welch algorithm.

Note that Elliot et al. have shown that the backward pass through the data can be eliminated, at the expense of increasing the space and time complexity of the learning algorithm to $O(N^4 T)$ (Elliot, Aggoun, & Moore 1995).

Other approaches analyze the underlying Markov chain of the HMM and approximate the state transition probability and the output probability making use of frequency counters. This can also be seen as an incremental adaptation of the segmental k-means algorithm (Moore & Ford 1998; Digalakis 1999). Finally, the Kullback-Leibler information measure can also be maximized incrementally to obtain HMMs with improved convergence and reduced memory requirements, compared to models estimated using offline EM algorithms (Krishnamurthy & Moore 1993).

The Incremental Baum-Welch Algorithm (IBW)

We first present the reestimation formulas used in our incremental Baum-Welch algorithm. These formulas are similar to those proposed by Stenger et al. (2001) for continuous models. Since our work focuses on discrete hidden Markov models, we present the reestimation formulas for each of the elements of the **B** matrix. A formulation suitable for incremental learning updates the values of $\overline{a}_{ij}$ and $\overline{b}_j(k)$ in the current time step given the values of those estimators in the previous time step. The initial probability distribution π does not need to be reformulated for each time step, since it corresponds to the expected frequency of being in state i at the specific time $t = 1$.

The estimator $\overline{a}_{ij}$ at the current time step T is given by:

$$\overline{a}_{ij}^{T} = \frac{\overline{a}_{ij}^{T-1}\left(\sum_{t=1}^{T-2}\gamma_t(i)\right) + \xi_{T-1}(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad (3)$$

and the output probability, $\overline{b}_j^T(k)$, is given by:

$$\overline{b}_j^{T}(k) = \frac{\overline{b}_j^{T-1}(k)\left(\sum_{t=1}^{T-1}\gamma_t(j)\right) + \psi(T,j,k)}{\sum_{t=1}^{T}\gamma_t(j)} \quad (4)$$

where $\psi(T,j,k)$ is an auxiliary function defined as:

$$\psi(T,j,k) = \begin{cases} 0 & \text{if } O_T \neq v_k \\ \gamma_T(j) & \text{otherwise} \end{cases} \quad (5)$$

These equations estimate the model parameters for each new observation in a stream. However the β-values (and therefore $\xi_t(i,j)$) cannot be computed incrementally because no observations after the current time are available. The incremental learning problem can be solved by approximating the probability of the partial observation from $t+1$ to the end, given the state i at time t and the model λ, defined as $\beta_t(i) = P(O_{t+1}O_{t+2}...O_T|q_t = i, \lambda)$. As suggested by Stenger et al. (2001), a simple approximation is given by $\beta_t(i) = \beta_{t+1}(i) = \beta_{t+2}(i) = ...\beta_T(i) = 1$ for $1 \leq i \leq N$. The probabilities $\gamma_T(i)$ and $\xi_{T-1}(i,j)$ also need to be recomputed at each time step, using (6) and (7).

$$\gamma_T(i) = \frac{\alpha_T(i)\beta_T(i)}{\sum_{i=1}^{N}\alpha_T(i)\beta_T(i)} = \frac{\alpha_T(i)}{\sum_{i=1}^{N}\alpha_T(i)} \quad (6)$$

$$\xi_{T-1}(i,j) = \frac{\alpha_{T-1}(i)a_{ij}b_j(O_T)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{T-1}(i)a_{ij}b_j(O_T)} \quad (7)$$

An iterative algorithm that make use of these formulas will be known henceforth as the incremental Baum-Welch algorithm, or IBW. This algorithm does not look ahead, nor does it require knowledge about the length of the sequence.

A New Backward Procedure for the Incremental Baum-Welch Algorithm (IBW+)

We now present the principal contribution of this paper, which is an improved method of approximating the β-values in the backward procedure of the incremental Baum-Welch algorithm.

As mentioned before, the incremental learning problem can be solved by approximating the β-values to 1.0, instead of estimating exact values for them. Therefore, a natural improvement for IBW consists of selecting a set of β-values that provide a better approximation to the probability of the partial observation from time $t+1$ to the end given the current state i at time t and the model λ, $\beta_t(i) = P(O_{t+1}O_{t+2}...O_T|q_t = i, \lambda)$ without requiring the entire sequence of T observations to be stored in memory.

Note that $\beta_t(i)$ increases exponentially toward 1.0 as t increases. The backward procedure for a state i computes the β-values in the following order: At time 0 compute $\beta_T(i)$, at time 1 compute $\beta_{T-1}(i)$; ...; at time $T-1$ compute $\beta_1(1)$; and at time T compute $\beta_0(i)$. Since $\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$ and each a_{ij}, $b_j(O_{t+1})$ and $\beta_{t+1}(j)$ term is less than 1.0 (often significantly less), the sequence $\beta_T(i)$,$\beta_{T-1}(i)$,...,$\beta_0(i)$ tends exponentially to zero. Since $\beta_T(i) = 1.0$ (by definition of the backward procedure), $\beta_t(i)$ increases exponentially toward 1.0.

Therefore, it is reasonable to assume that the backward procedure can be approximated by a decay function ω. Specifically, assuming that $\beta_{t+1}(j) = \omega(T-t, j)$, where T is the total length of the sequence and t is the current time step, $\gamma_t(i)$ is given by:

$$\gamma_t(i) = \frac{\alpha_t(i)\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\omega(T-t,j)}{\sum_{i=1}^{N}\alpha_t(i)\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\omega(T-t,j)}$$

Although many decay functions could be proposed to approximate the β-values (including those that look ahead several observations in the sequence), the series $\beta_T(i)$,$\beta_{T-1}(i)$,...,$\beta_0(i)$ tends exponentially to zero for any state, and therefore for sufficiently large sequences any decay function that approximates the β-values should also satisfy $\omega(T-t,j) - \omega(T-j,k) \approx 0$ for $j \neq k$. In other words, $\omega(T-t,j) \approx \omega(T-t,k)$. Should this be the case, the following equality holds

$$\gamma_t(i) = \frac{\alpha_t(i)\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})}{\sum_{i=1}^{N}\alpha_t(i)\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})} \quad (8)$$

This simple assumption helps us to solve the incremental learning problem. Equations (2) and (8) represent the same probability, and therefore, it is clear that:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1}) \quad (9)$$

Since the real $\beta_t(i)$ is based on a exponential decay function computed via the backward procedure, for large T this approximation seems to be appropriate. In any case, it provides a better approximation than $\forall t\ \forall i\ \beta_t(i) = 1.0$. Note that a backward procedure that recursively uses (9) requires a one-step look ahead in the sequence of observations and can be seen as an example of a *fixed-lag smoothing* algorithm (Moore 1973).

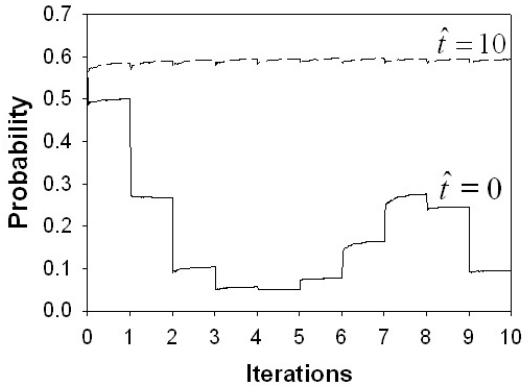

Figure 1: Change over time in estimates of a parameter of the model from source *S(2,2)*, when IBW+ uses smoothing factors of $\hat{t} = 0$ and $\hat{t} = 10$.

The estimation of $\xi_{T-1}(i,j)$ is also improved using (10).

$$\xi_{T-1}(i,j) = \frac{\alpha_{T-1}(i)a_{ij}b_j(O_T)\beta_T(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{T-1}(i)a_{ij}b_j(O_T)\beta_T(j)} \quad (10)$$

An incremental algorithm where $\beta_t(i)$ is approximated using (9) and (10) will be known as the improved incremental Baum-Welch algorithm or IBW+. Note that a scaling procedure is required for large sequences because the computation of the HMM's parameters will exceed the precision range of traditional machines (Rabiner 1989).

Complexity Analysis

Since the estimators at time T are computed with those at time $T-1$, it is not necessary to store estimators for all T observations. Assuming that for most of the practical applications $T \gg M$, the largest structure needed in the algorithm stores the values of $\xi_{T-1}(i,j)$, requiring $O(N^2)$ space. The time complexity of IBW+ is $O(N^2T)$, which is equivalent to the time complexity of BW. Nevertheless, experimental results described in the following sections suggest that IBW+ can converge faster than BW (as other incremental algorithms do), and also converges faster than IBW. However, there is no theoretical guarantee that $P(O|\lambda') \geq P(O|\lambda)$.

This algorithm requires one-step look ahead in the sequence of observations. In practice, an improved model is estimated with an insignificant effect on the space and time complexity of the algorithm.

Additional Improvements

We introduce some additional modifications of the incremental Baum-Welch algorithm that can further improve its convergence behavior.

Smoothing the Parameter Estimates

Preliminary experiments with IBW+ demonstrated that the likelihood of the model can sometimes decrease over time because the BW convergence assumptions do not hold. This decrease is typically small, but in some cases, the likelihood was found to decrease dramatically.

An examination of the experimental results revealed that during the probability estimations of **A**, **B** and π, the learning algorithms tend to forget the contributions of previous estimators at the beginning of each iteration. The drastic changes in the parameters of the HMM were found to be related to the initial values of the sufficient statistics described by $\sum_{t=1}^{T-2}\gamma_t(i)$, $\sum_{t=1}^{T-1}\gamma_t(i)$ and $\sum_{t=1}^{T}\gamma_t(i)$. Such statistics can be seen as weights for the estimation in the previous time step for **A** and **B**. If those weights are very close to zero, the contribution of the previous estimators is minimal.

A simple solution for smoothing the learning of the model over time consists of postponing the update of the parameters until some time $\hat{t} > 0$. After $\hat{t}$, enough statistics should have been collected to support the reestimation of **A**, **B** and π. Empirical evidence suggests that even small values for $\hat{t}$ can smooth the learning process because its main purpose is to avoid zero (or close) to zero statistics independent of the quality of the estimation.

Figure 1 shows how estimates of a single HMM parameter change over time when IBW+ uses smoothing factors $\hat{t} = 0$ and $\hat{t} = 10$. The sequence of observations was generated by a traditional Monte-Carlo simulation from the following 2-state and 2-symbols HMM, denoted *S(2,2)*.

$$\pi = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mathbf{A} = \begin{pmatrix} 0.3 & 0.7 \\ 0.1 & 0.9 \end{pmatrix} \mathbf{B} = \begin{pmatrix} 0.99 & 0.01 \\ 0.2 & 0.8 \end{pmatrix}$$

Learning from Multiple Observation Streams

To this point, the problem of learning incrementally from a single discrete data-stream has been considered. However, in many real-world applications, multiple sequences from a single source can be observed. Assuming that R individual observations are independent, the computation of the parameters of the HMM is modified as follows:

$$\overline{\pi}_i = \frac{1}{R}\sum_{r=1}^{R}\gamma_1^{(r)}(i)$$

$$\overline{a}_{ij}^{T} = \frac{\sum_{r=1}^{R}\overline{a}_{ij}^{T-1}\left(\sum_{t=1}^{T-2}\gamma_t^{(r)}(i)\right) + \xi_{T-1}^{(r)}(i,j)}{\sum_{r=1}^{R}\sum_{t=1}^{T-1}\gamma_t^{(r)}(i)}$$

$$\overline{b}_j^{T}(k) = \frac{\sum_{r=1}^{R}\overline{b}_j^{T-1}(k)\left(\sum_{t=1}^{T-1}\gamma_T^{(r)}(j)\right) + \psi(T,j,k)}{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_T^{(r)}(j)}$$

The estimators above can be applied for both the IBW and IBW+ algorithms, and are similar to the estimators proposed for offline Baum-Welch learning by Li et al. (2000), among others. Note that the space complexity of the learning algorithms increases to $O(N^2R)$, just as the space complexity of a multiple observation learning BW increases to $O(N^2TR)$.

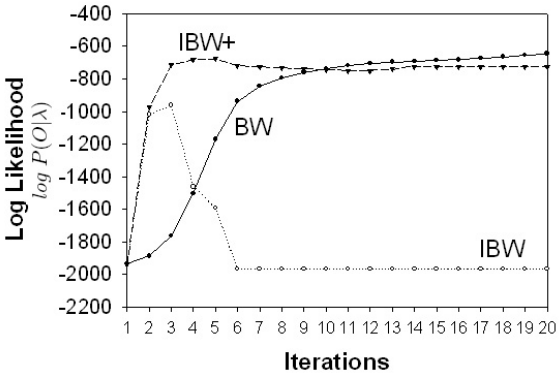

Figure 2: Comparison of the average convergence rates of BW, IBW, and IBW+, in learning an 8-state HMM from 50,000 observations of the source *S(8,0.3)*.

Empirical Results

This section compares the convergence behavior of BW, IBW, and IBW+ on real and synthetic training data. In all of the experiments, IBW used β-values of 1.0, as proposed by Stenger et al. (2001), and IBW+ approximated the β-values using (9) and smoothed the estimators using a smoothing factor of $\hat{t} = 10$. Experiments were performed on a Sun-Blade-100 with 2 gigabytes of RAM running Solaris 5.8.

Figure 2 shows likelihood values averaged over 10 executions of the learning algorithms, trained on sequences of 50,000 observations generated by a Markov chain with 8 states and a conditional relative entropy (CRE) [1] of 0.3. We denote this source by *S(8,0.3)*. Results are presented in a standard log-likelihood graph where the *x-axis* displays the number of iterations (visits to the data stream O) and the *y-axis* shows $log\ P(O|\lambda)$, where λ is the model being estimated with the learning algorithm. One iteration of BW consists of reading the entire sequence of observations, setting the α-values and β-values to zero and updating the parameters of the model via the Baum-Welch algorithm *once*. One iteration of IBW/IBW+ consists of setting the initial statistics to zero and updating the parameters of the model via the incremental estimators for *each* observation. Note that IBW+ converges faster than both BW and IBW. However, as discussed before, the likelihood for the incremental algorithms can decrease over time.

Figure 3 shows the difference between the log likelihoods of the training sequence, $\log(P(O|\lambda_{BW})) - \log(P(O|\lambda_{IBW+}))$, when BW and IBW+ are used to estimate sequences of increasing length from the source *S(2,2)*. Negative values indicate a higher log likelihood for IBW+. The results show that as the length of the sequence of observations increases, IBW+ generates better models than BW.

Figure 4 compares the convergence rates of BW, IBW and IBW+ when trained on five independent observation sequences ($R = 5$). To create the independent sequences, the first ten thousand words from the class *alt.atheism* of the

[1] A CRE of 0 indicates a deterministic source. In contrast, a value of 1 indicates a completely random source.

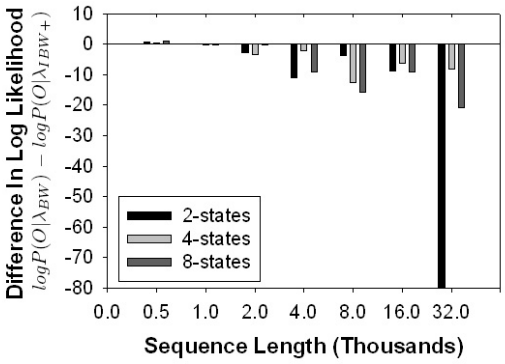

Figure 3: Difference in the log likelihood of models estimated with BW and IBW+ from *S(2,2)* after 10 iterations.

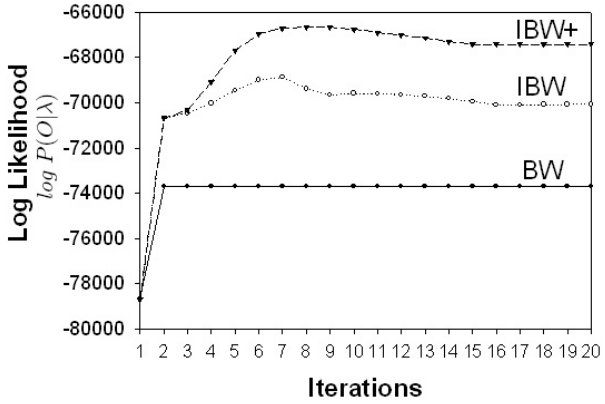

Figure 4: Comparison of the average convergence rates of BW, IBW, and IBW+, in learning an 8-state HMM from five text sequences from *alt.atheism*.

Newsgroups dataset [2] were divided into five sequences, each of length 2,000. Note that both incremental learning algorithms converge faster than BW, and the fastest convergence rate is achieved with IBW+.

Different stopping criteria are used by the IBW/IBW+ and BW algorithms. The reestimation formulas for the incremental learning algorithms are executed until a drop in the likelihood is detected. Since the likelihood never decreases for BW, the estimators in a traditional implementation of the Baum-Welch algorithm are executed up to a maximum number of iterations or to a point where the change in the likelihood is insignificant.

Table 1 compares training times and resulting model quality when the appropriate stopping criteria were used for each algorithm. IBW+ was executed until a drop in the likelihood was detected, and BW was executed for 20 iterations or until $log\ P(O|\lambda_{current}) - log\ P(O|\lambda_{previous}) \leq 10^{-10}$. Experiments were performed using two different data sets. The first is a synthetic data set of 100,000 observations generated by a Markov chain with 2 states and 2 symbols. The second is a sequence of library system calls generated by the Fast

[2] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

Source	Algorithm	$log\mathbf{P}(\mathbf{O}\|\lambda)$	Training Time
S(2,2)	BW	-60,932.7	4.5 mins.
	IBW+	-60,884.2	0.87 mins.
FFT	BW	-60,932.7	49.7 mins.
	IBW+	-89,141.4	8.9 mins.

Table 1: Computer resources and model quality

Fourier Transform (FFT) executing in a Linux cluster (taken from (Florez *et al.* 2005b)). The alphabet for the second task contains 15 symbols and the model contains 32 states. The results show that IBW+ consistently executes in a fraction of the time required for BW, and in some cases can produce models of superior quality.

Conclusion

We have introduced an incremental version of the Baum-Welch algorithm that approximates the β-values used in the backward step based on a one-step look-ahead buffer. Experimental results demonstrate that it converges faster than traditional Baum-Welch and outperforms previous incremental approaches. Moreover, an analysis of the IBW+ reestimation formulas indicates that the longer the sequence of observations, the better the approximation of the β-values. Experimental results confirm that the advantage of the new backward procedure improves with longer sequences. Although we presented this algorithm as an approach to incremental learning of HMMs with discrete observations, the same approach could also be applied to incremental learning of HMMs with continuous observations.

Acknowledgments

We thank the anonymous reviewers for helpful comments. This work was supported by the NSF award No. SCI0430354-04090852 and the Center for Computer Security Research at Mississippi State University.

References

Bengio, Y. 1999. Markovian models for sequential data. *Neural Computing Surveys* 2:129–162.

Binder, J.; Murphy, K.; and Russell, S. 1997. Space-efficient inference in dynamic probabilistic networks. In *International Joint Conference on Artificial Intelligence*.

Digalakis, V. V. 1999. Online adaptation of Hidden Markov Models using incremental estimation algorithms. *IEEE Trans. on Speech and Audio Processing* 7(3):253–261.

Elliot, R. J.; Aggoun, L.; and Moore, J. 1995. *Hidden Markov Models, Estimation and Control*. New York: Springer-Verlag.

Florez, G.; Liu, Z.; Bridges, S.; Skjellum, A.; and Vaughn, R. 2005a. Lightweight monitoring of MPI programs in real-time. *To Appear in Concurrency and Computation: Practice and Experience*.

Florez, G.; Liu, Z.; Bridges, S.; and Vaughn, R. 2005b. Integrating intelligent anomaly detection agents into distributed monitoring systems. *Journal of Network and Computer Applications*. To Appear.

Friedman, N., and Goldszmidt, M. 1999. Learning Bayesian networks with local structure. In Jordan, M., ed., *Learning in graphical models*. MIT Press. 421–459.

Gotoh, Y., and Silverman, H. F. 1996. Incremental ML estimation of HMM parameters for efficient training. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Koenig, S., and Simmons, R. 1996. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA '96)*, 2301 – 2308.

Krishnamurthy, V., and Moore, J. B. 1993. On-line estimation of Hidden Markov Model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on Signal Processing* 41(8):2557–2573.

Lane, T. 2000. *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. Ph.D. Dissertation, Purdue University.

Li, X.; Parizeau, M.; and Plamondon, R. 2000. Training Hidden Markov Models with multiple observations: A combinatorial method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(4):371–377.

MacDonald, I., and Zucchini, W. 1997. *Hidden Markov and Other Models for Discrete-valued Time Series*. Monographs on Statistics and Applied Probability. Chapman and HALL/CRC.

Moore, J., and Ford, J. 1998. Reduced complexity on-line estimation of Hidden Model parameters. In *Proceedings of the 1998 International Conference on Optimization: Techniques and Applications*, 1223–1230.

Moore, J. 1973. Discrete-time fixed-lag smoothing algorithms. *Automatica* 9:163–173.

Neal, R. M., and Hinton, G. E. 1999. A new view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M., ed., *Learning in Graphical Models*. MIT Press. 355–368.

Qiao, Y.; Xin, X.; Bin, Y.; and Ge, S. 2002. Anomaly intrusion detection method based on HMM. *Electronics Letters* 38(13).

Rabiner, L. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77 of *2*, 257–286.

Stenger, B.; Ramesh, V.; Paragios, N.; F.Coetzee; and Buhmann, J. M. 2001. Topology free Hidden Markov Models: Application to background modeling. In *Proceedings of the International Conference on Computer Vision*, 297–301.

Stiller, J., and Radons, G. 1999. Online estimation of Hidden Markov Models. *Signal Processing Letters* 6(8):213–215.

Warrender, C.; Forrest, S.; and Pearlmutter, B. A. 1999. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the IEEE Symposium on Security and Privacy*, 133–145.