Inline Updates for HMMs

Ashutosh Garg
Almaden Research Center
IBM Corporation
ashutosh@us.ibm.com

Manfred K. Warmuth
Dep. of Comp. Sc.
Univ. of Calif. at Santa Cruz
manfred@cse.ucsc.edu

Abstract

Most training algorithms for HMMs assume that the whole batch of observation sequences is given ahead of time. This is particularly the case for the the standard EM algorithm. However, in many applications such as speech, the data is generated by a temporal process.

Singer and Warmuth developed online updates for HMMs that process a single observation sequence in each update. In this paper we take this approach one step further and develop an *inline* update for training HMMs. Now the parameters are updated after processing a single symbol of the current observation sequence. The methodology for deriving the online and the new inline update is quite different from the standard EM motivation.

We show experimentally on speech data that even when all observation sequences are available (batch mode), then the online update converges faster than the batch update, and the inline update converges even faster. The standard batch EM update exhibits the slowest convergence.

1. Introduction

Hidden Markov models (HMMs) are among the most popular probabilistic word model in machine learning. The main applications are speech, vision and bioinformatics. Typically one is given a batch of observation sequences and the goal is to find a parameter setting that maximizes the product of the likelihoods of all sequences. An equivalent goal is to minimize a loss defined as the total negative loglikelihood of all observation sequence. The Estimation Maximization (EM) algorithm (in this context of HMMs called the Baum Welsch algorithm [1]) is traditionally used for this optimization problem. Singer and Warmuth [2] developed some alternate updates that converge faster than EM.

Based on a framework developed by Kivinen and Warmuth [3], their iterative parameter update are motivated by minimizing a sum of two terms. The first is a *divergence function* (to be explained below) between the old and updated HMMs parameters. This term has the function of keeping the updated parameters close to the previous ones. The second term is η times the loss we seek to minimize. Here $\eta > 0$ is a positive learning rate. The solution of this minimization problem gives an expression of the updated parameters $\widetilde{\boldsymbol{\theta}}$ i.t.o. the old parameter setting $\boldsymbol{\theta}$ and the "future" gradient of the loss, i.e. the gradient at $\widetilde{\boldsymbol{\theta}}$. In the update, the future gradients are approximated by the available "old" gradients at $\boldsymbol{\theta}$.

The key is to find a suitable divergence function. HMMs define a joint distribution between the visible observation sequences $\boldsymbol{x}$ and the hidden state sequences $\boldsymbol{s}$. We denote this distribution as $P(\boldsymbol{x}, \boldsymbol{s}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameter vector of our underlying HMMs of a given structure. Singer and Warmuth [2] use the relative entropy between the new and old joint distribution as the divergence function[1], i.e.

$$D(\widetilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \int_{\boldsymbol{x},\boldsymbol{s}} P(\boldsymbol{x}, \boldsymbol{s}|\widetilde{\boldsymbol{\theta}}) \ln \frac{P(\boldsymbol{x}, \boldsymbol{s}|\widetilde{\boldsymbol{\theta}})}{P(\boldsymbol{x}, \boldsymbol{s}|\boldsymbol{\theta})}. \qquad (1)$$

Online and batch *joint entropy* (JE) updates can be derived this way.[2] In the batch update, the total negative loglikelihood over all observations sequences is the loss function, whereas in the case of online, the negative loglikelihood of a *single* observation sequence becomes the loss.

In this paper we develop a third *inline* update (using again the joint entropy as a divergence function) in which we update the parameters after each observation of each observation sequence is processed. For this purpose we decompose the loss of each observation sequence $\boldsymbol{x}$ into a sum $-\ln P(\boldsymbol{x}|\boldsymbol{\theta}) = -\sum_{t=1}^{|\boldsymbol{x}|} \ln P(x_t|x_1 x_2 \ldots x_{t-1}, \boldsymbol{\theta})$ and use $-\ln P(x_t|x_1 x_2 \ldots x_{t-1}, \boldsymbol{\theta})$ as the loss for deriving the update. Again we need to approximate future gradients by old gradients.

We experimentally compare the batch, online, and inline joint entropy updates with EM on speech data. EM (which has buildin learning rate $\eta = 1$) converges slowly. Frequently there are plateaus in the loglikelihood curve and it takes EM many iterations to push through the plateaus. As was observed on limited experiments in

Manfred Warmuth acknowledges the support of NSF grant CCR-9821087.

[1] The same divergence function was used to derive updates for mixtures of Gaussians [4], where the joint distribution is between the observed points and identity of the hidden cluster.

[2] The (batch) EM update can also be derived with the same framework. However, a different relative entropy must be used and a key necessary simplification in the derivation requires $\eta = 1$. So EM has a hard-wired learning rate of 1 which is (as we shall see in the experiments) too low for batch training to too high for online training.

[2], the batch entropic update converges faster than EM. It uses a learning rate larger than one ($\eta \approx 1.1$) to do that. Further improvements are achieved with the online version of the JE update (now $\eta \approx 0.15$, i.e. smaller than one). Finally the inline JE update leads to the fastest convergence (with an even lower learning rate of $\eta \approx 0.015$). The are no online and inline equivalents to EM. However, surprisingly, the fastest converging batch update is the inline JE update which does multiple passes over the entire batch of observation sequences and updates the parameters after each symbol of each observation sequence is processed.

During each iteration, the batch updates do the following for each example observation sequence $\boldsymbol{x}$: Expand the HMM to a trellis of depth $\boldsymbol{x}$; compute the expected usage of each parameter by doing a forward and backward pass over the trellis; average the usages over all examples in the batch. Both the EM and batch joint entropy update rely on the same usage information to update their parameters. The online JE update only needs the usage information for a single example. The inline JE update is, however, quite different. In particular, it employs only a forward pass (Since the end of the word might not be known when the current symbol is processed).

2. Notations

In this paper be limit our discussion to discrete HMMs of a fixed structure. The state transition probabilities, are denoted as $a_{ij} = P(s_{t+1} = j|s_t = i, \boldsymbol{\theta})$, for $0 \leq i \leq N-1$, $1 \leq j \leq N$. (State 0 is a special start state and state N an absorbing final state.) The output probabilities are $b_{jq} = P(b_t = q|s_t = j)$, for $1 \leq j \leq N$, $1 \leq q \leq M$. All these parameters are denoted as $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{B}\}$. An HMM generates a sequence of symbols by starting in state 0 and iteratively does the following until state N is reached: Transit from the current state i to the next state using the probabilities specified by row i of matrix $\boldsymbol{A}$. After arriving at state j, a symbol is output based on the output probabilities of that state (row j of $\boldsymbol{B}$). Let $P(\boldsymbol{x}, \boldsymbol{s}|\boldsymbol{\theta})$ be the probability that HMM $\boldsymbol{\theta}$ generates observation sequence $\boldsymbol{x}$ on path $\boldsymbol{s}$ (which goes from states 0 to N and has $|\boldsymbol{x}|+1$ states): $P(\boldsymbol{x}, \boldsymbol{s}|\boldsymbol{\theta}) = \prod_{t=1}^{|\boldsymbol{x}|} a_{s_{t-1}s_t} b_{s_t, x_t}$. In addition, let $n_i(\boldsymbol{s})$ be the number of times state i occurs in $\boldsymbol{s}$ and let $n_i(\boldsymbol{\theta})$ be the expected usage of state i, i.e. $n_i(\boldsymbol{\theta}) = \sum_{\boldsymbol{s}} n_i(\boldsymbol{s}) P(\boldsymbol{s}|\boldsymbol{\theta})$. While these usages are computable in $O(N^3)$ time using essentially the Floyd-Warshall all-pairs shortest path algorithm, we will sometimes approximate them by related parameters.

3. Joint entropy updates for HMMs

We first rederive the batch and online updates of [2] and then develop the new inline update. Following [3] all updates are motivated using the following tradeoff equation:

$$\widetilde{\boldsymbol{\theta}} := \operatorname{argmin} \left(D(\widetilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) + \eta \operatorname{Loss}(\widetilde{\boldsymbol{\theta}}) \right). \qquad (2)$$

Here $\widetilde{\boldsymbol{\theta}}$ is the updated parameter vector and $\boldsymbol{\theta}$ the old parameter vector. $\operatorname{Loss}(\widetilde{\boldsymbol{\theta}})$ will be some negative loglikelihood depending on whether we are in the batch, online or inline case. Let us postpone the discussion of the loss for a moment. The divergence function measures the discrepancy between the new and the old parameter vectors and η is a non-negative learning rate. Intuitively, the lower η, the more emphasis is given to the divergence term and the updated parameter $\widetilde{\boldsymbol{\theta}}$ will stay close to the old parameter $\boldsymbol{\theta}$. For HMMs, we use the relative entropy between the new and old joint distribution as our divergence (1) and this divergence decomposes as follows [2]:

$$D(\widetilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \sum_{i=0}^{N-1} n_i(\widetilde{\boldsymbol{\theta}}) D(\widetilde{\boldsymbol{a}}_{i*}, \boldsymbol{a}_{i*}) + \sum_{j=1}^{N} n_j(\widetilde{\boldsymbol{\theta}}) D(\widetilde{\boldsymbol{b}}_{j*}, \boldsymbol{b}_{j*})).$$

Here $\boldsymbol{a}_{i*}$ and $\boldsymbol{b}_{j*}$ are rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ of dimension N and M, respectively. The divergence between probability vectors is the standard relative entropy, e.g. $D(\widetilde{\boldsymbol{a}}_{i*}, \boldsymbol{a}_{i*}) = \sum_{j=1}^{N} \widetilde{a}_{ij} \ln \frac{\widetilde{a}_{ij}}{a_{ij}}$.

Plugging this expression of the divergence into (2) and minimizing w.r.t. the constraints that each row of $\widetilde{\boldsymbol{A}}$ and $\widetilde{\boldsymbol{B}}$ must sum to one, we arrive at the following equations for the updated parameters:

$$\widetilde{a}_{ij} = \frac{a_{ij} e^{-\frac{\eta}{n_i(\widetilde{\boldsymbol{\theta}})} \frac{\partial Loss(\widetilde{\boldsymbol{\theta}})}{\partial a_{ij}}}}{Z} \quad \widetilde{b}_{jq} = \frac{b_{jq} e^{-\frac{\eta}{n_j(\widetilde{\boldsymbol{\theta}})} \frac{\partial Loss(\widetilde{\boldsymbol{\theta}})}{\partial b_{jq}}}}{Z'} \qquad (3)$$

Here Z and Z' assure that the rows $\widetilde{\boldsymbol{a}}_{i*}$ and $\widetilde{\boldsymbol{b}}_{j*}$ sum to one.

3.1. Batch and Online Learning

We define $\operatorname{Loss}(\boldsymbol{\theta})$ as $-\frac{1}{|X|} \sum_{\boldsymbol{x} \in X} \ln P(\boldsymbol{x}|\boldsymbol{\theta})$. In the batch case, X is the whole batch of observation sequences. In the online case X consists of the current single observation sequence $\boldsymbol{x}$ and $\operatorname{Loss}(\boldsymbol{\theta}) = -\ln P(\boldsymbol{x}|\boldsymbol{\theta})$. When online algorithm are applied as batch algorithms then they simply cycle over the batch while processing one observation sequence at a time.

The derivatives of the loglikelihoods [3] are

$$\frac{\partial}{\partial a_{ij}} \log P(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{t=1}^{|\boldsymbol{x}|} \frac{P(s_{t-1} = i, s_t = j|\boldsymbol{x}, \boldsymbol{\theta})}{a_{ij}}$$

$$\frac{\partial}{\partial b_{jq}} \log P(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{t=1:x_t=q}^{|\boldsymbol{x}|} \frac{P(s_t = j|\boldsymbol{x}, \boldsymbol{\theta})}{b_{jq}}$$

[3] Note that in speech literature [1], the following notations are common: $\xi_t(i,j) = P(s_{t-1} = i, s_t = j|\boldsymbol{x}, \boldsymbol{\theta})$ and $\gamma_t(j) = P(s_t = j|\boldsymbol{x}, \boldsymbol{\theta})$.

Note that the new $\widetilde{\boldsymbol{\theta}}$ parameters appear on the left and right hand sides of the update equations (3). So we replace all $\widetilde{\boldsymbol{\theta}}$ parameters on the right hand sides by the corresponding $\boldsymbol{\theta}$ parameters. We also further approximate $n_i(\boldsymbol{\theta})$ by $\sum_{\boldsymbol{x}\in X} n_i(\boldsymbol{x},\boldsymbol{\theta})/|X|$, where $n_i(\boldsymbol{x},\boldsymbol{\theta})$= $\sum_{\boldsymbol{s}} n_i(s)P(\boldsymbol{s}|\boldsymbol{x},\boldsymbol{\theta})$, and arrive at the batch JE update [2]:

$$\widetilde{a}_{ij} = \frac{a_{ij}e^{\frac{\eta}{\sum_{\boldsymbol{x}\in X} n_i(\boldsymbol{x},\boldsymbol{\theta})}\sum_{\boldsymbol{x}\in X}\sum_{t=1}^{|\boldsymbol{x}|}\frac{P(s_{t-1}=i,s_t=j|\boldsymbol{x},\boldsymbol{\theta})}{a_{ij}}}}{Z}$$

$$\widetilde{b}_{jq} = \frac{b_{jq}e^{\frac{\eta}{\sum_{\boldsymbol{x}\in X} n_j(\boldsymbol{x},\boldsymbol{\theta})}\sum_{\boldsymbol{x}\in X}\sum_{t=1;x_t=q}^{|\boldsymbol{x}|}\frac{P(s_t=j|\boldsymbol{x},\boldsymbol{\theta})}{b_{jq}}}}{Z'}$$

For the online update[4], X consists of a single observation $\boldsymbol{x}$ (i.e. $|X| = 1$) and thus in that case, the sum $\sum_{\boldsymbol{x}\in X}$ disappears in the above update equations.

3.2. Inline JE Updates

We decompose the online loss $-\ln P(\boldsymbol{x}|\boldsymbol{\theta})$ into a sum $-\sum_{t=0}^{|\boldsymbol{x}|-1}\ln P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta})$ and use $\text{Loss}(\boldsymbol{\theta}) = -\ln P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta})$ as the inline loss for processing symbol x_{t+1}. Let $\boldsymbol{\theta}_t$ denote the parameters right before x_{t+1} is processed. Based on the above loss, the parameters are updated to $\boldsymbol{\theta}_{t+1}$ right after x_{t+1} is observed. We rewrite the conditional probability as:

$$\begin{aligned} & P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t) \\ = & \sum_{k,k'} P(x_{t+1}|s_{t+1}=k,\boldsymbol{\theta}_t)P(s_{t+1}=k|s_t=k',\boldsymbol{\theta}_t) \\ & P(s_t=k'|x_1,...,x_t,\boldsymbol{\theta}_t) \\ = & \sum_{k,k'} a_{k'k,t}b_{x_{t+1}k,t}P(s_t=k'|x_1,...,x_t,\boldsymbol{\theta}_t) \end{aligned} \quad (4)$$

To obtain the update equations for the parameters, the derivatives of the loss function need to be computed.

$$\begin{aligned} & \frac{\partial}{\partial a_{ij,t}}\log P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t) \\ = & \frac{1}{P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t)}\frac{\partial}{\partial a_{ij,t}}P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t) \end{aligned}$$

Using (4), we continue as follows:

$$\begin{aligned} & \frac{\partial}{\partial a_{ij,t}}P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t) \\ = & b_{x_{t+1}j,t}P(s_t=i|x_1,...,x_t,\boldsymbol{\theta}_t) + \\ & \sum_{k,k'} a_{k'k,t}b_{x_{t+1}k,t}\frac{\partial}{\partial a_{ij,t}}P(s_t=k'|x_1,...,x_t,\boldsymbol{\theta}_t) \end{aligned} \quad (5)$$

[4]The corresponding batch EM update is: $\widetilde{a}_{ij} = \sum_{\boldsymbol{x}\in X}\sum_{t=1}^{|\boldsymbol{x}|}P(s_{t-1}=i,s_t=j|\boldsymbol{x},\boldsymbol{\theta})/Z$ and $\widetilde{b}_{jq} = \sum_{\boldsymbol{x}\in X}\sum_{t=1}^{|\boldsymbol{x}|}P(s_{t-1}=i,s_t=j|\boldsymbol{x},\boldsymbol{\theta})/Z'$. It can be derived using (1) with the batch loss and the divergence $D(\widetilde{\boldsymbol{\theta}},\boldsymbol{\theta}) = \frac{1}{|X|}\sum_{\boldsymbol{x}\in X}P(\boldsymbol{s}|\boldsymbol{x},\boldsymbol{\theta})\ln\frac{P(\boldsymbol{s}|\boldsymbol{x},\boldsymbol{\theta})}{P(\boldsymbol{s}|\boldsymbol{x},\widetilde{\boldsymbol{\theta}})}$. When $\eta = 1$, then (1) simplifies to $-\frac{1}{|X|}\sum_{\boldsymbol{x}\in X}P(\boldsymbol{s}|\boldsymbol{x},\boldsymbol{\theta})\ln P(\boldsymbol{s},\boldsymbol{x}|\widetilde{\boldsymbol{\theta}}) + \text{const.}$. Minimizing the this w.r.t. $\widetilde{\boldsymbol{\theta}}$ gives the above EM update.

Similarly, the derivatives w.r.t. the b_{jq} variables become:

$$\begin{aligned} & \frac{\partial}{\partial b_{jq,t}}P(x_{t+1}|x_1,...,x_t,\boldsymbol{\theta}_t) \\ = & I(x_{t+1}=q)\sum_{k'} a_{k'j,t}P(s_t=k'|x_1,...,x_t,\boldsymbol{\theta}_t) + \\ & \sum_{k,k'} a_{k'k,t}b_{x_{t+1}k,t}\frac{\partial}{\partial b_{jq,t}}P(s_t=k'|x_1,...,x_t,\boldsymbol{\theta}_t) \end{aligned} \quad (6)$$

where $I(x_{t+1}=q)$ is 1 if $x_{t+1}=q$ and 0 otherwise. To obtain the inline JE updates, we first update the below variable list $\mathcal{L}_{t-1}$ to $\mathcal{L}_t$ using the approximate update equations given below. We then compute the above $\boldsymbol{\theta}_t$ derivatives (using $\mathcal{L}_t$) and approximate the usages $n_i(\boldsymbol{\theta}_t)$ by $\sum_{\boldsymbol{s}} n_i(\boldsymbol{s})P(\boldsymbol{s}|x_1x_2\ldots x_{t+1},\boldsymbol{\theta}_t)$. Finally, we obtain the updated parameters $\boldsymbol{\theta}_{t+1}$ by plugging $\boldsymbol{\theta}_t$, the $\boldsymbol{\theta}_t$ derivatives, and the approximate[5] usages $n_i(\boldsymbol{\theta}_t)$ into the r.h.s. of JE equations (3).

$$\begin{aligned} \mathcal{L}_{t-1}: \quad & P(x_1,x_2,...,x_{t-1},s_{t-1}=k|\boldsymbol{\theta}_{t-1}), \\ & \frac{\partial}{\partial a_{ij,t-1}}P(x_1,x_2,...x_{t-1},s_{t-1}=k|\boldsymbol{\theta}_{t-1}), \\ & \frac{\partial}{\partial b_{jq,t-1}}P(x_1,x_2,...x_{t-1},s_{t-1}=k|\boldsymbol{\theta}_{t-1}) \end{aligned}$$

Approximate updates[6] of $\mathcal{L}_{t-1}$ to $\mathcal{L}_t$:

$$\begin{aligned} & P(x_1,x_2,...,x_t,s_t=j|\boldsymbol{\theta}_t) \\ = & \sum_i P(x_1,x_2,...,x_{t-1},s_{t-1}=i|\boldsymbol{\theta}_{t-1})a_{ij,t}b_{jx_t,t} \\ & \frac{\partial}{\partial a_{ij,t}}P(x_1,x_2,...,x_t,s_t=k|\boldsymbol{\theta}_t) \\ = & I(j=k)b_{jx_t,t}P(x_1,x_2,...,x_{t-1},s_{t-1}=i|\boldsymbol{\theta}_{t-1}) \\ & +\sum_{k'} a_{k'k,t}b_{kx_t,t}\frac{\partial}{\partial a_{ij,t-1}}P(x_1,x_2,...,x_{t-1},s_{t-1}=k'|\boldsymbol{\theta}_{t-1}) \\ & \frac{\partial}{\partial b_{jq,t}}P(x_1,x_2,...,x_t,s_t=k|\boldsymbol{\theta}_t) \\ = & I(x_t=q,j=k)\sum_i a_{ij,t}P(x_1,x_2,...,x_{t-1},s_{t-1}=i|\boldsymbol{\theta}_{t-1}) \\ & +\sum_{k'} a_{k'k,t}b_{kx_t,t}\frac{\partial}{\partial b_{jq,t-1}}P(x_1,x_2,...,x_{t-1},s_{t-1}=k'|\boldsymbol{\theta}_{t-1}) \end{aligned}$$

4. Experimental Results for Speech

We present two sets of experiments on the TI Digit dataset for which we compare the convergence of the log-likelihood for the updates discussed in this paper. The raw speech data was first represented in terms of 13 cepstral coefficients. We then used the K-means algorithm to discretize the cepstral vectors into 32 bins. This let us

[5]Alternatively, the exact $O(N^3)$ computations for the $n_i(\boldsymbol{\theta}_t)$ may be used.

[6]The update equations are approximate because $\boldsymbol{\theta}_t$ is sometimes replaced by $\boldsymbol{\theta}_{t-1}$

represent each utterance of a spoken digit as a discretized string with alphabet size 32. For both sets of experiments we used a left-to-right HMM with one start, one final and five intermediate states ($7 \times 7/2$ transition and 7×32 output probabilities).

In the first set of experiments we compare the batch EM and JE updates and the online JE update. All the algorithms use the same information computed via a forward and backward pass over each obervation sequence. The online algorithm only processes one observation sequence at a time and one pass over the entire batch of observation sequences corresponds to a single batch update. In general, both the online and the batch updates spend $O(N^2 + NM)$ per symbol, where N is the number of states and M is the size of the output alphabet.

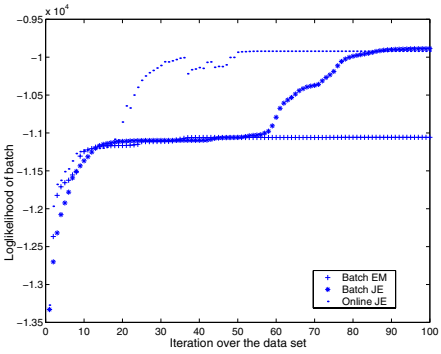

Figure 1: Comparison of the batch EM and JE, and online JE updates.

Fig. 1 gives the loglikelihood plots for the three updates discussed so far. We took 20 sequences for digit *one* and give the loglikelihood as a function of each iteration over the data set. Each point on the x-axis corresponds to one iteration over the whole batch of 20 sequences (i.e. one batch update and 20 online updates). Typically EM gets stuck in plateaus. The batch JE update transitions through the plateaus, but the online update transitions much earlier. We used learning rate $\eta = 1.1$ for the batch JE update and $\eta = 0.15$ for online JE update. Slight changes of these learning rates don't significantly affect the performance of the algorithms.

The second experiment compares the performances of the online and inline JE updates. The inline update costs $O(N^3 + N^2M)$ per symbol, which is a factor of N larger than all other updates discussed in this paper.[7] However, the inline update does not have to wait for the end of the entire observation sequence to process a symbol. If the data is stored in memory, then one might stick with the online JE update. In Fig. 2, each point on the x-axis corresponds to a pass over one observation sequence (one online update and $|\boldsymbol{x}|$ many inline updates). We used learning rate $\eta = 0.15$ for the online and $\eta = 0.015$ for the inline JE update. The inline update is less stable w.r.t. changes of the learning rate and if we account for the additional computational cost of the inline update (factor of N), then the slightly faster convergence of the inline update is diminished.

[7] The sum over k in (5,6) can be preprocessed.

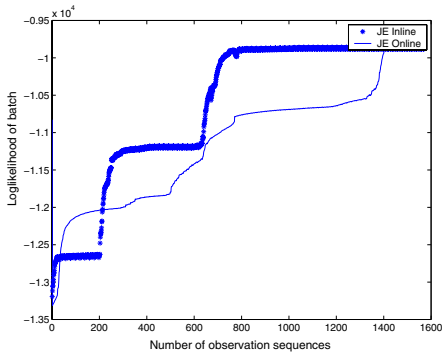

Figure 2: Performance comparison of Inline JE and Online JE algorithms for HMM.

5. Conclusion

For speech, EM is typically run for only a small number of iterations. The advantage of the batch and online JE updates only becomes pronounced when number of passes over the data is high. The inline JE update can converge quite fast, but its performance is likely to depend on the application, since it relies only on a forward pass.

In this paper we developed inline updates for discrete HMMs where the observation probability the nodes multinomials. However, similar inline updates should be derivable for continuous density HMMs, where the multinomials are replaced by mixtures of Gaussians. (For the case of a single Gaussian mixture, the joint entropy updates where already developed in [4].)

It is an open question whether the resulting inline updates for continuous HMMs are stable enough and again beat the corresponding batch and online updates.

6. References

[1] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[2] Y. Singer and M. K. Warmuth, "Training algorithms for hidden markov models using entropy based distance functions," in *Advances in Neural Information Processing Systems 9. (NIPS*96)*, M. Mozer, M. Jordan, and T. Petsche, Eds. MIT Press, London, UK, 1997, pp. 641–647.

[3] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction," *Information and Computation*, vol. 132, no. 1, pp. 1–64, Jan. 1997.

[4] Y. Singer and M. K. Warmuth, "A new parameter estimation method for gaussian mixtures," in *Advances in Neural Information Processing Systems 11 (NIPS*98)*. MIT Press, London, UK, 1998.