Active Learning for Hidden Markov Models: Objective Functions and Algorithms

Brigham Anderson BRIGHAM@CMU.EDU
Andrew Moore AWM@CS.CMU.EDU
Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Hidden Markov Models (HMMs) model sequential data in many fields such as text/speech processing and biosignal analysis. Active learning algorithms learn faster and/or better by closing the data-gathering loop, i.e., they choose the examples most informative with respect to their learning objectives. We introduce a framework and objective functions for active learning in three fundamental HMM problems: model learning, state estimation, and path estimation. In addition, we describe a new set of algorithms for efficiently finding optimal greedy queries using these objective functions. The algorithms are fast, i.e., linear in the number of time steps to select the optimal query and we present empirical results showing that these algorithms can significantly reduce the need for labelled training data.

1. Introduction

In machine learning applications, the quality of learned concepts are often limited by the amount of data available, because in the real world data is finite and often expensive to aquire. One solution to this problem is to have the learner explicitly compute the value of each potential piece of data, and only ask for the most informative data (Cohn et al., 1994). A central question in active learning, then, is how to define value of information. A simple definition is used by *uncertainty sampling*, which prefers queries whose label the learner is unsure of (Lewis & Catlett, 1994; Cohn et al., 1995). This has the drawback of not explicitly accounting for sample variance, so the learner may find itself fascinated by label noise containing no information. A more robust approach is *query by committee* (QBC), which prefers "controversial" points. The algorithm maintains a population of models, and selects queries that engender the most disagreement among these models (Seung et al., 1992). Intuitively, this is appealing because it will try to choose the query that maximally divides the version space of models. Another set of methods concerned with the size of the model space are *entropy-based* objective functions, which directly attempt to find queries that minimize the posterior's entropy (Mackay, 1992; Tong & Koller, 2000). Another approach to active learning is direct *error reduction*, which values queries according to how much they are expected to reduce future classification error (Roy & McCallum, 2001).

Active learning can be differentiated into *pool*-based and *stream*-based problems. The pool-based task occurs when the learner can choose from a preexisting set of examples. The stream-based version only allows the learner to sequentially decide whether to accept or reject a single query. This paper will address pool-based queries.

There are two primary contributions of this paper. First, it is an attempt to clarify the issues in HMM active learning, as there are many heuristics currently being applied. Second, we provide efficient algorithms for each (save one) of the objective functions introduced, and believe the algorithms for cost-based state active learning and path learning to be novel and relevant to other graphical models.

1.1. Hidden Markov Models (HMMs)

An HMM is defined by the parameter θ, which is a tuple of five parameters $\mathcal{S}$, O, A, b, and p_1.

$\mathcal{S}$ is the *state space*, a set of N states $\{1,\ldots,N\}$.
O is the *observation space*, a set of T symbols $\{1,\ldots,M\}$.
A is the $N \times N$ *transition matrix* where element $a_{ij} = P(S_{t+1} = j | S_t = i)$
B are the *output probabilities* in which $b_i(o) = P(o|S = i)$
$p_1(S_1)$ describes a $N \times 1$ *initial state distribution* at time 1.

A sequence of hidden states, $\Pi = \{S_1, S_2, ..., S_T\}$ produces a sequence of observations, $\mathbf{O} = \{O_1, O_2, ..., O_T\}$. From (Rabiner, 1990), the vectors α, β, and γ make themselves

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

useful, as well as the matrices ξ_t.

$$\alpha_t[i] = P(\mathbf{O}_1^t, S_t = i|\theta) \quad (1)$$
$$\beta_t[i] = P(\mathbf{O}_{t+1}^T|S_t = i, \theta) \quad (2)$$
$$\gamma_t[i] = P(S_t = i|\mathbf{O}, \theta) \quad (3)$$
$$\xi_t[i,j] = P(S_t = i, S_{t+1} = j|\mathbf{O}, \theta) \quad (4)$$

where $\mathbf{O}_l^k$ is the subsequence $\{O_l, O_{l+1}, \cdots, O_{k-1}, O_k\}$. The vector γ_t will be referred to as the *belief state* at time t. The belief state incorporates all the observations $\mathbf{O}$ and summarizes the probability of each state at time t. These quantities are obtained in $O(TN^2)$ from the Forward-Backward algorithm. The following will also prove useful in Section 4:

$$F_t[i,j] = P(S_{t+1} = j|S_t = i, \mathbf{O}) = \xi_t[i,j]/\gamma_t[i] \quad (5)$$
$$R_t[i,j] = P(S_{t-1} = j|S_t = i, \mathbf{O}) = \xi_{t-1}[i,j]/\gamma_t[i] \quad (6)$$

Note that there are T different F_t and R_t matrices. They are identical to the transition matrix A, except they are *conditioned on the observations*. A hidden Markov model is equivalent to an inhomogeneous Markov chain using F_t for forward transition probabilities. E.g., $\gamma_{t+1} = F_t\gamma_t$.

2. HMM Active Learning Framework

Suppose that we are learning an HMM to recognize human activity in an office setting. The observations come from various sensors that can measure the user's motion, sound levels, keystrokes, and mouse movement, and the hidden state is the activity that the user is engaged in (meeting, email, word processing, reading, coding, etc.) As the HMM, you are allowed to ask the user a handful of questions once per week such as "in this scene from 4:02pm Wednesday, what were you doing?" We also assume a non-omniscient labeller, so some noise in the labelling process should be modelled. At the end of the week, which time steps are most profitable for querying?

The active learning HMM is equivalent to a standard HMM with one difference, some observations are hidden and queryable. Denote by Q_t a hidden observation at time t.

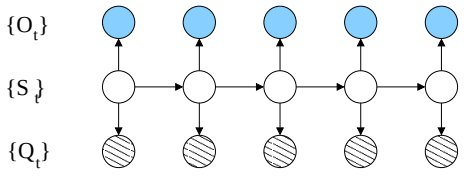

Figure 1. Example HMM active learning task. The learner can choose to observe a query of any time step. The unshaded nodes are the hidden states $\{S_t\}$, the shaded nodes are the visible observations $\{O_t\}$, and the patterned nodes are hidden-but-queryable query observations $\{Q_t\}$.

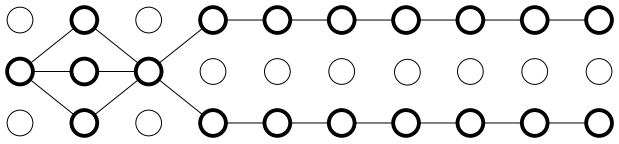

Figure 2. State active learning vs. path active learning. This is an example three-state HMM evolving over 10 time steps from left to right. The connections between states represent the only possible transitions that are consistent with the observations, but the true states are not known. Active *state* learning would prefer any of time steps 4-10, as knowing the state of any one of them will disambiguate the last 7 timesteps. However, only 1 bit of information is learned in that case, i.e., whether the upper or lower path was taken. Active *path* learning would prefer time step 2, as up to $\log 3$ bits can be learned from it.

Figure 1 shows the dependencies among S, O, and Q. The values of all Q_t are hidden until a query allows the learner to see a particular value. In the activity recognition task, for instance, there is one query variable per time step, the value of which is the answer that the user would give to a "what are you doing?" query. A query Q_t is a flexible representation that can specify the result of a labelling, e.g., "an expert classification of the state at time t", or a test of some kind, e.g., "the result of a measurement at time t" while also allowing the tests to give indirect, ambiguous, and/or noisy results. Queries that give the exact state value, as in (Scheffer et al., 2001; Tur et al., 2003), are modelled when $b_i(i) = 1$.

Query costs are optionally specifiable; a cost function $c(Q_t, S_t, t)$ can depend on the type of query, the true underlying state of the system, and the timestep. The framework can thus support many types of cost-sensitive learning. In the activity tracking example, true/false queries may cost less than asking the user to choose from a list, and querying about recent events may be less annoying (cost less) than asking about distant events.

2.1. HMM Tasks

HMMs have two sets of latent variables, the parameters Θ and the hidden states Π, and the loss function for state learning is further determined by whether one is interested in the hidden states on a state-by-state basis or the sequence as a whole (see Figure 2.) The active learning framework can be applied to all three objectives. *State Learning* corresponds to tasks where the goal is to maximize the number of correctly labelled states. Applications of this type commonly occur, for instance, in biological sequence labelling, where the distribution over individual states is more useful than finding the single most likely path. *Path Learning* has the goal to find a single maximum likelihood path of hidden states. These tasks occur when the sequence of states must be considered as a whole, such as in text and speech

recognition. *Model Learning/Discrimination* attempts to minimize confusion about which model generated the data. Note that this is identical to the *classification* task, in which the objective is also to discriminate between models that may have generated an observation sequence. It is also relevant to quantifying disagreement in QBC.

3. Loss Functions

In active learning, the loss function is determined by what one dislikes about one's current belief state. The form of the loss function is $L(p)$, where p are the beliefs, in this case, the joint distribution $p(X,\Theta)$ between the hidden variable X and the model Θ. There are many loss functions to choose from, but two types are common: 1) uncertainty about the hidden variable, and 2) expected error on a task. We will refer to these as entropy-based and cost-based loss functions. In either case, we define the value of information as the expected reduction in loss once Q is known:

$$VOI(Q;p) = L(p) - E_Q[L(p|Q=q)] \tag{7}$$

This is the expected loss reduction if we simply observe Q. However, once a loss function is defined, we can also use $L(p|Q=q)$ to evaluate the gain to be had from *interventions* where Q is deliberately clamped to some value q, as in (Tong & Koller, 2000; Steck & Jaakkola, 2002). However, one would need to ensure that only nodes that are "downstream" of the intervention are measured.

3.1. Information

The first objective minimizes entropy, and the corresponding loss functions can be found in the first column of Table 1. For example, if we wish to select a Q_t to learn the most about some hidden variable X, we would choose the query that maximally reduces the expected Shannon entropy of $p(X)$. An entropy loss function $L(p(X)) = H(p(X))$ implies

$$VOI(Q;p) = H(p(X)) - E_Q[H(p(X|Q=q))] \tag{8}$$
$$= H(X) - H(X|Q) \tag{9}$$
$$= H(Q) - H(Q|X) \tag{10}$$

Recall that the entropy of a distribution p over x is $H(X) = -\sum_x p(x)\log p(x)$, and that the conditional entropy $H(X|Q) = \sum_q p(q)\sum_x p(x|q)\log p(x|q)$. Equations 9 and 10 are equivalent by the symmetry property of mutual information.

This is in fact equivalent to another information-based measure used in active learning, *Kullbeck-Leibler divergence*. This measures change in a distribution, where $\mathbf{KL}(p_1||p_2) = \sum_w p_1(w)\log p_2(w)/p_1(w)$. With **KL**, the objective is to choose Q such that the posterior over one's beliefs is expected to diverge maximally from its current distribution. This is identical to using Shannon entropy, since the *expectation* of ΔH and **KL** are identical (Mackay, 1992). So

Table 1. Loss functions used.

	Information	**Cost**
Model	$H(\Theta)$	$\sum_i\sum_j P(\theta=i)P(\theta=j)c_{ij}$
Path	$H(\Pi)$	$1-\sum_\pi P(\pi)^2$
States	$\sum_t H(S_t)$	$\sum_t\sum_i\sum_j P(S_t=i)P(S_t=j)c_{ij}$

$$VOI(Q;p) = H(X) - H(X|Q) \tag{11}$$
$$= E_Q[\mathbf{KL}(p(X)||p(X|q))] \tag{12}$$

and note that they are both equivalent to the mutual information between X and Q, since by definition $\mathbf{MI}(X;Q) = H(X) - H(X|Q)$. This entropy-reducing objective function has a large body of supporting theory. Query by Committee is a Monte Carlo approximation to entropy minimization, and has been proven to exponentially reduce prediction error in number of queries (Freund et al., 1997).

3.2. Cost

In many instances, however, entropy is inappropriate. The entropy criterion places equal value on information gained about all parts of the version space, but one is often only interested in specific areas. The costs of confusing some states may be far different than for others, for example, the cost associated with confusing a medium-priority and a low-priority email will be different from confusing a high-priority with a low-priority email. These misclassification costs can be specified by a cost matrix C, in which c_{ij} is the cost of mislabeling class i as class j.

In classification, the true misclassification cost is $I[\text{argmax}_x P(x) = x^*]c_{x,x^*}$ where $I[\cdot]$ is the indicator function, x^* is the true label, and c_{ij} is the cost of confusing label i for j. This cost will be the same whether the classifier had 100% confidence or 51% confidence in its answer. Since HMMs produce probabilistic estimates of state, we will use the extra information in a margin-based cost function where the true cost for a particular labelling will be $\sum_x P(x)c_{x^*,x}$. Taking the expectation over x^* gives the Bayes risk

$$L(p) = \sum_i\sum_j P(x=i)P(x=j)c_{ij} \tag{13}$$

In contrast to entropy, this loss function is pairwise decomposable, which will make some computations far more efficient. The second column of Table 2 lists the cost-based loss functions we will use.

In cost-sensitive learning, the queries themselves may have different costs, $cost(Q,S_t,t)$, which can depend on the type

of query, the underlying state at t, and the timestep. Being able to optimally weigh query costs versus misclassification risk is desirable in many real-world settings. The entropy objective is less suited to these applications.

Once the loss function has been calculated, the active learning task becomes finding the optimal Q^* such that $Q^* = \text{argmax}_Q \; VOI(Q) + E_Q[cost(q)]$. In what follows we will omit observation costs for clarity, as they are straightforward to include.

4. State Active Learning

HMMs are often used to probabilistically label individual states, where the goal is to get as many states correct as possible, as opposed to guessing the single correct path. This inference problem typically uses the Forward-Backward algorithm (Rabiner, 1990). An example application is estimating which parts of proteins correspond to certain types of structures (Durbin et al., 2000). We will describe two types of loss functions for the state learning task: information-based and cost-based.

4.1. Information-based

This performance metric occurs when minimizing summed state entropies. As mentioned, the statewise entropy loss function is the sum $L(p) = \sum_{t=1}^{T} H(S_t|\mathbf{O})$. So the loss resulting from a particular query value q at time t is

$$\begin{aligned} L(p|Q_t = q) &= \sum_{k=1}^{T} H(S_k|Q_t = q) \\ &= \sum_{k=1}^{T}\sum_{i}^{N} \gamma_{k|Q_t=q}[i] \log \gamma_{k|Q_t=q}[i] \end{aligned} \tag{14}$$

which can be evaluated using

$$\begin{aligned} \gamma_{t|Q_t=q}[i] &= P(S_t = i|\mathbf{O}, Q_1 = q) \\ &= P(Q_t = q|S_t = i)P(S_t = i)/P(Q_t) \\ &= b_i(q)\gamma_t[i]/\sum_i b_i(q)\gamma_t[i] \end{aligned} \tag{15}$$

Note that (14) and (7) imply

$$VOI(Q_t) = \sum_{t=1}^{T} H(S_t) - H(S_t|Q_t) \tag{16}$$

However, computing expected entropy reduction can be expensive because (14) must be recomputed TM times per round of query selection. The result is cost quadratic in the number of timesteps, $O(T^2N^2M)$.

4.2. Cost-based

The cost-based objective function measures the expected misclassification costs summed over the individual states. As before, this allows for weighting different errors separately. The loss function is the confusion risk

$$L(p) = \sum_{t=1}^{T}\sum_{i}^{N}\sum_{j}^{N} P(S_t = i)P(S_t = j)c_{ij} \tag{17}$$

$$= \sum_{t=1}^{T} \gamma_t' C \gamma_t \tag{18}$$

where the apostrophe indicates transpose. The difficulty occurs when we compute the consequences of observing a query; the information can propagate to all of the other states, changing each of their contributions to the total. Thus, for each possible value of each potential query we must do two things, 1) propagate the changed state beliefs across all time steps, and 2) recalculate Equation 18. As described this would be an $O(T^2N^2M)$ operation.

Suppose, for example, we only evaluate one possible outcome of seeing one query, Q_1, the query at $t = 1$. If we observe that $Q_1 = q$, the total loss is defined to be

$$L(p|Q_1 = q) = \sum_{t=1}^{T} \gamma'_{t|Q_1=q} C \gamma_{t|Q_1=q} \tag{19}$$

Naively, we would need to recalculate all γ for each timestep before recalculating (18). Recalling Equation 5, we can write (19) as

$$\begin{aligned} L(p|Q_1 = q) = {} & \gamma'_{1|Q_1=q} C \gamma_{1|Q_1=q} \\ & + (F_1\gamma_{1|Q_1=q})' C (F_1\gamma_{1|Q_1=q}) \\ & + (F_2F_1\gamma_{1|Q_1=q})' C (F_2F_1\gamma_{1|Q_1=q}) \\ & + \cdots \\ & + (F_T\cdots F_1\gamma_{1|Q_1=q})' C (F_T\cdots F_1\gamma_{1|Q_1=q}) \end{aligned} \tag{20}$$

and gather all the matrix terms into a single matrix M_1 such that the total loss *in terms of* γ_1 is

$$L(p|Q_1 = q) = \gamma'_{1|Q_1=q} M_1 \gamma_{1|Q_1=q} \tag{21}$$

thus we can use M_1 to determine the total effect of any change in beliefs at $t = 1$ by a single matrix multiplication, reducing complexity by a factor of T. We want to be able to do this for all time steps, so we need T matrices M_t

$$M_t[i,j] = \sum_{k}^{T}\sum_{u}^{N}\sum_{v}^{N} P(S_k = u|S_t = i)P(S_k = v|S_t = j)c_{uv} \tag{22}$$

to clarify a bit further, define

$$\mathbf{F}_{k\to l} = F_l F_{l-1}\cdots F_{k+2}F_{k+1}F_k \qquad \text{for } l > k \tag{23}$$

$$\mathbf{R}_{k\to l} = R_l R_{l-1}\cdots R_{k+2}R_{k+1}R_k \qquad \text{for } l < k \tag{24}$$

then each M_t is

$$M_t = \sum_{k=1}^{t-1} \mathbf{R}'_{t\to k} C \mathbf{R}_{t\to k} + C + \sum_{l=t+1}^{T} \mathbf{F}'_{t\to l} C \mathbf{F}_{t\to l} \quad (25)$$

These useful matrices can be calculated efficiently using dynamic programming. We can calculate *all* matrices M_t at once in $O(TN^2)$. Compute the following recursively

$$M_T^f = \mathbf{0} \quad (26)$$

$$M_1^b = C \quad (27)$$

$$M_t^f = F_t'(M_{t+1}^f + C)F_t \qquad 2 \le t \le T \quad (28)$$

$$M_t^b = R_t'(M_{t-1}^b)R_t + C \qquad T-1 \ge t \ge 1 \quad (29)$$

$$M_t = M_t^f + M_t^b \qquad 1 \le t \le T \quad (30)$$

Now one can compute expected loss for each query using

$$L(p|Q_t = q) = \gamma'_{t|Q_t=q} M_t \gamma_{t|Q_t=q} \quad (31)$$

and VOI can now be computed directly from (7). Note that the cost of the recursion (26)-(30) is $O(TN^2)$, and the cost of computing expected loss for all queries from (31) is $O(TN^2M)$.

5. Path Active Learning

Another common HMM task is to find the most likely *path* of hidden states given some observations. Sometimes called decoding, this differs from the state estimation task in that one is trying to learn about the distribution of the whole sequence, $P(\Pi)$, instead of trying to maximize the number of individually correct states. It may be, for instance, that the individually most likely states together form an impossible path. Examples where decoding is used include text/speech processing, deducing human activities, and extracting likely trajectories from images. The Viterbi algorithm is typically used in these inferences.

5.1. Information-based

The entropy-reduction objective function for this task is $L(p) = H(\Pi|\mathbf{O})$. We can directly maximize our value of information via (9), i.e.,

$$VOI(Q_t) = H(\Pi) - H(\Pi|Q_t) \quad (32)$$

By definition, $H(\Pi) = \sum_\pi P(\pi)\log P(\pi)$, which is a summation over the entire path space. We can avoid a direct attack on the sum by noting the symmetry property of mutual information, which enables us to write VOI as

$$VOI(Q_t) = H(Q_t) - H(Q_t|\Pi) \quad (33)$$

The entropy $H(Q_t)$ can be computed directly from $P(Q_t)$, which can be obtained from γ_t and $b(Q_t)$. However, the conditional entropy $H(Q|\Pi)$ appears just as difficult as (32). Except we can now exploit the conditional independence between Q_t and Π given S_t, so

$$VOI(Q_t) = H(Q_t) - H(Q_t|S_t) \quad (34)$$

$$= H(S_t) - H(S_t|Q_t) \quad (35)$$

which is now easy to compute, knowing

$$H(Q_t|S_t) = \sum_i^N P(S_t = i) H(Q_t|S_t = i) \quad (36)$$

$$= \sum_i^N \gamma_t[i] \sum_k^M b_i[k] \log b_i[k] \quad (37)$$

So the cost for path active learning with entropy is just $O(TNM)$. Note that (34) is illuminating on at least two points: 1) the first term is a selection criterion for uncertainty sampling[1], so the nonoptimal behaviour of uncertainty sampling can be traced to ignoring the conditional entropy, and 2) in the special case where queries correspond to a noiseless state label, as in (Scheffer et al., 2001; Tur et al., 2003), the second term in (34) is always zero, so state-entropy uncertainty sampling is the optimal entropy-based strategy *for path active learning*.

An advantage of entropy in general is that a closed form expression for many continuous distributions exists. Gaussians are one example, so active learning using entropy can be exact for HMMs with Gaussian outputs.

5.2. Cost-based

The goal of active path estimation here is to find the query Q_t that minimizes the expected cost among paths. Because enumerating misclassification costs for every possible pair of paths is not obviously practical, we will use equal-cost loss here. Note, however, that other weighting schemes are possible, such as weighting by state or by time step. The loss function is thus

$$L(p) = \sum_i^{|\Pi|} \sum_j^{|\Pi|} P(\pi_i) P(\pi_j) I[i \ne j]$$

$$= 1 - \sum_{\pi\in\Pi} P(\pi)^2 \quad (38)$$

where the indicator function $I[\cdot]$ takes the place of the cost matrix. The summation in Equation 38 is over the entire space of paths, which has $|S|^T$ members. This seems daunting, but dynamic programming again offers a solution. In fact, the sum can be calculated in $O(TN^2)$ for all Q_t. Define T vectors μ_t such that

$$\mu_t[i] = \sum_{\pi\in\Pi} P(\pi|S_t = i)^2 \quad (39)$$

[1] There are many measures of uncertainty aside from entropy: variance, margin, and confidence intervals have all been used.

We can compute all μ_t by the following recursion:

$$\mu_1^f = \mathbf{1} \tag{40}$$
$$\mu_T^b = \mathbf{1} \tag{41}$$
$$\mu_t^f = (R_t \odot R_t)\mu_{t-1}^f \qquad 2 \le t \le T \tag{42}$$
$$\mu_t^b = (F_t \odot F_t)\mu_{t+1}^b \qquad T-1 \ge t \ge 1 \tag{43}$$
$$\mu_t = \mu_t^f \odot \mu_t^b \qquad 1 \le t \le T \tag{44}$$

where $\odot$ is the Hadamard operator which indicates elementwise multiplication, e.g., the ij-th element of $A \odot B$ is $a_{ij}b_{ij}$. The loss resulting from any possible query observation is simply

$$L(p|Q_t = q) = 1 - (\gamma_{t|q} \odot \gamma_{t|q})'\mu_t \tag{45}$$

Now we can calculate $VOI(Q_t)$ for any Q_t by (7). The cost of calculating VOI for all queries is linear in T, i.e., $O(TN^2 + TNM)$.

6. Model Active Learning/Classification

Model learning is a central task of HMM inference. The loss function is meant to allow the learner to minimize confusion about which model generated the data. However, this is identical to the *classification* task, in which the objective is also to discriminate between models. Thus the following loss functions are appropriate for both model learning and classification. In QBC, the space Θ is the committee for that iteration. In classification, Θ is the space of competing models under consideration. Note that in both the following algorithms, the competing models do not have to have the same number of states, so active model selection is possible.

6.1. Information-based

We wish to select the query Q_t to minimize model uncertainty, so the loss function will be $L(p) = H(p(\Theta|\mathbf{O}))$. As in Section 5, we can directly maximize our value of information by maximizing mutual information, but now we have two equivalent ways to do so:

$$VOI(Q_t) = H(\Theta) - H(\Theta|Q_t) \tag{46}$$
$$= H(Q_t) - H(Q_t|\Theta) \tag{47}$$

for the first, (46), the first term is independent of what query is selected and can be ignored. The second term is $H(\Theta|Q_t) = \sum_q P(q)\sum_\theta P(\theta|q)\log P(\theta|q)$, where $P(\theta|q)$ can be obtained from Bayes rule. For the second equation,

$$H(Q_t|\Theta) = \sum_m P(\theta = m)H(Q_t|\theta = m) \tag{48}$$

and we can directly calculate

$$P(Q_t = q|\theta = m) = \sum_i b_i(q)\gamma_t^{(m)}[i] \tag{49}$$

$$P(Q_t = q) = \sum_m P(\theta = m)\sum_i b_i(q)\gamma_t^{(m)}[i] \tag{50}$$

$P(\Theta)$ is our prior over models and $\gamma_t^{(m)}[i] = P(S_t = i|\theta = m)$.

The value of information, $VOI(Q)$, for model learning can be directly computed from either (46) or (47). If a closed form expression exists for the entropy of $p(\Theta)$ or $p(Q_t)$, then use (46) or (47), respectively. However, one of either Θ or Q_t must be discrete, whether intrinsically or by sampling. Posterior discributions over HMM model parameters are derived in (MacKay, 1997; Rezek & Roberts, 2002). The cost of one round of selection is $O(TN^2M|\Theta|)$.

6.2. Cost-based

If the space of models has different costs associated with different types of model-selection errors, then a cost-based criterion may be more appropriate than entropy. Minimizing model misclassification error will use the loss function $L(p) = \sum_i \sum_j P(\theta_i)P(\theta_j)c_{ij}$ where C is a cost matrix in which c_{ij} is the cost of guessing model i when the true model is j. To calculate L, define T matrices N_k where

$$N_k[i,j] = \sum_u \sum_v P(\theta = u|S_k = i)P(\theta = v|S_k = j)c_{uv} \tag{51}$$

noting that

$$P(\theta = m|S_t = i) = \frac{\gamma_t^{(m)}[i]P(\theta = j)}{\sum_k \gamma_t^{(k)}[i]P(\theta = k)} \tag{52}$$

Now the loss function can be easily calculated by

$$L(p|Q_t = q) = \gamma'_{t|Q_t=q} N_t \gamma_{t|Q_t=q} \tag{53}$$

Note that we can change Equation 51 to be conditioned on the value of Q_t instead of S_t, which enables active learning of models with different numbers of states. The cost for evaluating all queries is linear in the number of time steps: $O(TN^2M|\Theta|^2)$. The cost is quadratic in the number of models only if we want to specify every possible model confusion, however, if only relative importance of the models needs to be included, the cost is linear in $|\Theta|$ as we can use the loss function $L(p) = 1 - \sum_i P(\theta = i)^2 w_i$, where w_i is the weight of model i.

7. Experiments

In this section we empirically investigate the algorithm's performance in simulation and on an activity recognition task. The simulation experiments were conducted by generating random HMMs with 5 states and a single output

having two possible symbols. The query variables returned a noisy state estimate, i.e., $Q_t = s_t$ 50% of the time and a random state otherwise. Each HMM was then used to generate an observation sequence of 100 time steps. The performance on each task was compared with random query selection and uncertainty selection, which selected the query attached to the state with the greatest entropy. Note that the entropy criterion could be made to perform arbitrarily worse than the cost criterion by an asymmetric cost matrix, however, in these experiments the cost is 0/1.

State Learning: The learner was allowed to select and observe a query, after which it reestimated the state probabilities. Performance was measured as classification error over states, $1 - \sum_t P(S_t = s_t^* | \mathbf{O}, \theta^*)$. Due to computational costs, the entropy-based version of state active learning was omitted. "0/1 Cost" refers to the cost-based VOI function with an equal-cost cost matrix. The results are in Figure 3. Cost-based VOI did uniformly better than random and uncertainty sampling. *Path Learning:* Performance was measured as the negative log probability assigned to the single true underlying path, $-\log P(\Pi^* | \mathbf{O}, \theta^*)$. Figure 4 has the results. For this example, the entropy-based VOI objective was best, closely followed by cost-based VOI. *Model Learning:* For each run, a random HMM was used to generate a training sequence and a test sequence, each of 100 observations. After each query is selected and observed from the training sequence, a new maximum likelihood model, $\hat{\theta}$, is learned via Baum-Welch. The performance metric is the classification error in the test sequence, using the same metric as in Figure 3. Since the objective function for model learning requires a population of models, in this experiment a population of size 3 was generated via parametric bootstrap from $\hat{\theta}$ from the previous iteration. Results are in Figure 5. In this small simulation, cost-based and entropy-based VOI performed equally well, and both were superior to random sampling.

The algorithm was also tested on an activity recognition task. The time series was a sequence of 20,000 keystrokes from a user, where the observations were the key category (alphanumeric, punctuation, symbol, space, enter, control/alt, backspace, or arrow key), the duration of the keypress (msec), and the transition time to the next key (msec). The underlying state was the type of application the user was typing in (email, coding, writing paper, shell, or other.) All observations and states had been automatically recorded and a model had been learned before the experiment. The states were then hidden from the learner and the task was to classify as many states correctly as possible. The algorithm was allowed to request queries for any time t, which in this case returned the application that was being used (the state). Random sampling, uncertainty sampling, and VOI-cost sampling were run on the data. VOI-entropy was not applied as the computation cost was too large. The

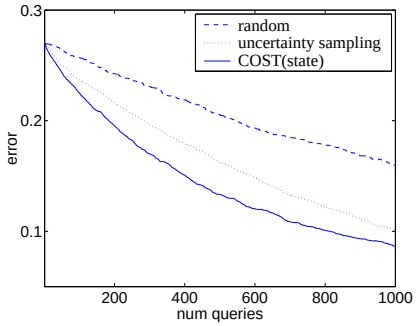

Figure 6. state active learning for activity tracking. First 1000 queries.

results are shown in Figure 6. VOI-cost was the best performer in the first 1000 (5%) of the queries, after which uncertainty sampling caught up.

8. Related Work

The application of active learning to general graphical models has been addressed in (Tong & Koller, 2000; Tong & Koller, 2001; Steck & Jaakkola, 2002) in which constructive queries are used where variables are clamped to value assignments. The application of active learning specifically to Hidden Markov Models has seen previous attention from the speech and text processing fields. These approaches have been implementations of both uncertainty sampling (Scheffer et al., 2001; Tur et al., 2003) and QBC (Tur et al., 2003) methods applied to the model learning task. Optimal *nonmyopic* sets of noiseless queries are described by (Krause & Guestrin, 2005) at a cost cubic in the number of time steps. The stream-based state active learning problem has been addressed with POMDPs by (Krishnamurthy, 2002).

9. Discussion & Future Work

The *VOI* computations are fast enough that incorporating the information learned from the queries into the HMM's beliefs becomes the primary bottleneck, especially in model learning. Future work will address the design of new learning algorithms that can quickly incorporate new queries, such as in online learning (Minka, 2001).

Computing *VOI* can be made even faster by only updating the variables that are affected by information learned from the last query. This is possible since changes beliefs exponentially decay in its effects as the distance along the chain increases (Boyen & Koller, 1998). Thus, after the computation for the first round of query selection, computation is thereafter *constant* w.r.t. the length of the sequence, as long as an arbitrarily small amount of error is permitted in *VOI*.

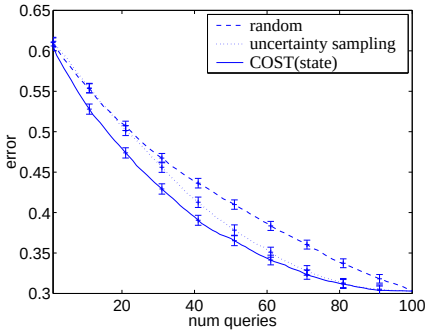

Figure 3. State active learning, simulation data

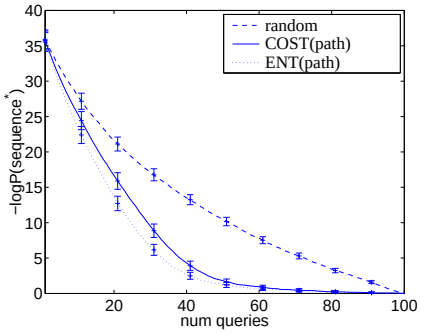

Figure 4. Path active learning, simulation data

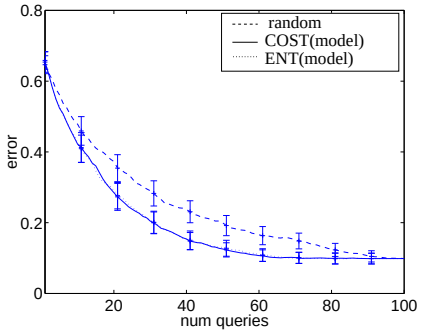

Figure 5. Model active learning, simulation data

10. Conclusions

This work describes the problem of active data selection for the HMM tasks of model learning, state learning, and path learning. Two families of myopically optimal objective functions have been described for three inference tasks as well as fast algorithms for evaluating them. Empirical studies have demonstrated the improved performance of algorithms using these active learning methods.

Acknowledgments

We thank Sajid Siddiqi and the reviewers for their helpful comments. This work partially funded by DARPA grant #NBCHD030010 and NSF ITR grant #CCF-0121671.

References

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 33–42).

Cohn, D. A., Atlas, L., & Ladner, R. E. (1994). Improving generalization with active learning. *Machine Learning, 15*, 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1995). Active learning with statistical models. *Advances in Neural Information Processing Systems* (pp. 705–712). The MIT Press.

Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. (2000). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge Univ. Press. Durbin.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning, 28*, 133–168.

Krause, A., & Guestrin, C. (2005). *Optimal nonmyopic value of information in graphical models* (Technical Report). Carnegie Mellon University.

Krishnamurthy, V. (2002). Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing, 50*, 1382–1397.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of ICML-94, 11th International Conference on Machine Learning* (pp. 148–156). New Brunswick, US: Morgan Kaufmann Publishers, San Francisco, US.

Mackay, D. (1992). Information-Based Objective Functions for Active Data Selection. *Neural Computation, 4*, 589–603.

MacKay, D. (1997). *Ensemble learning for hidden markov models* (Technical Report). University of Cambridge.

Minka, T. (2001). *A family of algorithms for approximate bayesian inference*. Doctoral dissertation, MIT.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected apllications in speech recognition. In A. Waibel and K.-F. Lee (Eds.), *Readings in speech recognition*, 267–296. San Mateo, CA: Kaufmann.

Rezek, I., & Roberts, S. J. (2002). Ensemble hidden markov models for biosignal analysis.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proc. 18th International Conf. on Machine Learning* (pp. 441–448). Morgan Kaufmann, San Francisco, CA.

Scheffer, T., Decomain, C., & Wrobel, S. (2001). Active hidden Markov models for information extraction. *Lecture Notes in Computer Science, 2189*, 309+.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Computational Learning Theory* (pp. 287–294).

Steck, H., & Jaakkola, T. (2002). Unsupervised active learning in large domains. *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)* (pp. 469–476). San Francisco, CA: Morgan Kaufmann Publishers.

Tong, S., & Koller, D. (2000). Active learning for parameter estimation in bayesian networks. *NIPS* (pp. 647–653).

Tong, S., & Koller, D. (2001). Active learning for structure in bayesian networks. *IJCAI* (pp. 863–869).

Tur, G., Schapire, R., & Hakkani-Tur, D. (2003). Active learning for spoken language understanding.