Monte Carlo Hidden Markov Models

Sebastian Thrun and John Langford

December 1998

CMU-CS-98-179

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a learning algorithm for hidden Markov models with continuous state and observation spaces. All necessary probability density functions are approximated using samples, along with density trees generated from such samples. A Monte Carlo version of Baum-Welch (EM) is employed to learn models from data, just as in regular HMM learning. Regularization during learning is obtained using an exponential shrinking technique. The shrinkage factor, which determines the effective capacity of the learning algorithm, is annealed down over multiple iterations of Baum-Welch, and early stopping is applied to select the right model. We prove that under mild assumptions, Monte Carlo Hidden Markov Models converge to a local maximum in likelihood space, just like conventional HMMs. In addition, we provide empirical results obtained in a gesture recognition domain, which illustrate the appropriateness of the approach in practice.

This research is sponsored in part by DARPA via AFMSC (contract number F04701-97-C-0022), TACOM (contract number DAAE07-98-C-L032), and Rome Labs (contract number F30602-98-2-0137). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of DARPA, AFMSC, TACOM, Rome Labs, or the United States Government.

Keywords: annealing, any-time algorithms, Baum-Welch, density trees, early stopping, EM, hidden Markov models, machine learning, maximum likelihood estimation, Monte Carlo methods, temporal signal processing

1 Introduction

Over the last decade or so, hidden Markov models have enjoyed an enormous practical success in a large range of temporal signal processing domains. Hidden Markov models are often the method of choice in areas such as speech recognition [28, 27, 42], natural language processing [5], robotics [34, 23, 48], biological sequence analysis [17, 26, 40], and time series analysis [16, 55]. They are well-suited for modeling, filtering, classification and prediction of time sequences in a range of partially observable, stochastic environments.

With few exceptions, existing HMM algorithms assume that both the state space of the environment and its observation space are discrete. Some researchers have developed algorithms that support more compact feature-based state representations [15, 46] which are nevertheless discrete; others have successfully proposed HMM models that can cope with real-valued observation spaces [29, 19, 48]. Kalman filters [21, 56] can be thought of as HMMs with continuous state and action spaces, where both the state transition and the observation densities are linear-Gaussian functions. Kalman filters assume that the uncertainty in the state estimation is always *normally* distributed (and hence unimodal), which is too restrictive for many practical application domains (see e.g., [4, 18]).

In contrast, most "natural" state spaces and observation spaces are continuous. For example, the state space of the vocal tract of human beings, which plays a primary role in the generation of speech, is continuous; yet HMMs trained to model the speech-generating process are typically discrete. Robots, to name a second example, always operate in continuous spaces; hence their state spaces are usually best modeled by continuous state spaces. Many popular sensors (cameras, microphones, range finders) generate real-valued measurements, which are better modeled using continuous observation spaces. In practice, however, real-valued observation spaces are usually truncated into discrete ones to accommodate the limitations of conventional HMMs. A popular approach along these lines is to learn a *code-book* (vector quantizer), which clusters real-valued observations into finitely many bins, and thus maps real-valued sensor measurements into a discrete space of manageable size [54]. The discreteness of HMMs is in stark contrast to the continuous nature of many state and observation spaces.

Existing HMM algorithms possess a second deficiency, which is frequently addressed in the AI literature, but rarely in the literature on HMMs: they do not provide mechanisms for adapting their computational requirements to the available resources. This is unproblematic in domains where computation can be carried out off-line. However, trained HMMs are frequently employed in time-critical domains, where meeting deadlines is essential. *Any-time* algorithms [9, 58] address this issue. Any-time algorithms can generate an answer at any time; however, the quality of the solution increases with the time spent computing it. An any-time version of HMMs would enable them to adapt their computational needs to what is available, thus providing maximum flexibility and accuracy in time-critical domains. Marrying HMMs with any-time computation is therefore a desirable goal.

This paper presents *Monte Carlo Hidden Markov Models (MCHMMs)*. MCHMMs employs continuous state and observation spaces, and once trained, they can be used in an any-time fashion. Our approach employs Monte Carlo methods for approximating a large, non-parametric class of density functions. To combine multiple densities (e.g., with Bayes rule), it transforms sample sets into density trees. Since continuous state spaces are sufficiently rich to overfit any data set, our approach uses shrinkage as mechanism for regularization. The shrinkage factor, which determines the effective capacity of the HMM, is annealed down over multiple iterations of EM, and early stopping is applied to choose the right model. We prove the convergence of MCHMMs to a local maximum in likelihood space. This theoretical results justifies the use of MCHMM in a wide array of applications. In addition, empirical results are provided obtained for an artificial domain and a gesture recognition domain, which illustrates the robustness of the approach in practice.

The remainder of this paper is organized as follows. Section 2 establishes the basic terminology for generalizing HMMs to continuous state and observation spaces. We will then, in Section Section 3, discuss commonly used sampling schemes and provide an algorithm for generating piece-wise constant density trees from these samples. A key result in this section is a proof of asymptotic consistency of both sample-based and tree-based representations. Section 4 describes our approach for complexity control (regularization) using shrinkage, followed by the statement of the MCHMM algorithm in Section 5. Section 6 proves the MCHMM convergence theorem, which states that under mild assumptions MCHMMs converge with high probability. Empirical results are described in Section 7, which specifically investigates MCHMM in the finite sample case. The empirical results show that MCHMMs work well even in the non-asymptotic case. Section 7 also provides an experiment that characterizes the relation between sample set size (computation) and accuracy. Finally, related work is discussed in Section 8 and the paper is summarized in Section 9.

2 Generalized Hidden Markov Models

This section introduces *generalized hidden Markov models* (in short: *GHMM*). GHMMs generalize conventional hidden Markov models (HMMs) in that all spaces, state and observation, are continuous. Our description closely follows that of Rabiner [43], with densities replacing finite probability distributions throughout. Throughout this paper, we assume all event spaces and random variables (e.g., state, observations) are measurable. We also assume that unless otherwise specified, all probability distributions are continuous and possess continuous density functions. Further below, when introducing density trees, we will also assume that densities are non-zero over a compact, bounded region, and that they obey a Lipschitz condition.

A GHMM is a partially observable, time-invariant Markov chain with continuous state and observation spaces and discrete time. Let x denote a *state variable* (a measurable random variable), defined over some continuous space (e.g., $\Re^k$ for some k). At each time $t \geq 1$, the HMM's state is denoted x_t. Initially, at time $t = 1$, the state of the HMM is selected randomly according to the density π. State transitions are governed by a conditional probability density, denoted $\mu(x' \mid x)$ and called *state transition density*. Densities are measurable functions over the set of Borel sets, hence the Riemann integral

$$\int_{x_0}^{x_1} \mu(x' \mid x)\, dx \tag{1}$$

measures, for $x_0 < x_1$, the probability $Pr(x_0 \leq x' < x_1 \mid x)$ that the state succeeding x lies in $[x_0, x_1)$.

In HMMs (and thus in GHMMs), state cannot be observed. Instead, only a probabilistic projection of the state is observable. Let b_t denote a measurable random variable that models the observation at time t. Observations are generated according to a probability density conditioned on the state of the HMM (called the *observation density*), denoted $\nu(b \mid x)$. If $b_0 < b_1$,

$$\int_{b_0}^{b_1} \nu(b \mid x)\, db \tag{2}$$

measures the probability that the observation b is in $[b_0, b_1)$, given that the state of the HMM is x. Thus, a generalized HMM is uniquely defined through three densities:

$$\lambda \;=\; \{\pi, \mu, \nu\}. \tag{3}$$

Putting computational limitations aside for the moment, knowledge of λ is sufficient to tackle a variety of interesting practical problems:

- Computing distributions over states and observations at arbitrary points in time,
- Generating representative example trajectories in state and observation space,
- Determining the likelihood of example trajectories under an HMM, and
- Classifying data sequences generated by mixtures of labeled HMMs.

Algorithms for these problems are described in detail in [43]; they are easily transferred from the finite to the continuous case.

In practice, the densities λ are often unknown and have to be estimated from data. The data, denoted d, is a sequence of observations[1], denoted

$$d \;=\; \{O_1, O_2, \ldots, O_T\}. \tag{4}$$

Here O_t denotes the observation at time t. The total number of observations in d is T.

The well-known Baum-Welch algorithm [2, 33, 43] provides a computationally efficient and elegant approach for learning π, μ, and ν. Baum-Welch begins with an initial model, denoted $\lambda^{(0)}$. It iterates two steps, an E-step and an M-step (see also [12]). In the n-th E-step, distributions for the various state variables x_t are computed under a fixed model $\lambda^{(n)}$ (with $n \geq 0$). The n-th M-step uses these distributions to derive a new, improved model $\lambda^{(n+1)}$. As shown for example in [33, 37], both steps increase the data likelihood $Pr(d \mid \lambda)$, or leave it unchanged if, and only if, a local maximum in the likelihood function has been reached.

In the E-step, distributions are computed for the state variables x_t conditioned on a fixed model λ and the data d. Recall that in the discrete case,

$$\alpha_t^{(n)}(x) \;=\; Pr(x_t = x \mid O_1, \ldots, O_t, \lambda^{(n)}) \tag{5}$$
$$\beta_t^{(n)}(x) \;=\; Pr(O_{t+1}, \ldots, O_T \mid x_t = x, \lambda^{(n)}) \tag{6}$$
$$\gamma_t^{(n)}(x) \;=\; Pr(x_t = x \mid d, \lambda^{(n)}) \tag{7}$$
$$\xi_t^{(n)}(x, x') \;=\; Pr(x_t = x, x_{t+1} = x' \mid d, \lambda^{(n)}) \tag{8}$$

[1]For simplicity of the presentation, we only present the case in which the data consist of a single sequence. The extension to multiple sequences is straightforward but requires additional notation.

The continuous case is analogous; however, here α and β are densities, and thus may be larger than 1. Following [43], these densities are computed incrementally; α is computed forward in time, and β backwards in time (for which reason this algorithm is often referred to as the *forward-backward algorithm*. Initially,

$$\begin{aligned} \alpha_0^{(n)}(x) &= \pi^{(n)}(x) && (9) \\ \beta_T^{(n)}(x) &= 1 && (10) \end{aligned}$$

and for all other α_t and β_t:

$$\begin{aligned} \alpha_t^{(n)}(x) &= \int \alpha_{t-1}^{(n)}(x')\ \mu^{(n)}(x \mid x')\ \nu^{(n)}(O_t \mid x)\ dx' && (11) \\ \beta_t^{(n)}(x) &= \int \beta_{t+1}^{(n)}(x')\ \mu^{(n)}(x' \mid x)\ \nu^{(n)}(O_{t+1} \mid x')\ dx' && (12) \end{aligned}$$

Bayes rule governs the conditional density over the state space at time t:

$$\gamma_t^{(n)}(x) = \frac{\alpha_t^{(n)}(x)\ \beta_t^{(n)}(x)}{\int \alpha_t^{(n)}(x')\ \beta_t^{(n)}(x')\ dx'} \tag{13}$$

Similarly, the state transition densities $\xi^{(n)}$ are computed as

$$\xi_t^{(n)}(x, x') = \frac{\alpha_t^{(n)}(x)\ \mu^{(n)}(x' \mid x)\ \nu^{(n)}(O_{t+1} \mid x')\ \beta_{t+1}^{(n)}(x)}{\int\int \alpha_t^{(n)}(\bar{x})\ \mu^{(n)}(\bar{x}' \mid \bar{x})\ \nu^{(n)}(O_{t+1} \mid \bar{x}')\ \beta_{t+1}^{(n)}(\bar{x})\ d\bar{x}\ d\bar{x}'} \tag{14}$$

This computation is completely analogous to the finite case, replacing conditional probabilities by conditional densities.

The M-step uses $\gamma_t^{(n)}(x)$ and $\xi_t^{(n)}(x, x')$ to compute a new model $\lambda^{(n+1)}$, using the maximum likelihood estimator:

$$\begin{aligned} \pi^{(n+1)}(x) &= \gamma_0^{(n)}(x) && (15) \\ \mu^{(n+1)}(x' \mid x) &= \frac{\sum_{t=1}^{T-1} \xi_t^{(n)}(x, x')}{\sum_{t=1}^{T-1} \gamma_t^{(n)}(x)} && (16) \\ \nu^{(n+1)}(b \mid x) &= \frac{\sum_{t=1}^{T} I_{O_t=b}\ \gamma_t^{(n)}(x)}{\sum_{t=1}^{T} \gamma_t^{(n)}(x)} && (17) \end{aligned}$$

Here I_{cond} denotes an indicator variable that is 1 if the condition *cond* is true, and 0 otherwise. A straightforward result is the convergence of GHMMs under approprate conditions.

Theorem 1. (GHMM Convergence Theorem) *If all distributions of a GHMM λ possess density functions that are Lipschitz (and differentiable) in all variables, then for almost all (measure 1) points the steps outlined above do not decrease the probability density of the observation sequence. They do not improve the probability density of the observation sequence at each iteration if, and only if, the distributions at the beginning of an EM step are at a critical point (local maximum, minimum, or saddle point).*

Proof. Only sketched here (see also [2, 20, 33], and see [47] for an extension to certain real-valued spaces). For finite-state finite-observation HMMs, all densities and variables (μ, ν, π, α, β, γ, and ξ) can be implemented by vectors, and the Baum-Welch algorithm has been proven to converge [2]. Juang [19] has shown convergence to local maxima for HMMs with a finite number of states and continuous observations where the observation densities are a mixture of log concave or ellipsoidal symmetrical densities. In particular, the class of ellipsoidal symmetrical densities includes Gaussians. A GHMM can be viewed as the limit as the number of Gaussians is allowed to increase to infinity, and then the number of states is allowed to increase to infinity in Juang's analysis. Since Gaussians meet the assumptions of Juang's analysis, any continuous $\nu(O|x)$ can be the limit.

$$\lim_{K\to\infty} \sum_{k=1}^{K} c_{kx}\ b_{kx}(O) \tag{18}$$

where $\sum_k c_{kx} = 1$ and $b_{kx}(O)$ is a Gaussian.

Juang's analysis contains the following forms of manipulation:

$$\int_Y \int_X f(x,y)\ dx\ dy \quad = \quad \int_X \int_Y f(x,y)\ dy\ dx \tag{19}$$

$$\nabla_Y \int_X f(x,y)\ dx \quad = \quad \int_X \nabla_Y f(x,y)\ dx \tag{20}$$

where $f(x,y)$ is some function proportional to a density. Differentiability of all densities is a sufficient assumption for these statements to hold true. □

3 Density Approximation

This section describes sample-based and tree-based methods for density approximation. The notion of *asymptotic consistency* is introduced and results, along with error bounds, are given for a popular sampling algorithm and a tree method. Throughout this section, we assume that all density functions are centered on a compact and bounded region in $\Re^k$ (for an arbitrary k) and that they are Lipschitz.

3.1 Sampling

Samples are values drawn from the domain of a density function, where each sample is annotated by a non-negative real value [30, 36, 57]. Sample sets are (finite) sets of samples, annotated by numerical probability values. More specifically, let f be a probability density function, and let N denote a positive number (the cardinality of a sample set). Then a *sample set* is a set

$$X \quad = \quad \{\langle x_1, p_{x_1}\rangle \ \ldots, \langle x_N, p_{x_N}\rangle\} \quad \text{with} \ \sum_{n=1}^{N} p_{x_n} = 1 \tag{21}$$

where $x_n \in \mathrm{dom}(f)$ and $p_{x_n} \in [0,1]$ for all n with $1 \leq n \leq N$. Sample sets can be thought of as discrete distributions over the event space $\{x_1, \dots, x_N\}$ and probability distribution defined by $\{p_{x_1}, \dots, p_{x_N}\}$ [36, 41].

One popular sampling method is called *importance sampling*, which was originally introduced by Rubin [44]. Importance sampling approximates a density f by drawing samples x from a distribution with density g, and assigning a weight p_x proportional to $\frac{f(x)}{g(x)}$. Obviously, the density g must be non-zero over the support of f, i.e.,

$$f(x) > 0 \quad \Longrightarrow g(x) > 0 \tag{22}$$

We will distinguish two special cases:

1. **Likelihood-weighted sampling.** If $g \equiv f$, we will refer to the sampling algorithm as *likelihood-weighted sampling*. Here we draw samples according to f and assigns equal probability to each sample in the sample set [22]:

$$\begin{aligned} x_n & \quad \text{is drawn according to } f \\ p_{x_n} &= N^{-1} \end{aligned} \tag{23}$$

   In many cases, likelihood-weighted sampling can easily be implemented using *rejection sampling* [36].

2. **Uniform sampling.** If g is uniform over (a superset of) the support of f, we will call the sampling algorithm *uniform sampling*. This sampling algorithm draws values randomly (uniformly) from the support of f (which is assumed to be bounded), and assigns to each value x a probability proportional to its density $f(x)$:

$$\begin{aligned} x_n & \quad \text{is drawn uniformly from } \mathrm{dom}(f) \\ p_{x_n} &= f(x_n) \left[\sum_{i=1}^{N} f(x_i)\right]^{-1} \end{aligned} \tag{24}$$

Uniform sampling covers the domain of a density uniformly regardless of the nature of the density function. Likelihood-weighted sampling populates the space according to f, so that the density of sample is proportional to the density f. Likelihood-weighted sampling can be said to "waste" fewer samples in low-likelihood regions of f [41]. In other words, if X_{u} and X_{lw} are sample sets generated from the same distribution using uniform sampling, and likelihood-weighted sampling, respectively, the following holds true in expectation:

$$\sum_{\langle x, p_x \rangle \in X_{\mathrm{u}}} f(x) \quad \leq \quad \sum_{\langle x, p_x \rangle \in X_{\mathrm{lw}}} f(x) \tag{25}$$

were the equality holds in expectation if and only if f is uniform [51]. Henceforth, we will focus our attention on likelihood-weighted sampling whenever we have an explicit representation of a probability distribution (and importance sampling otherwise).

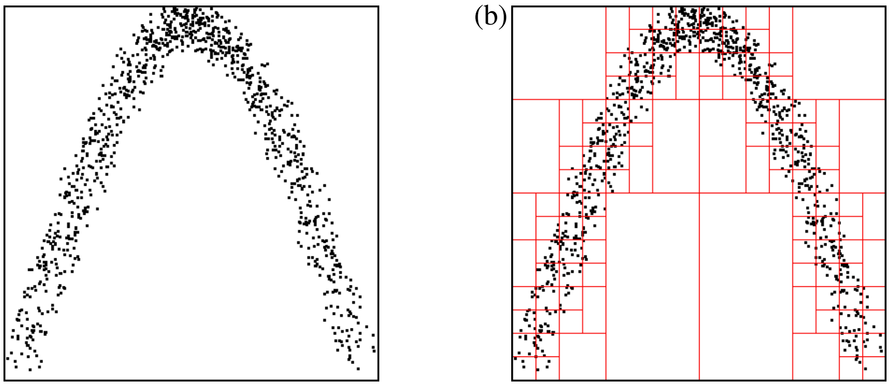

Figure 1: (a) Data set (b) partitioned by a density tree.

Figure 1a shows a sample set drawn by likelihood-weighted sampling from a distribution that resembles the shape of a sine wave in 2D. All probabilities p_x of the sample set shown there are the same, and the samples are concentrated in a small region of the $\Re^2$. In practice, likelihood-weighted sampling is often given preference over uniform sampling—specifically if the target density is known to only populate (with non-zero measure) a small subspace of its domain.

Both sampling methods can equally be applied to *sample from a sample set* (resampling). Let X be a sample set. Under uniform sampling, a new sample set is generated by randomly drawing samples $\langle x, p_x \rangle$ from X with a uniform distribution (regardless of the p_x values). The p_x-values in the new sample set are then re-normalized so that they add up to 1. Under likelihood-weighted sampling, samples are drawn from X according to the (discrete) probability distribution induced by their p_x-values. Each sample is then assigned the same probability. Sampling from sample sets plays an important role in the Monte Carlo HMM algorithm described below.

3.2 Asymptotic Consistency

The central idea of sampling is to *represent* density functions by finite sample sets, which are computationally advantageous. So what does it mean for a sample set X to "represent" a density f? Obviously, sample sets represent discrete distributions and thus cannot represent distributions that possess densities (hence are continuous). However, we will call a sampling method *asymptotically consistent* if for $N \to \infty$, X converges to f with probability 1 when integrated over the system of half-open Borel sets:

$$\lim_{N \to \infty} \sum_{\langle x, p_x \rangle \in X} I_{x_0 \leq x < x_1} \, p_x \quad = \quad \int_{x_0}^{x_1} f(x) \, dx \qquad \text{w.p. } 1 \tag{26}$$

Recall the system of half-open intervals is an inducing system for the sigma algebra over the $\Re^k$; hence, it suffices to show convergence for those sets.

A key observation is that both sampling algorithms are asymptotically consistent—as are all importance sampler with $f(x) > 0 \implies g(x) > 0$ [51]. This is formalized for the more important likelihood-weighted sampling technique in the following theorem, which also provides a probabilistic error bound for the finite case.

Theorem 2. *The likelihood-weighted sampling method is asymptotic consistent. For finite* N*,* $\varepsilon > 0$ *and* $x_0 < x_1$*, the likelihood that the error between a density* f *and a sample* X *is larger than* ε *can be bounded as follows:*

$$Pr\left(\left| \sum_{\langle x,p_x\rangle \in X} I_{x_0 \le x < x_1}\; p_x - \int_{x_0}^{x_1} f(x)\; dx \right| > \varepsilon \right) \quad \le \quad 2\; e^{-2\varepsilon^2 N} \tag{27}$$

Thus, according to the theorem the error between f and its sample decreases at the familiar rate $\frac{1}{\sqrt{N}}$.

Proof. The convergence follows directly from the Central Limit Theorem (see also [13, 41, 51]). The bound follows from the Hoeffding bound, under the observation that for any half-open interval $[x_0, x_1)$ each sample can be viewed as a zero-one "guess" of the size of the area $\int_{x_0}^{x_1} f(x)\; dx$. □

More generally, the convergence rate of importance sampling is in $O(\frac{1}{\sqrt{N}})$ if $\frac{f(x)}{g(x)} < \infty$. The variance of the error depends on the "mismatch" between f and g (see e.g., [51], pg. 33). The N will be reduced by a constant factor of $\max_x \frac{f(x)}{g(x)} < \infty$ for appropriate $g(x)$.

3.3 Density Trees

While sample sets are sufficient to approximate continuous-valued distributions, they differ from those in that they are *discrete*, that is, even in the limit they assign non-zero likelihood to only a countable number of points. This is problematic if one wants to *combine* densities represented through different sample sets: For example, let f and g be two density functions defined over the same domain, and let X be a sample of f and Y a sample of g. Then with probability 1, none of the samples in X and Y are identical, and thus it is *not* straightforward how to obtain an approximation of their product $f \cdot g$ from X and Y. Notice that multiplications of densities are required by the Baum-Welch algorithm (see e.g., Equation (13)).

Density trees, which are quite common in the statistical literature [24, 35, 38, 39], transform sample sets into density functions. Unfortunately, not all tree growing methods are asymptotically consistent when applied to samples generated from a density f. We will describe a simple algorithm which we will prove to be asymptotically consistent.

Our algorithm annotates each node in the tree with a hyper-rectangular subspace of $\mathrm{dom}(f)$, denoted by V (or V_i for the i-th node). Initially, all samples are assigned to the root node, which covers the entire domain of f. A node i is split whenever the following two conditions are fulfilled:

1. At least $\sqrt{N}$ samples $\langle x, p_x\rangle \in X$ fall into in V_i.

2. Its depth, i.e., its distance from the root node, does not exceed $\lfloor \frac{1}{4} \log_2 N \rfloor$.

If a node is split, its interval v is divided into two equally sized intervals along its longest dimension. These intervals are assigned to the two children of the node. Otherwise, a node becomes a leaf node of the density tree.

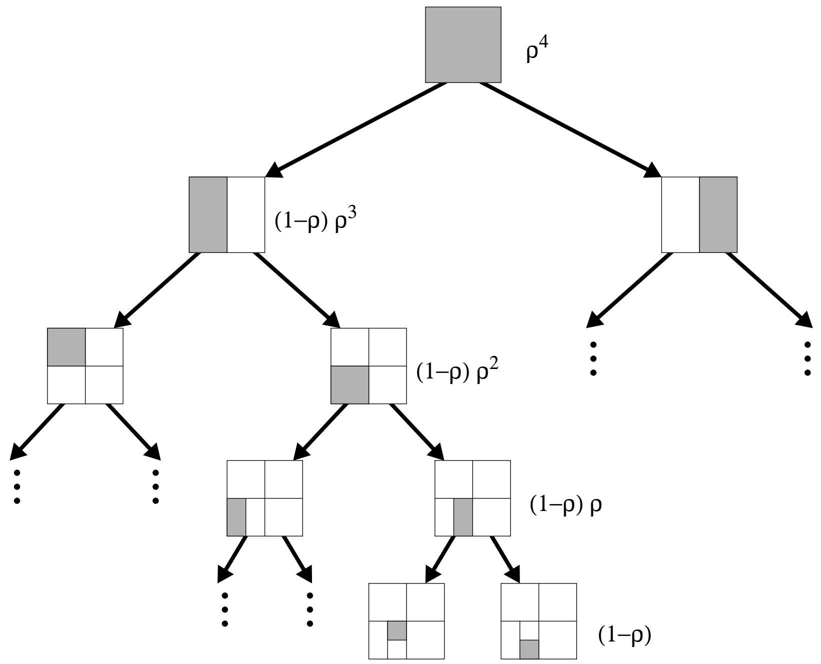

Figure 2: Density tree. Each branch cuts the area under a node into two equally-sized, rectangular halves, shown in grey.

An example of such a density tree defined over $\Re^2$ is illustrated in Figure 2. Here the root node covers the entire domain of f. Each child covers exactly half the space of each parent node. The areas covered by each leaf are mutually exclusive; their union covers the entire space. Figure 1b shows a density tree generated for the sample set shown in Figure 1a. The reader may notice that the criteria of the conditions above have been chosen to facilitate our proof of asymptotic consistency; in practice, a much wider range of criteria for stopping tree growth can be applied.

Density trees define density functions that continue discrete sample sets into $\Re^k$. Let f be a density function, X a sample drawn from this density, and for all $x \in \mathrm{dom}(f)$ let $i(x)$ denote the leaf whose region V_i contains x. Furthermore, let $\hat{\sigma}_i$ denote the relative frequency of samples in the i-th node weighted by their respective p_x values:

$$\hat{\sigma}_i \quad = \quad \sum_{\langle x, p_x \rangle \in X} I_{x \in V_i}\, p_x \tag{28}$$

Then the density function of a tree, denoted $\hat{f}$, is defined as follows:

$$\hat{f}(x) \quad = \quad \frac{\hat{\sigma}_{i(x)}}{|V_{i(x)}|} \qquad (\forall x \in \mathrm{dom}(f)) \tag{29}$$

The numerator of (29) describes the weighted relative frequency that a sample falls into the interval of the node $i(x)$. The denominator is the size of the interval, which is necessary to transform a relative frequency into a density over the interval $V_{i(x)}$.

The density function $\hat{f}(x)$ can be equally defined for *internal nodes* (non-leaf nodes). This will be important below, where estimates at different levels of the tree are mixed for regularization.

3.4 Asymptotic Consistency of Density Trees

An important result is the asymptotic consistency of density trees. This result, which justifies the use of density trees are non-parametric approximation, will be formalized in the following theorem:

Theorem 3. *Density trees are asymptotically consistent, that is, for all $x_0, x_1 \in \mathrm{dom(f)}$ with $x_0 < x_1$ and for any density tree $\hat{f}(x)$ generated from a sample set X generated from a density f, the following holds true:*

$$\lim_{N\to\infty} \int_{x_0}^{x_1} \hat{f}(x)\; dx \;\; = \;\; \int_{x_0}^{x_1} f(x)\; dx \qquad \textit{w.p. 1.} \tag{30}$$

We will first prove a related result, namely that of pointwise stochastic convergence

$$\lim_{N\to\infty} \hat{f}(x) \;\; = \;\; f(x) \qquad (\forall x \in \mathrm{dom}(f)) \qquad \text{w.p. 1} \tag{31}$$

which under the assumptions made in this paper implies the theorem. The proof is carried out in two stages, each of which is described by its own lemma.

Lemma 1. *The density $\hat{f}$ converges to a function $\bar{f}$ which is defined through the same tree as $\hat{f}$, but with the weighted relative frequencies $\hat{\sigma}_i$ replaced by their true frequencies, denoted $\bar{\sigma_i}$:*

$$\bar{\sigma_i} \;\; = \;\; \int_{V_i} f(x)\; dx. \tag{32}$$

Lemma 2. *The density $\hat{f}$ converges to f.*

Proof of Lemma 1. For each leaf i, the Hoeffding bound states that

$$Pr(|\bar{\sigma_i} - \hat{\sigma}_i| > \varepsilon) \;\; < \;\; 2\; e^{-2\varepsilon^2 N}. \tag{33}$$

Our tree growing rule limits the depth of the tree to $\lfloor \frac{1}{4} \log_2 N \rfloor$. Hence, there are at most

$$2^{\frac{1}{4}\log_2 N} \;\; = \;\; N^{\frac{1}{4}} \tag{34}$$

nodes in the tree. Thus, the probability that there exists a leaf whose empirical frequency $\hat{\sigma}_i$ deviates from the true probability $\bar{\sigma_i}$ is bounded by

$$Pr(\exists\ \text{leaf}\ i : |\bar{\sigma_i} - \hat{\sigma}_i| > \varepsilon) \;\; < \;\; 2\; N^{\frac{1}{4}}\; e^{-2\varepsilon^2 N}. \tag{35}$$

The error of the empirical frequency estimates is now translated into density errors. Recall that according to (29), the relative frequency is related to the density of a tree by the volume V_i of each leaf. Consequently,

$$|\bar{f}(x) - \hat{f}(x)| \;\; = \;\; |V_i|^{-1}\; |\bar{\sigma_i} - \hat{\sigma}_i| \tag{36}$$

Observing that the volume of the interval covered by a leaf $|V_i|$ is at least $N^{-\frac{1}{4}}$—which directly follows from the depth limit imposed when growing density trees—we obtain

$$
\begin{aligned}
|\bar{f}(x) - \hat{f}(x)| \quad &\leq \quad N^{-\frac{1}{4}}\ |\bar{\sigma}_i - \hat{\sigma}_i| && (37)\\
\Longleftrightarrow \quad N^{\frac{1}{4}}\ |\bar{f}(x) - \hat{f}(x)| \quad &\leq \quad |\bar{\sigma}_i - \hat{\sigma}_i| && (38)
\end{aligned}
$$

Substituting this into (35) yields:

$$
Pr(|\bar{f}(x) - \hat{f}(x)| > N^{\frac{1}{4}}\varepsilon) \quad < \quad 2\ N^{\frac{1}{4}}\ e^{-2\varepsilon^2 N}, \tag{39}
$$

which with the substitution $\varepsilon' = N^{\frac{1}{4}}\varepsilon$ leads to the following error bound between $\hat{f}$ and $\bar{f}$:

$$
Pr(|\bar{f}(x) - \hat{f}(x)| > \varepsilon') \quad < \quad 2\ N^{\frac{1}{4}}\ e^{-2\varepsilon'^2\sqrt{N}}. \tag{40}
$$

Obviously, the right hand-size of (40) converges to 0 as $N \to \infty$, which proves pointwise convergence of $\hat{f}$to$\bar{f}$. □

It remains to be shown that $\bar{f}$ converges of f.

<u>Proof of Lemma 2.</u> It suffices to show that with high likelihood, any leaf i that covers an interval with non-zero measure, i.e.,

$$
\bar{\sigma}_i \quad > \quad 0 \tag{41}
$$

will be split infinitely often as N is increased. The desired convergence then follows directly from the fact that f obeys a Lipschitz condition. The proof is given for likelihood-weighted sampling.

Let i be a leaf node, and let $\mathrm{depth}(i)$ denote its depth. Furthermore, let n_i be the number of samples in node i:

$$
n_i \quad = \quad \sum_{\langle x,p_x\rangle \in X} I_{x\in V_i} \quad = \quad N\ \hat{\sigma}_i \tag{42}
$$

The second equality exploits the assumption that samples are generated by likelihood-weighted sampling. Recall that the node i is split if $n_i \geq \sqrt{N}$, for sufficiently large N. Without loss of generality, let us assume that

$$
N \quad > \quad \max\left\{\frac{1}{(\bar{\sigma}_i - \varepsilon)^2}\ ,\ 4^{\mathrm{depth}(i)}\right\} \qquad \text{and} \qquad \varepsilon < \bar{\sigma}_i \tag{43}
$$

The second term in the max ensures that the depth limit $\lfloor \frac{1}{4}\log_2 N \rfloor$ in the tree growing rule does not restrict splitting node i. The other assumptions (43) imply that

$$
\bar{\sigma}_i \quad > \quad \frac{1}{\sqrt{N}} + \varepsilon \tag{44}
$$

The Hoeffding bound now yields the desired result:

$$
\begin{aligned}
Pr(\bar{\sigma}_i - \hat{\sigma}_i > \varepsilon) \quad &\leq \quad e^{-2\varepsilon^2 N} && (45)\\
\Longrightarrow \quad Pr\left(\frac{1}{\sqrt{N}} + \varepsilon - \hat{\sigma}_i > \varepsilon\right) \quad &\leq \quad e^{-2\varepsilon^2 N} && (46)
\end{aligned}
$$

$$\Longleftrightarrow \quad Pr\left(\frac{1}{\sqrt{N}} > \hat{\sigma}_i\right) \quad \leq \quad e^{-2\varepsilon^2 N} \tag{47}$$

$$\Longleftrightarrow \quad Pr\left(\frac{1}{\sqrt{N}} > \frac{n_i}{N}\right) \quad \leq \quad e^{-2\varepsilon^2 N} \tag{48}$$

$$\Longleftrightarrow \quad Pr\left(\sqrt{N} > n_i\right) \quad \leq \quad e^{-2\varepsilon^2 N} \tag{49}$$

Thus, with high probability $n_i \geq \sqrt{N}$ and node i is split. The pointwise convergence of $\bar{f}$ to f now follows from our assumption that f is Lipschitz. □

Proof of the Theorem 3. We have already shown that

$$\lim_{N\to\infty} \hat{f}(x) \quad = \quad f(x) \qquad \text{w.p. 1 and } \forall x \in \mathrm{dom}(f). \tag{50}$$

Theorem 3 follows trivially from the triangle inequality, since $\mathrm{dom}(f)$ is bounded and Lipschitz (which implies that f is bounded). □

The termination conditions for growing trees (see itemized list in Section 3.3) where chosen to facilitate the derivation of Theorem 3. In practice, these conditions can be overly restrictive, as they often require large sample sets to grow reasonable-sized trees. Our actual implementation sidesteps these conditions, and trees are grown all the way to the end. While the convergence results reported here are not applicable any longer, our implementation yielded much better performance specifically when small sample sets were used (e.g., 100 samples).

4 Regularization Through Shrinkage and Annealing

We will now resume our consideration of GHMMs. The continuous nature of the state space in GHMMs, if represented by arbitrary sample sets and/or trees, can easily overfit *any* data set, no matter how large. This is because GHMMs are rich enough to assign a different state to each observation in the training data (of which there are only finitely many), making it essentially impossible to generalize beyond sequences other than the ones presented during training. A similar problem arises in conventional HMMs, if they are given more states than samples in the data set. In GHMMs the problem is inherent, due to the topology of continuous spaces. Thus, some kind of regularization is needed to prevent overfitting from happening.

Our approach to regularization is based on *shrinkage* [50]. Shrinkage is a well-known statistical technique for "lumping together" different estimates from different data sources. In a remarkable result by Stein [50], shrinking estimators were proven to yield uniformly better solutions over unbiased maximum-likelihood estimators in multivariate Gaussian estimations problems (see also [53]). Shrinkage trees were introduced in [32]. Instead of using the density estimates at the leafs of a tree, shrinkage trees mix those densities with densities obtained further up in the tree. These internal densities are less specific to the region covered by a leaf node; however, their estimates are usually obtained from more data, making them less susceptible to variance in the data.

Figure 3 shows an example using shrinkage with an exponential factor, parameterized by ρ (with $0 \leq \rho \leq 1$). Here each node in the tree, with the exception of the root node, weighs its own density estimate by $(1-\rho)$, and mixes it with the density estimate from its parent using the weighting factor ρ. As a result, every node along the path contributes to the density estimate at the

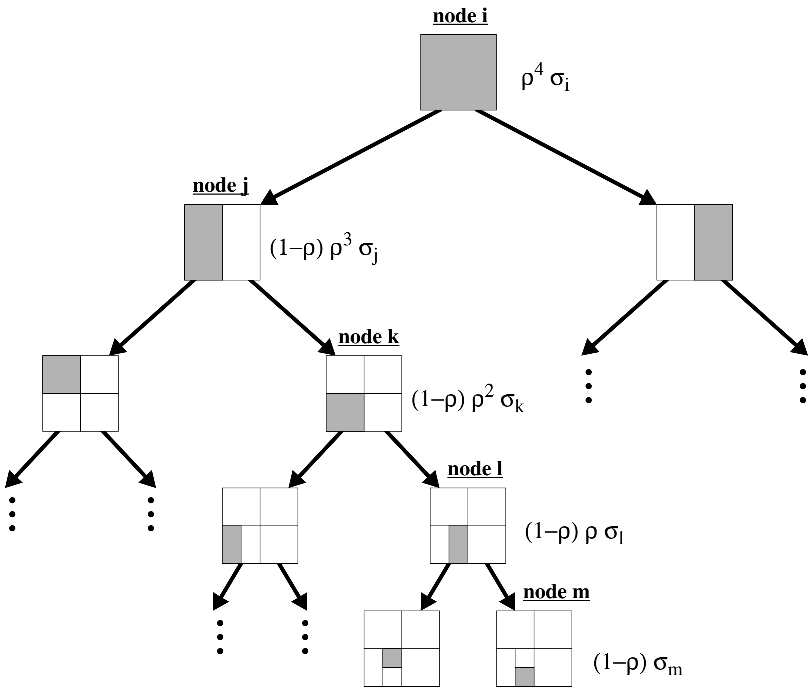

Figure 3: Shrinkage for complexity control. The density of each node is an exponentially weighted mixture of densities of the nodes along the path to the leaf. Shown here is an example for node m, whose path leads through nodes i, j, k, and l. The ρ-terms describe the mixture coefficients for this example. If ρ is 0, only the leaf estimate is used. For larger values of ρ, estimates from data beyond the leaf data are used to estimate the target density.

leaf; however, its influence decays exponentially with the distance from the leaf node. Obviously, the value of ρ determines the amount of shrinkage. If $\rho = 1$, only the root node is consulted, hence, the probability density induced by the tree is uniform. If $\rho = 0$, on the other hand, there is no shrinking and only the estimates in the leaf nodes determine the shape of the density function. For intermediate values of ρ, estimates along the entire path are combined.

Since the optimal value of ρ is problem-specific—de facto it depends on the nature of the (unobservable) state space—our approach uses annealing and cross validation to determine the best value for ρ. More specifically,

$$\rho^{(n)} \quad = \quad \bar{\rho}^{n-1} \tag{51}$$

where $\bar{\rho} < 1$ is a constant (e.g., 0.9) and n denotes the iteration counter of the Baum-Welch algorithm (starting at 1). Thus, ρ starts with $\rho^{(0)} = 1$, for which nothing can be learned, since every density is uniform. The parameter ρ is then annealed towards zero in an exponential fashion. Cross validation (early stopping) is applied to determine when to stop training.

Table 1: The MCHMM algorithm at-a-glance.

Model Initialization: Initialize $\lambda = \{\pi, \mu, \nu\}$ by three randomly drawn sets of samples of the appropriate dimension. Generate density trees from these samples. Set $\rho = 1$, and chose an initial sample set size $N > 0$.

E-step:

1. Copy the sample set representing π into α_0 (c.f., (9)).
2. Set $\beta_T = 1$.
3. Computation of α_t (c.f., (11)). For each t with $1 < t \leq T$ do:
   (a) Generate N samples $\langle x', p_{x'} \rangle$ from the sample set representing α_{t-1} using likelihood-weighted sampling.
   (b) For each sample $\langle x', p_{x'} \rangle$, generate the conditional density $\mu(x \mid x')$ using the tree-version of μ. Sample a single x from this tree, using likelihood-weighted sampling.
   (c) Set p_x to a value proportional to $\nu(O_t \mid x)$, where O_t is the t-th observation in the data set. This density value is obtained using the tree representing ν.
   (d) Generate a tree from the new sample set.
4. Computation of β_t (c.f., (12)). For each t with $1 \leq t < T$ do:
   (a) Generate N samples $\langle x', p_{x'} \rangle$ from the sample set representing β_{t+1} using likelihood-weighted sampling.
   (b) For each sample $\langle x', p_{x'} \rangle$, generate the conditional density $\mu(x' \mid x)$ using the tree-version of μ. Sample a single x from this tree, using likelihood-weighted sampling.
   (c) Set p_x to a value proportional $\nu(O_{t+1} \mid x')$, where O_{t+1} is the $t+1$-th observation in the data set. This density value is obtained using the tree representing ν.
   (d) Generate a tree from the new sample set.
5. Computation of γ_t (c.f., (13)). For each t with $1 \leq t \leq T$ do:
   (a) Generate $N/2$ sample from α_t by likelihood weighted sampling and assign to each sample $\langle x, p_x \rangle$ a probability proportional to $\beta_t(x)$, using the tree approximation of β_t.
   (b) Generate $N/2$ sample from β_t by likelihood weighted sampling and assign to each sampled $\langle x$ a probability p_x proportional to $\alpha_t(x)$, using the tree approximation of α_t.

M-step:

1. Estimation of the new state transition density μ (c.f., (16)): Pick N random times $t \in \{1, \ldots, T-1\}$ and generate samples $\langle x, p_x \rangle$ and $\langle x', p_{x'} \rangle$ from γ_t, and γ_{t+1}, respectively, by likelihood-weighted sampling. Add $\langle (x, x'), N^{-1} \rangle$ into the sample set representing μ. Generate a tree from the sample set.
2. Estimation of the new observation density ν (c.f., (17)): Pick N random $t \in \{1, \ldots, T\}$ and generate a sample $\langle x, p_x \rangle$ from γ_t by likelihood-weighted sampling. Add $\langle (x, O_t), N^{-1} \rangle$ into the sample set representing ν. Generate a tree from the sample set.
3. Estimation of the new initial state distribution π (c.f., (15)): Copy the sample set γ_0 into π. Generate a tree from the sample set.

Annealing: Set $\rho \leftarrow \rho\bar{\rho}$. Stop when the likelihood of an independent cross-validation set is at its maximum.

Sample set size: Increase N.

5 Monte Carlo HMMs

We are now ready to present the main algorithm of this paper, along with the main theoretical result: The Monte Carlo algorithm for GHMMs, called *Monte Carlo hidden Markov models* (in short *MCHMM*). A MCHMM is a computational instantiation of a GHMM that represents all densities through samples and trees. It applies likelihood-weighted sampling for forward and backward projection (c.f., Equations (11) and (12)), and it uses annealing and cross-validation to determine the best shrinkage factor. To ensure convergence, the number of samples N is increased over time.

The learning algorithm for MCHMM is depicted in Table 1. MCHMMs use both sample-based and tree representations during learning. After learning, it suffices to store only the tree-based version of the model $\lambda = \{\pi, \mu, \nu\}$; all sample sets can be discarded. When applying a trained MCHMM to a new data set (e.g., for analysis, prediction, or classification), only the "forward" densities $\alpha_t(x)$ have to be estimated (just like in conventional HMMs, Kalman filters [21, 31], or dynamic belief networks [8, 45]). For that, no trees have to be grown. Instead, samples for $\alpha_{t+1}(x)$ are obtained by sampling from the sample set representing $\alpha_t(x)$ (see also Section 3.1) and the tree representing μ. The p_x-values of $\alpha_{t+1}(x)$ are determined using the tree representing ν. This recursive resampling technique, known as sampling/importance resampling [44, 49] applied to time-invariant Markov chains, converges at the rate $1/\sqrt{N}$ (if T is finite, c.f., Section 3.2). Sampling/importance resampling, which will further be discussed in Section 8, has been successfully applied in domains such as computer vision and robotics [11, 18].

6 Error Analysis and Convergence Results

This section presents the major convergence result for MCHMM. Under mild assumptions, MCHMMs can be shown to converge with probability 1 to models λ that locally maximize likelihood; just like conventional HMMs. The proof builds on the well-known results for discrete HMMs (see g.e., [2, 19, 20, 33, 47]), and shows that if the sample size N is sufficiently large, the deviation between the MCHMMM and the corresponding GHMM can be bounded arbitrarily tightly with high probability, exploiting the asymptotic consistency of the two major approximations: samples and density trees.

6.1 Relative Error

The process of Monte Carlo simulation introduces error into the Baum-Welch algorithm. In order to understand how this affects convergence, we need to understand how an initial error will propagate through an EM step. This analysis will also cover errors introduced in the middle of an EM step as long as such errors converge to 0 with large samples, because errors introduced in mid-step will be indistinguishable from an error introduced at the beginning of the calculation.

What we would like to prove is that a small absolute initial error will imply a small absolute final error. Unfortunately, this is not possible because several calculations are normalized integrals of products. Consider the integral $\int_x f(x)g(x)dx$. If $f(x) = step(x - 1/2)$ and $g(x) = 1 - f(x)$ then $\int_x f(x)g(x)\ dx = 0$. However, if $\bar{f}(x) = f(x) + \varepsilon$ and $\bar{g}(x) = g(x) + \varepsilon$ are integrated, we get a non-zero integral: $\int_{x=0}^{1} \bar{f}(x)\bar{g}(x)dx = \varepsilon$. First, notice that the calculation of $\alpha_t(x)$ is of this form. Assume, for the moment, that $\beta_t(x) = 1$. Then $\gamma_t(x) = \alpha_t(x) / \int \alpha_t(x')dx'$. Consequently, if $\alpha_t(x)$ was small ($< \varepsilon$) everywhere, $\bar{\gamma}_t(x)$ can potentially be very far from $\gamma_t(x)$. Furthermore, any allowable error can produce an arbitrarily large error in the output.

However, if we have a small *relative* error,

$$\frac{\bar{f}(x) - f(x)}{f(x)} \quad < \quad \varepsilon \tag{52}$$

we will be able to prove that a small initial error implies a small final error. The restriction to small relative error is significant because density trees only guarantee a small absolute error. An extra assumption must be placed on the initial distribution in order to guarantee that the density tree produced by sampling from the distribution will have small relative error.

To simplify the proof that a small initial error produces a small final error, we only consider transforming functions to first order, $f(\bar{x}) \sim f'(x)(\bar{x} - x)$. There is no zeroth order error because the limit as the relative error approaches 0 is $f(x)$, $\lim_{\bar{x} \to x} f(\bar{x}) = f(x)$. Higher order error becomes small quickly for all transforming functions which we consider. Consequently, we will be able to state that for all ε, δ, there exists some number of samples, N, s.t. $f(\bar{x}) - f(x) \leq f'(x)(\bar{x} - x) + \varepsilon$ with probability $1 - \delta$.

Our analysis does not consider the shrinkage factor ρ, which, since $\lim_{n \to \infty} \rho^{(n)} = 0$, has asymptotically no effect.

6.2 Convergence of MCHMMs

We will now state the central result of this section: the MCHMM convergence theorem and an important corollary. The proof of the theorem will be developed throughout the remainder of this section.

Theorem 4 (MCHMM Convergence Theorem). *If approximation of the underlying distributions by density trees causes only a finite relative error which decreases to* 0 *as* $N \to \infty$ *and an EM step in a GHMM starting with* $\pi^{(n)}, \mu^{(n)}, \nu^{(n)}$ *produces output distributions* $\pi^{(n+1)}, \mu^{(n+1)}, \nu^{(n+1)}$ *then for all* ε, δ *there exists an* N *s.t. the output of an MCHMM iteration taking* $\pi^{(n)}, \mu^{(n)}, \nu^{(n)}$ *as inputs with probability* $1-\delta$ *satisfies* $|\pi^{(n+1)}(x) - {\pi'}^{(n+1)}(x)| < \varepsilon$, $|\mu^{(n+1)}(x'|x) - {\mu'}^{(n+1)}(x'|x)| < \varepsilon$, *and* $|\nu^{(n+1)}(O_t|x) - {\nu'}^{(n+1)}(O_t|x)| < \varepsilon$ *where* ${\pi'}^{(n+1)}$, ${\mu'}^{(n+1)}$, *and* ${\nu'}^{(n+1)}$ *are the output distributions of the MCHMM iteration.*

The proof of Theorem 4 will be presented below, after introducing a collection of useful lemmas. However, Theorem 4 implies the following important corollary:

Corollary. *Under the same assumptions as in Theorem 4, any strictly monotonically increasing schedule of* N*'s will cause a MCHMM to converge to a local maximum in with probability 1.*

Proof of the corollary. The convergence of GHMMs is a straightforward extension of the well-known convergence proof for HMMs as outlined in the proof of Theorem 1. The action of a GHMM iteration causes movement through the space of distributions. We can view this action of a GHMM iteration as inducing a vector field in the (infinite dimensional) space of distributions with a magnitude proportional to the change in likelihood and a direction given by $\lambda^{(n)}(x, x') -$

$\lambda^{(n+1)}(x, x')$. The action of an MCHMM with probability $1 - \delta$ will have a result in a perturbed vector with the error bounded by ε. The repeated action of a MCHMM with N increasing will create a Cauchy sequence where the ε decreases to 0 and δ decreases to 0. This Cauchy sequence will limit to local maxima with probability 1. □

It is important to notice that this corollary does *not* state that an MCHMM and GHMM starting with the same distribution will converge to the *same* local maximum. Such a result will generally not hold true, as the finiteness of N might influence the specific local maximum to which a MCHMM converges.

It is also interesting to note that the noise introduced by MCHMMs in the convergence step may be beneficial. The argument is informal because the noise of an MCHMM has both systematic (due to representation) and random (due to sampling) effects. First, consider a GHMM which accidentally starts out at a local minimum (or a saddle point). The GHMM might be stuck here, but a MCHMM will, with probability 1, step off of the local minima and move towards a local maxima. In addition, consider a landscape containing many local maxima on it's flanks. A GHMM could easily become trapped in a local maxima while an MCHMM with a low N will be kicked out of the local maxima with high probability. In this way, an MCHMM could gain some of the benefits of simulated annealing.

6.3 Propagation of Error

Before we can prove the MCHMM convergence theorem, we will need to develop an analysis of the propagation of errors through transforming functions.

Let us start by assuming that we have some initial error $\overline{\pi}(x) = \pi(x) + \Delta_\pi(x)$, $\overline{\mu}(x|x') = \mu(x|x') + \Delta_\mu(x, x')$ and $\overline{\nu}(O_t|x) = \nu(O_t|x) + \Delta_\nu(O_t, x)$. These errors can, for example, be the errors introduced by approximating a distribution with density trees. It will also be convenient to talk about the relative error

$$\Delta^r_\pi(x) = \left|\frac{\Delta_\pi(x)}{\pi(x)}\right| \tag{53}$$

$$\Delta^r_\mu(x|x') = \left|\frac{\Delta_\mu(x|x')}{\mu(x|x')}\right| \tag{54}$$

$$\Delta^r_\nu(O_t|x) = \left|\frac{\Delta_\nu(O_t|x)}{\nu(O_t|x)}\right| \tag{55}$$

$$\Delta^r_f(y) = \left|\frac{\Delta_f(y)}{f(y)}\right| \tag{56}$$

and the largest relative error over the input range

$$\Delta^r_\pi = \max_x \Delta^r_\pi(x) \tag{57}$$

$$\Delta^r_\mu = \max_{x,x'} \Delta^r_\mu(x|x') \tag{58}$$

$$\Delta^r_\nu = \max_{x,O_t} \Delta^r_\nu(O_t|x) \tag{59}$$

$$\Delta^r_f = \max_y \Delta^r_f(y) \tag{60}$$

The propagation of errors through individual calculations of Baum-Welch will be characterized by taking a derivative. Let $\bar{x} = x + \Delta(x)$. The definition of a derivative is $\frac{d}{dx} f(x) = \lim_{\bar{x} \to x} \frac{f(\bar{x}) - f(x)}{\bar{x} - x}$. As the difference between 2 distributions, $\bar{x}$ and x approaches 0, the difference between the resulting calculations, $f(\bar{x})$ and $f(x)$, will approach $f'(x)\Delta(x)$ implying $\Delta_f(x) = f(\bar{x}) - f(x) = f'(x)\Delta(x)$ in the limit that $\Delta(x) \to 0$. In particular,

$$\Delta_{f+g}(x) = \Delta_f(x) + \Delta_g(x) \tag{61}$$

$$\Delta_{\frac{g}{f}}(x) = \frac{-g(x)\Delta_f(x) + f(x)\Delta_g(x)}{f(x)^2} \tag{62}$$

$$\Delta_{fg}(x) = f(x)\Delta_g(x) + g(x)\Delta_f(x) \tag{63}$$

$$\Delta_{\int f} = \int \Delta_f \tag{64}$$

The last statement is only true for integration over compact regions with finite valued $f(x)$, because $\int \bar{f}(x) - f(x) dx = \int \bar{f}(x) dx - \int f(x) dx$ under these assumptions. These assumptions were already made to prove convergence of density trees so they are not an additional restriction.

6.4 Error Bounds for Auxiliary Variables

Given the above rules and notation we can calculate how the error will propagate through the calculation of a $\alpha_t(x)$. The following lemmas establish bounds on the relative errors of all quantities calculated in a Baum-Welch iteration given some initial error. These lemmas are necessary to prove the MCHMM convergence theorem.

Lemma 3. $\Delta^r_{\alpha_t} \leq (t-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi$ *to first order.*

Proof. According to the property (64),

$$\Delta_{\alpha_t} = \Delta_{\int \alpha_{t-1}\mu\nu}(x) = \int_x \Delta_{\alpha_{t-1}\mu\nu}(x)\ dx \tag{65}$$

Using (63), this expression is equivalent to

$$= \int_x \mu(x)\nu(x)\Delta_{\alpha_{t-1}}(x) + \nu(x)\alpha_{t-1}(x)\Delta_\mu(x) + \mu(x)\alpha_{t-1}(x)\Delta_\nu(x)\ dx$$

$$= \int_x \mu(x)\nu(x)\alpha_{t-1}(x)\left(\frac{\Delta_{\alpha_{t-1}}(x)}{\alpha_{t-1}(x)} + \frac{\Delta_\nu(x)}{\nu(x)} + \frac{\Delta_\mu(x)}{\mu(x)}\right)\ dx \tag{66}$$

Notice that the integral can be bounded by a maximum,

$$\Delta_{\alpha_t} \leq \alpha_t(x) \max_x \left(\left|\frac{\Delta_{\alpha_{t-1}}(x)}{\alpha_{t-1}(x)} + \frac{\Delta_\nu(x)}{\nu(x)} + \frac{\Delta_\mu(x)}{\mu(x)}\right|\right) \tag{67}$$

and,

$$\Delta_{\alpha_t} \geq -\alpha_t(x) \max_x \left(\left|\frac{\Delta_{\alpha_{t-1}}(x)}{\alpha_{t-1}(x)} + \frac{\Delta_\nu(x)}{\nu(x)} + \frac{\Delta_\mu(x)}{\mu(x)}\right|\right) \tag{68}$$

Division by $\alpha_t(x)$and application of the triangle inequality yields the relative error bounds

$$\frac{\Delta_{\alpha_t}}{\alpha_t(x)} \leq \max_x \left(\left| \frac{\Delta_{\alpha_{t-1}}(x)}{\alpha_{t-1}(x)} \right| + \left| \frac{\Delta_\nu(x)}{\nu(x)} \right| + \left| \frac{\Delta_\mu(x)}{\mu(x)} \right| \right) \tag{69}$$

$$\frac{\Delta_{\alpha_t}}{\alpha_t(x)} \geq -\max_x \left(\left| \frac{\Delta_{\alpha_{t-1}}(x)}{\alpha_{t-1}(x)} \right| + \left| \frac{\Delta_\nu(x)}{\nu(x)} \right| + \left| \frac{\Delta_\mu(x)}{\mu(x)} \right| \right) \tag{70}$$

Consequently, we have that $\Delta^r_{\alpha_t} \leq \Delta^r_{\alpha_{t-1}} + \Delta^r_\nu + \Delta^r_\mu$ The equation is recursive, terminating for $t = 0$ with $\Delta^r_{\alpha_1} = \Delta^r_\pi$. At each step in the recursion, at most $\Delta^r_\nu + \Delta^r_\mu$ is added to the error which implies

$$\Delta^r_{\alpha_t} \leq (t-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi$$

which proves Lemma 3. □

The calculation of the relative β_t error is analogous, and the resulting bound it stated in the following lemma:

Lemma 4. $\Delta^r_{\beta_t} \leq (T-t)(\Delta^r_\nu + \Delta^r_\mu)$ *to first order.*

The proof is omitted, since it is similar to the proof for $\Delta^r_{\alpha_t}$.

The calculation of the error of γ_t is more difficult.

Lemma 5. $\Delta^r_\gamma \leq 2(\Delta^r_\alpha + \Delta^r_\beta) = 2(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu))$ *to first order.*

Proof. To simplify the notation, we will assume a t subscript in the following proof. The results are independent of t.

$$\begin{aligned}
\Delta_\gamma(x) &= \Delta_{\frac{\alpha\beta}{\int \alpha\beta}}(x) \\
&= \frac{\Delta_{\alpha\beta}(x) \int_x \alpha(x)\beta(x)dx - \alpha(x)\beta(x)\Delta_{\int \alpha\beta}(x)}{\left(\int_x \alpha(x)\beta(x)\right)^2} \\
&= \frac{[\beta(x)\Delta_\alpha(x) + \alpha(x)\Delta_\beta(x)] \int_x \alpha(x)\beta(x)dx}{(\int_x \alpha(x)\beta(x)dx)^2} \\
&\qquad - \frac{\alpha(x)\beta(x) \int_x [\alpha(x)\Delta_\beta(x) + \beta(x)\Delta_\alpha(x)]dx}{(\int_x \alpha(x)\beta(x)dx)^2} \\
&\leq \frac{[\beta(x)\Delta_\alpha(x) + \alpha(x)\Delta_\beta(x)] \int_x \alpha(x)\beta(x)dx}{(\int_x \alpha(x)\beta(x)dx)^2} \\
&\qquad + \frac{\alpha(x)\beta(x)[\int_x \alpha(x)\beta(x)dx] \max_x \left(\left|\frac{\Delta_\beta(x)}{\beta(x)}\right| + \left|\frac{\Delta_\alpha(x)}{\alpha(x)}\right| \right)}{(\int_x \alpha(x)\beta(x)dx)^2} \\
&\leq \alpha(x)\beta(x) \frac{\left[\frac{\Delta_\alpha(x)}{\alpha(x)} + \frac{\Delta_\beta(x)}{\beta(x)}\right] + \max_x \left(\left|\frac{\Delta_\beta(x)}{\beta(x)}\right| + \left|\frac{\Delta_\alpha(x)}{\alpha(x)}\right| \right)}{\int_x \alpha(x)\beta(x)dx} \qquad (71)
\end{aligned}$$

Through similar manipulations, we get:

$$\Delta_\gamma(x) \geq -\alpha(x)\beta(x) \frac{\left[\frac{\Delta_\alpha(x)}{\alpha(x)} + \frac{\Delta_\beta(x)}{\beta(x)}\right] + \max_x \left(\left|\frac{\Delta_\beta(x)}{\beta(x)}\right| + \left|\frac{\Delta_\alpha(x)}{\alpha(x)}\right| \right)}{\int_x \alpha(x)\beta(x)dx} \tag{72}$$

Now, we can divide by a factor of γ to get

$$\Delta^r_\gamma \quad \leq \quad 2(\Delta^r_\alpha + \Delta^r_\beta) \tag{73}$$

and apply Lemmas 3 and 4 for any value of t to get

$$\Delta^r_\gamma \quad \leq \quad 2(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu)) \tag{74}$$

which completes the proof. □

The bounds for Δ^r_ξ are similar to those of Δ^r_γ, as documented by the following lemma.

Lemma 6. $\Delta^r_\xi \leq 2(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu))$ *to first order.*

The proof of Lemma 6 is analogous to that of Lemma 5 and therefore omitted. Notice that the bounds of Δ^r_ξ and Δ^r_γ are independent of t.

The M step calculations are done similarly. Let π' = the next π distribution, μ' =the next μ distribution, and ν' = the next ν distribution. Then the following lemmas provide bounds for the error of π', μ', and ν'.

Lemma 7. $\Delta^r_{\pi'} = 2(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu))$ *to first order.*

Proof. Lemma 7 follows from Lemma 5, since $\Delta^r_{\pi'} = \Delta^r_{\gamma_0}$.

Lemma 8. $\Delta^r_{\mu'} = 4(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu))$ *to first order.*

Proof. By property (62),

$$\begin{aligned}
\Delta_{\mu'}(x'|x) &= \Delta_{\frac{\sum\xi}{\sum\gamma}}(x'|x) \\
&= \left[-\left(\sum_{t=1}^{T-1}\gamma_t(x)\right)\Delta_{\sum\xi}(x,x') + \left(\sum_{t=1}^{T-1}\xi_t(x,x')\right)\Delta_{\sum\gamma}(x)\right]\frac{1}{\left(\sum_{t=1}^{T-1}\gamma_t(x)\right)^2} \\
&= \left[-\left(\sum_{t=1}^{T-1}\gamma_t(x)\right)\left(\sum_{t=1}^{T-1}\Delta_{\xi_t}(x,x')\right)\right. \\
&\qquad \left. + \left(\sum_{t=1}^{T-1}\xi_t(x,x')\right)\left(\sum_{t=1}^{T-1}\Delta_{\gamma_t}(x)\right)\right]\frac{1}{\left(\sum_{t=1}^{T-1}\gamma_t(x)\right)^2} \\
&= \left[\sum_{t=1}^{T-1}\xi_t(x,x')\right]\left[-\frac{\sum_{t=1}^{T-1}\Delta_{\xi_t}(x,x')}{\sum_{t=1}^{T-1}\xi_t(x,x')} + \frac{\sum_{t=1}^{T-1}\Delta_{\gamma_t}(x)}{\sum_{t=1}^{T-1}\gamma_t(x)}\right]\frac{1}{\sum_{t=1}^{T-1}\gamma_t(x)} \\
&\leq \left[\sum_{t=1}^{T-1}\xi_t\right]\max_t\left(\frac{\Delta_{\xi_t}}{\xi_t} + \frac{\Delta_{\gamma_t}}{\gamma_t}\right)\frac{1}{\sum_{t=1}^{T-1}\gamma_t}
\end{aligned} \tag{75}$$

and, analogously,

$$\Delta_{\mu'}(x'|x) \quad \geq \quad -\left[\sum_{t=1}^{T-1}\xi_t\right]\max_t\left(\frac{\Delta_{\xi_t}}{\xi_t} + \frac{\Delta_{\gamma_t}}{\gamma_t}\right)\frac{1}{\sum_{t=1}^{T-1}\gamma_t} \tag{76}$$

These equations are of the form:

$$\Delta_{\mu'}(x'|x) \quad \leq \quad \mu(x'|x)\max_t\left(\frac{\Delta_{\xi_t}}{\xi_t} + \frac{\Delta_{\gamma_t}}{\gamma_t}\right) \tag{77}$$

and

$$\Delta_{\mu'}(x'|x) \quad \geq \quad -\mu(x'|x) \max_t \left(\frac{\Delta_{\xi_t}}{\xi_t} + \frac{\Delta_{\gamma_t}}{\gamma_t} \right) \tag{78}$$

which implies that $\Delta^r_{\mu'} \leq (\Delta^r_\xi + \Delta^r_\gamma)$. The lemma follows from the bounds for Δ^r_γ and Δ^r_ξ stated in Lemma 5 and 6. □

Lemma 9. $\Delta^r_{\nu'} = 4(\Delta^r_\pi + (T-1)(\Delta^r_\mu + \Delta^r_\nu))$ *to first order.*

The proof is analogous to the proof of Lemma 8.

Theorem 5. *The error introduced in a round of Baum-Welch with some initial* $\Delta^r_\pi, \Delta^r_\mu$, *and* Δ^r_ν *to first order is* $\Delta^r_{\pi'} = 2((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi)$, $\Delta^r_{\mu'} = 4((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi)$, *and* $\Delta^r_{\nu'} = 4((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi)$.

Proof. Theorem 5 follows directly from Lemmas 7 to 9. □

6.5 Implications for MCHMMs

The analysis of error propagation for GHMMs implies that as $N \to \infty$ and the relative errors approach 0 the error of a step in the generalized Baum-Welch algorithm tends towards the first order errors: $\forall \varepsilon > 0 : \exists N : \Delta^r_{\pi'} = 2((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi) + \varepsilon$, $\Delta^r_{\mu'} = 4((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi) + \varepsilon$, and $\Delta^r_{\nu'} = 4((T-1)(\Delta^r_\nu + \Delta^r_\mu) + \Delta^r_\pi) + \varepsilon$. These first order errors also converge to 0.

$$\lim_{N\to\infty} \Delta^r_{\nu'} \quad = \quad 0 \tag{79}$$

$$\lim_{N\to\infty} \Delta^r_{\mu'} \quad = \quad 0 \tag{80}$$

$$\lim_{N\to\infty} \Delta^r_{\pi'} \quad = \quad 0 \tag{81}$$

We are now ready to present the central result of this section, a proof of convergence for MCHMMs.

Proof of Theorem 4 (MCHMM Convergence Theorem). According to Theorem 5, the first order relative errors of ${\pi'}^{(n+1)}$, ${\mu'}^{(n+1)}$, and ${\nu'}^{(n+1)}$ are linear in the relative errors of ${\pi'}^{(n)}$, ${\mu'}^{(n)}$, and ${\nu'}^{(n)}$. By assumption, in the limit as $N \to \infty$ these errors go to 0 with probability 1, implying the theorem. □

MCHMMs converge to a local maximum, under the additional assumption that the relative error of all distributions converge to 0 as $N \to \infty$. Notice that the ρ parameter was not used here—the ρ parameter selects between various models, while we only proved convergence within one model. However, since $\rho \longrightarrow 0$, its effect vanishes over time.

The MCHMM Convergence Theorem establishes the soundness of our algorithm, and shows that MCHMMs can indeed be applied for learning a large range of non-parametric statistical models with real-valued state and observation spaces. Empirical results using finite sample sets, described in the next section, suggest that the algorithm is stable even for small sample set sizes, and indeed maximizes data likelihood in a Monte Carlo-fashion.

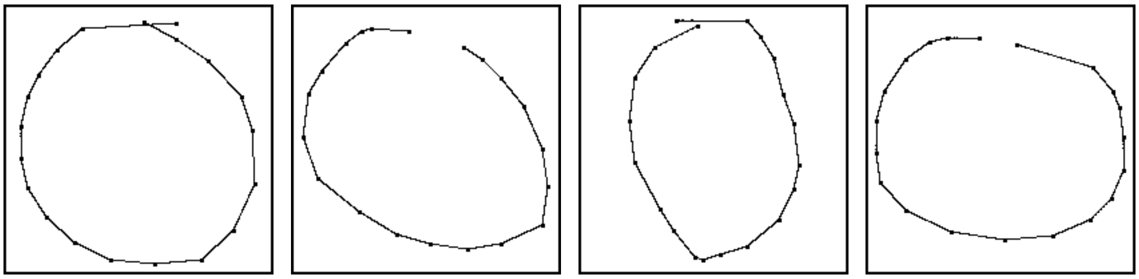

Figure 4: Examples of gestures. The first two gestures were drawn counterclockwise; whereas the other two were drawn clockwise. The MCHMM learning task is to differentiate them.

7 Experimental Results

We have applied MCHMMs to two problems, an artificial one which was chosen for demonstration purposes, and a more difficult real-world gesture recognition problem. The experiments address the following questions:

- Do MCHMMs converge empirically? If so, how good are the resulting MCHMMs?
- What accuracies can be obtained when using MCHMMs for discrimination?
- How does the sample set size affect computational and accuracy trade-offs?

The first data set, called the *noisy oscillation dataset*, consists of multiple sequences that basically oscillate around two points. Observations are 10-dimensional and governed by

$$O_t \quad = \quad \begin{cases} (0.25 + \varepsilon_{t,1}, 0.25 + \varepsilon_{t,2}, 0.25 + \varepsilon_{t,3}, \ldots, 0.25 + \varepsilon_{t,10}) & \text{if } t \text{ odd} \\ (0.75 + \varepsilon_{t,1}, 0.75 + \varepsilon_{t,2}, 0.75 + \varepsilon_{t,3}, \ldots, 0.75 + \varepsilon_{t,10}) & \text{if } t \text{ even} \end{cases} \tag{82}$$

where $\varepsilon_{t,i}$ (with $1 \leq t \leq 20$ and $1 \leq i \leq 10$) are independent and identically distributed noise variables with zero-centered triangular distribution over $[-0.15; 0.15]$. To test the discriminatory accuracy of the learned model, we also generated a second, similar data set:

$$O_t \quad = \quad \begin{cases} (0.25 + \varepsilon_{t,1}, 0.75 + \varepsilon_{t,2}, 0.25 + \varepsilon_{t,3}, \ldots, 0.75 + \varepsilon_{t,10}) & \text{if } t \text{ odd} \\ (0.75 + \varepsilon_{t,1}, 0.25 + \varepsilon_{t,25}, 0.75 + \varepsilon_{t,3} \ldots, 0.25 + \varepsilon_{t,10}) & \text{if } t \text{ even} \end{cases} \tag{83}$$

using new, independent noise variables. Notice that despite the fact that there is a good amount of noise in the data, these data sets are relatively easy to discriminate, since their observations fall into different regions in the $\Re^{10}$.

The second data set consisted of a collection of hand-drawn gestures, represented in $\Re^2$. Figure 4 shows examples. Once drawn, all gestures in our data base look quite similar. However, some of the gestures were drawn clockwise, whereas others were drawn counterclockwise. Here we are interested in discriminating clockwise from counterclockwise gestures. Notice that this problem is more difficult then the artificial one, as the observations alone (stripped of their temporal order) are insufficient for discrimination; instead, the MCHMM has to learn a meaningful model of the internal state.

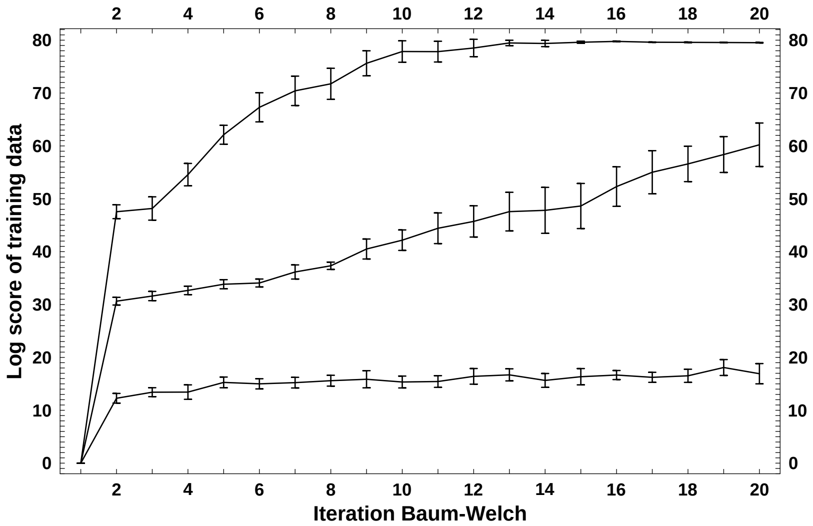

Figure 5: Log score of training data as a function of the iteration of EM, for the synthetic data set. Top curve: 1000 samples for all densities. Middle curve: 1000 samples for μ and ν, 100 samples for α, β, γ, and π. Bottom curve: 100 samples for μ and ν, 10 samples for α, β, γ, and π. These graphs illustrate that MCHMMs tend to maximize the data likelihood, even in the finite case. Each result is averaged over 10 independent experiments; 95% confidence bars are also shown.

Figures 5 and 6 show results obtained for the first dataset. Shown in both figures are curves that characterize the "log score" as a function of the number of iterations. This score is the real-valued analogue to the likelihood; however, since densities can be larger than one, the score can also be larger than one, and hence its logarithm is not bounded above.

The curves in Figure 5 show average log score for the *training set* as a function of the number of iterations, averaged over 10 independent experiments (each curve). The different curves were obtained using different numbers of samples which, contrary to the theoretical results, were kept constant throughout the experiments:

Sample set size N for...	μ, ν	$\alpha, \beta, \gamma, \pi$
top curve:	1,000	1,000
middle curve:	1,000	100
bottom curve:	100	10

Figure 5 also provides 95% confidence bars for these results. The difference in absolute levels to which the scores appear to converge stem from the fact that the maximum tree depth, and hence the maximum of the tree density $\hat{f}$, is a function of the sample set size; hence, if the number of samples is small, the tree will remain shallow. The key observation, however, is their monotonicity. These curves demonstrate that each iteration in MCHMM indeed increases the data likelihood (or leaves it unchanged), even if only finitely many samples are used. This finding, which we also consistently observed for other data sets and parameter settings, illustrate that MCHMMs are well-behaved in practice, that is, the finiteness of the sample set appears not to cause catastrophic problems.

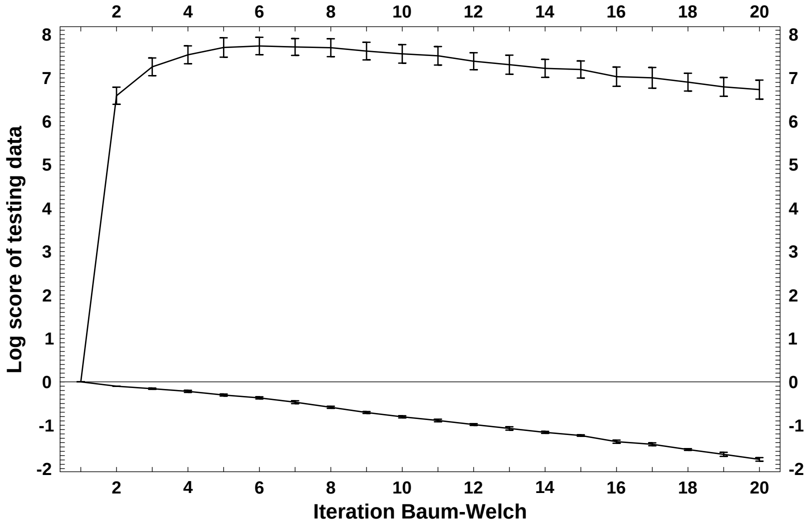

Figure 6: Log score of testing data as a function of the iteration of EM, for the synthetic data set. The top curve shows the log score of data generated from the model used in training, whereas the bottom curve shows the log score for data generated from a different model. 95% confidence bars are also shown.

Figure 6 shows results for independent testing data. The upper curve in Figure 6 depicts the log score for set of random sequence that are generated from the same model as the training sequences. The bottom curve depicts the log score for independently generated sequences using the other model, for which the MCHMM was not trained. In both cases, only a single data set (of length 20) was used for training, and testing was performed using 20 data sets generated independently. $N = 100$ samples were used throughout for all densities α, β, γ, and π, and $N = 1,000$ samples were used for μ and ν, to account for their higher dimensionality. The initial shrinkage factor was $\rho = 1$, which was annealed down at the rate $\bar{\rho} = 0.9$. In all experiments, the dimension of the state space was $k = 2$. The results obtained for lower-dimensional observation spaces, which are not shown here, were even better. In all cases, the score clearly reflected that the MCHMM had learned a good model. The classification accuracy for test sets was consistently 100%.

The result in Figure 6 illustrate the discriminative power of MCHMM for this data set. It also demonstrates the effect of annealing: The testing data likelihood is maximal at iteration 6 (thus the optimal ρ is $\hat{\rho} = 0.53$), beyond which the MCHMM starts overfitting the data. In our experiments, the optimal stopping point was consistently found within ± 1 step, using a single, independently generated sequence for cross-validation. In all our experiments, the MCHMM consistently discriminated the two different classes without error, regardless of the settings of the individual parameters (we did not run experiments with $N < 10$). The bars in Figure 6 are confidence bars at the 95% confidence level.

Figure 7 shows the corresponding result for the more difficult gesture recognition problem. These results were obtained using a single gesture for training only, and using 50 gestures for testing. The top curve depicts the log score for gestures drawn in the same direction as the training gesture, whereas the bottom curve shows the log scores for gestures drawn in opposite direction. As easily noticed in Figure 7, the difference between classes is smaller than in Figure 6—which

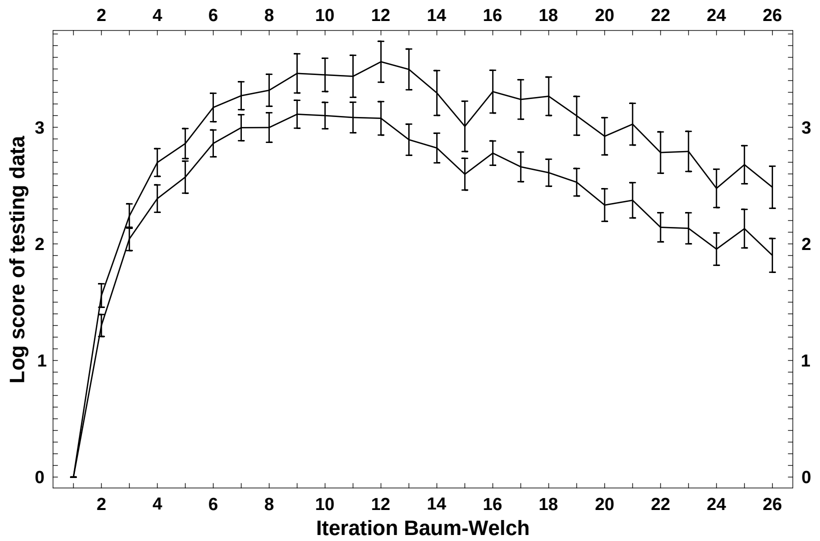

Figure 7: Log score for the gesture data base, obtained for independent testing data. The top curve shows the log score of gestures drawn in same same way as the training gesture, whereas the bottom curve shows the log score of gestures drawn in opposite direction.

comes at no surprise—, yet the score of the "correct" class is still a factor of 4 to 5 larger than that of the opposite class. Figure 7 also shows the effect of annealing. The best shrinkage value is obtained after 12 iterations, where $\rho = 0.28$. As in the previous case, cross-validation using a single gesture performs well. On average, the classification accuracy when using cross-validation for early stopping is 86.0%. This rate is remarkably high, given that only a single gesture per class was used for training.

A key advantage of MCHMM over HMMs lies in the ease of trading off computational complexity and accuracy. In conventional HMM, the computational complexity is fixed for any given (trained) model. In contrast, MCHMM permit variable sampling set sizes during run-time, after the model is trained. This leads to a nice *any-time* algorithm [9, 58]. Once trained, a MCHMM can control its computational complexity on-line by dynamically adjusting the sample size, depending on the available resources. The any-time property is important for many real-world applications, where computation resources are bounded. To the best of our knowledge, the any-time nature is a unique property of MCHMMs, which is not shared by previous HMM algorithms.

Figure 8 illustrates the trade-off between computation and accuracy empirically for the gesture data base. Shown there is the trade-off between the number of samples and the likelihood of the testing data. Notice that the horizontal axis is logarithmic. All of these points are generated using a model λ obtained using early stopping. The sample set size was generated *after* training, to investigate the effect of computational limitations on the on-line performance of the MCHMM. As in Figure 7, the top curve in Figure 8 depicts the log score of gestures drawn in the same direction as the training data, whereas the bottom curve shows the log score of gestures drawn in opposite direction.

The result in Figure 8 illustrates that the score (and hence the accuracy) of both data sets increases with the sample set size. This is not surprising, as the accuracy of the approximations

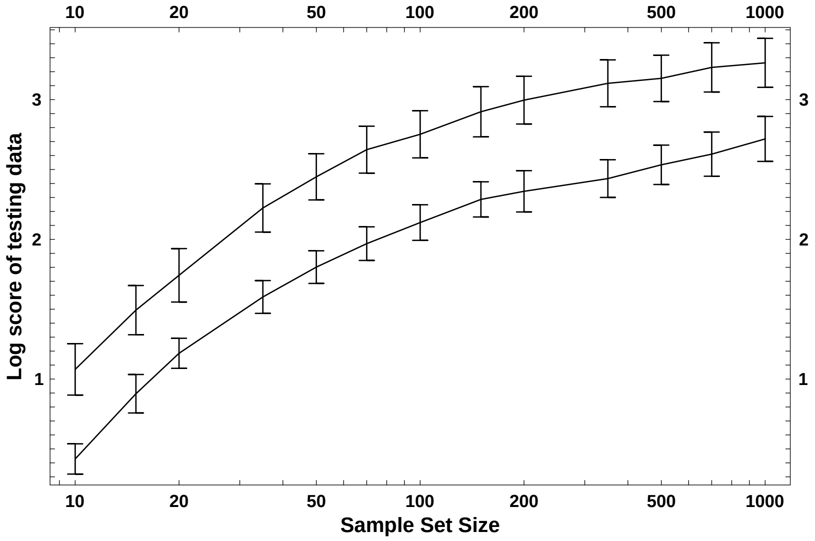

Figure 8: Trade-off between sample set size and log score, for the gesture data set. The more samples are available, the higher the score, and hence the more accurate the results. However, even extremely small sample sets yield reasonable discrimination.

increases with the sample set size. In addition, Figure 8 suggests that the distance between clockwise and counterclockwise gestures is approximately constant in log score space (hence it grows in score). This illustrates that better results are achieved with larger sample sets. In our experiments, however, even small sample sets yielded good discrimination (after training with larger sample sets). For example, we observed on average 16% classification error with $N = 10$ samples. The processing of such sample sets is several orders of magnitude faster than the hand motion involved in gesture generation. This makes MCHMMs extremely fast for on-line tracking and discrimination in this domain.

Following suggestions by Koller[2], we evaluated the utility of *smoothing* in the context of MCHMMs [13, 24]. Smoothing transforms those sample sets that represent posteriors over states (α and β) into more compact representations such as density trees, which are then used for likelihood-weighted sampling instead of the original samples. To apply smoothing to MCHMMs, the basic learning algorithm (see Table 1) was modified as follows:

Step 3a: Generate N samples $\langle x', p_{x'} \rangle$ from the *tree* representing α_{t-1} using likelihood-weighted sampling.

Step 4a: Generate N samples $\langle x', p_{x'} \rangle$ from the *tree* representing β_{t+1} using likelihood-weighted sampling.

Notice that this is a minor modification of the basic algorithm, as trees are readily computed for the various densities α and β; but now they are also used for likelihood-weighted sampling. Obviously, smoothing reduces the variance of the various α and β at the cost of increasing bias. It has been suggested that smoothing with density trees can further improve performance [24].

[2] personal communication

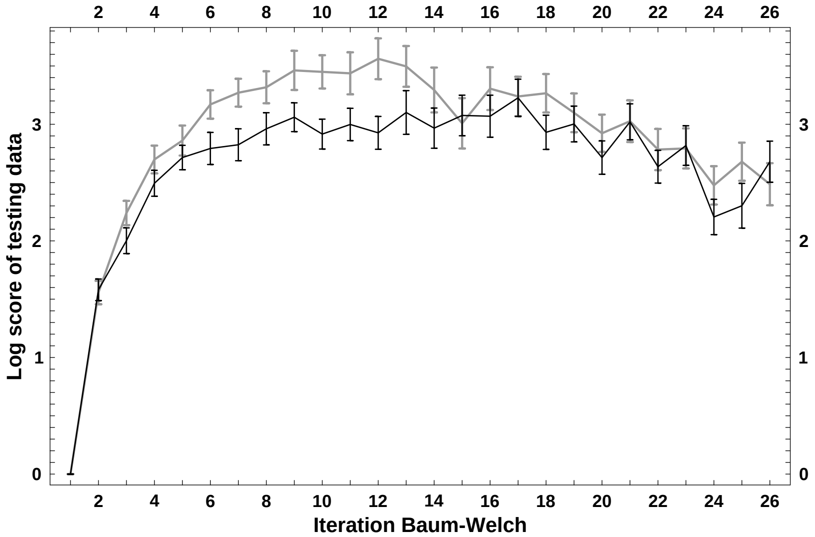

Figure 9: MCHMMs with (black curve) and without (grey curve) density tree smoothing of α and β, under otherwise equal conditions. Each curve plots the log score of the testing data as a function of the number of Baum-Welch iterations. The grey curve is equivalent to the top curve in Figure 7. These results illustrate the negative effect of smoothing on log score of the testing data.

Unfortunately, in our experiments we were unable to confirm the utility of smoothing in the context of MCHMMs. For the artificial data set, the generalization performance of MCHMMs with smoothing was indistinguishable from that obtained without smoothing, both in terms of log score of the testing data and the class discrimination accuracy. Smoothing actually worsened the performance for the gesture data set. Figure 9 shows the log score of testing data using smoothing (black curve), and compares it with the corresponding log scores obtained without smoothing (gray curve, copied from Figure 7). Smoothing was found to reduce the log score of the testing data; thus, MCHMMs trained with smoothing are less capable of "explaining" new examples. This is not surprising, as smoothing is not an information loss-free operation, and it is not clear that the inductive bias provided by a tree representation is beneficial. The reduction of the log score led to a slight reduction of the discrimination accuracy. Using cross-validation, the average classification accuracy for MCHMMs with smoothing was 82.0%, which is 4.0% smaller than the accuracy obtained without smoothing.

8 Related Work

Hidden Markov Models [2, 43] have been successfully applied to a huge range of applications requiring temporal signal processing. As indicated in the introduction of this paper, most state-of-the-art speech recognition systems rely on HMMs (see, for example, the various papers in [54]). Recently, extensions of HMMs have been successfully applied in the context of mobile robotics [23, 48, 52], where they have led to improved solutions of the *concurrent mapping and localization* problem, which is generally acknowledged as one of the most difficult problems in robotics [6, 25].

These are just two examples of successful application areas of HMM; the number of successful applications is numerous.

Most HMM algorithms differ from MCHMMs in that they only apply to discrete state and observation spaces. While the majority of HMM approaches represent states individually (in a flat way), some more recent approaches have extended HMMs to loosely coupled factorial representations [15, 46], which represent state by features but are nevertheless discrete. Others have successfully proposed HMM models that can cope with real-valued observation spaces [29, 19]. Our approach generalizes HMMs to continuous state and observation spaces with a large, non-parametric range of state transition and observation densities. It also differs from previous HMM approaches in that it provides a mechanisms for complexity control (regularization); in previous approaches, the complexity of the internal state space had to be calibrated carefully by hand. Finally, MCHMMs provide a mechanisms to trade off computational requirements and accuracy in an any-time fashion, which differs from previous approaches whose computational requirements were not adjustable during run-time [9, 58]. We envision that all of these advances are important for a broad range of practical problems.

Sampling techniques, one of the two methods used for density approximation in MCHMMs, have recently gained popularity in the applied statistics and AI literature. Various researchers have applied sampling methods in the context of state estimation in dynamical systems [3, 13, 22, 30, 41] and learning [7]. A nice introduction into the use of sampling for density approximation can be found in [36]. Our approach to state estimation (computation of the αs) is essentially equivalent to the *condensation algorithm* proposed by Isard and Blake [18] and the *Markov localization algorithm* proposed by Dellaert and colleagues [11, 10], both of which are basically versions of the well-known *sampling/importance resampling* (SIR) algorithm [44, 49]. Similar approaches are known as *particle filters* [41], *bootstrap* [14], and *survival of the fittest* [22]. All these approaches are concerned with state estimation in an HMM-like or Kalman filter-like fashion. Thus, they rely on the a priori availability of μ and ν, which are learned from data by the MCHMM algorithm proposed here. To our knowledge, the application of sampling-based methods to learning non-parametric state transition models and observation models in HMMs is new, as is our proposal for the integration of a forward (α) and backward (β) phase using trees. The theoretical results in this paper demonstrate that our approach can be applied to a large class of problems, assuming that sufficiently many samples are used. Trees have frequently been used for density approximation, most recently in [24, 35, 38, 39]. However, we are not aware of a proof of asymptotic consistency for density trees, although we suspect that such a result exists.

9 Conclusion

We have presented a new algorithm for hidden Markov models, called Monte Carlo Hidden Markov Models (MCHMM). MCHMMs extend HMMs to real-valued state and observation spaces. They represent all densities by samples, which are transformed into probability density functions using density trees. Both representations were proven to be asymptotically consistent under minimal assumptions on the nature of the density that is being approximated. Because the continuous state spaces are rich enough to fit (and over-fit) arbitrary data sets, our approach uses shrinkage to reduce its complexity. The shrinkage parameter is gradually annealed down over time, and cross-validation (early stopping) is used to select the best model complexity. The pervasive use of Monte Carlo sampling led to the design of an any-time implementation, capable of dynamically trading off computational complexity and the accuracy of the results. Consequently, MCHMMs are well-suited for time-critical applications that are frequently encountered in the real world.

We have proved the asymptotic consistency of MCHMMs for a large class of probability density functions. Empirical results, carried out in an artificial domain and a more challenging gesture recognition domain, demonstrate the our approach generalizes well even when trained with extremely scarce data. Additional experiments characterize the natural trade-off between sample set size and accuracy, illustrating that good results may be achieved even from extremely small sample sets.

While the theoretical results of this paper hold only in the limit, actual applications have to live with finite sample sets. Thus, an interesting and open question is under what conditions MCHMM with finite sample sets converges, and what error to expect. While principal error bounds of this type are relatively easy to obtain in static approximation setting, the recursive nature of Baum-Welch makes it difficult to obtain bounds for MCHMM. The empirical results described here, however, suggest that MCHMM is a robust method that works well in practice even with relatively small sample sets.

We conjecture that MCHMMs are better suited for many real-world application domains such as speech and robotics applications than conventional HMMs, for primarily three reasons: their support of continuous representations of observations and state, their built-in mechanisms for model selection, which reduces the burden of picking the "right" model (e.g., number of states in conventional HMMs), and, finally, their support of any-time computation, which makes them extremely compliant in time-critical applications with bounded computational resources.

Acknowledgement

The authors gratefully acknowledge inspiring discussions with Frank Dellaert, Nir Friedman, Dieter Fox, Daphne Koller, and Larry Wasserman.

References

[1] P. Baldi and Y. Chauvin. Smooth online learning algorithms for hidden Markov models. *Neural Computation*, 6:307–318, 1994.

[2] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[3] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 33–42, Madison, WI, 1998.

[4] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. Technical Report CMU-CS-98-139, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1998.

[5] E. Charniak. *Statistical Language Learning*. MIT Press, Cambridge, Massachusetts, 1993.

[6] I.J. Cox. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[7] J.F.G. de Freitas, S.E. Johnson, M. Niranjan, and A.H. Gee. Global optimisation of neural network models via sequential sampling-importance resampling. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, 1998.

[8] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[9] T. L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceeding of Seventh National Conference on Artificial Intelligence AAAI-92*, pages 49–54, Menlo Park, CA, 1988. AAAI, AAAI Press/The MIT Press.

[10] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. Submitted for publication, 1998.

[11] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. Submitted for publication, 1998.

[12] A.P. Dempster, A.N. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[13] A Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Cambridge University, Department of Engineering, Cambridge, UK, 1998.

[14] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[15] Z. Ghahramani and M.I. Jordan. Factorial hidden markov models. In *Advances in Neural Information Processing Systems 8 (NIPS)*, pages 472–478, Cambridge, MA, 1996. MIT Press.

[16] B. Hannaford and P. Lee. Hidden markov model analysis of force torque information in telemanipulation. *International Journal of Robotics Research*, 10(5):528–539, 1991.

[17] J. Henderson, S. Salzberg, and K. Fasman. Finding genes in human DNA with a hidden Markov model. In *Proceedings of Intelligent Systems for Molecular Biology '96*, Washington University, St. Louis, MI, 1996.

[18] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *International Journal of Computer Vision*, in press 1998.

[19] B.H. Juang. Maximum likelihood estimation for mixture multivariate stochastic observations of markov chains. *AT&T Technical Journal*, 64(6), 1985.

[20] B.H. Juang, S.E. Levinson, and M.M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of markov chains. *IEEE Transaction and Information Theory*, 32(2), 1986.

[21] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[22] K. Kanazawa, D. Koller, and S.J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the 11th Annual Conference on Uncertainty in AI*, Montreal, Canada, 1995.

[23] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[24] D. Koller and R. Fratkina. Using learning for approximation in stochastic processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1998.

[25] D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors. *AI-based Mobile Robots: Case studies of successful robot systems*, Cambridge, MA, 1998. MIT Press.

[26] A. Krogh, I.S. Mian, and D. Haussler. A hidden markov model that finds genes in e. coli dna. *Nucleic Acids Research*, 22:476–8–4778, 1994.

[27] K.-F. Lee. *Automatic Speech Recognition: The Development of the SPHINX System*. Kluwer Publishers, Boston, MA, 1989.

[28] K.-F. Lee. Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990. Also appeared in the IEEE Transactions on Acoustics, Speech, and Signal Processing.

[29] L.A. Liporace. Maximum likelihood estimation for multivariate observations of markov sources. *IEEE Transactions in Information Theory*, 28(5), 1982.

[30] J. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93, 1998.

[31] P. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1*. Academic Press, Inc, 1979.

[32] A. McCallum, R. Rosenfeld, T. Mitchell, and A.Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1998.

[33] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, New York, 1997.

[34] M. Meilă and M.I. Jordan. Learning fine motion by markov mixture of experts. In *Advances in Neural Information Processing Systems 8 (NIPS)*, pages 1003–1009, Cambridge, MA, 1996. MIT Press.

[35] A.W. Moore, J. Schneider, and K. Deng. Efficient locally weighted polynomial regression predictions. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann Publishers, 1997.

[36] R.M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, Toronto, CA, 1993.

[37] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Press, 1998.

[38] S. M. Omohundro. Efficient Algorithms with Neural Network Behavior. *Journal of Complex Systems*, 1(2):273–347, 1987.

[39] S. M. Omohundro. Bumptrees for Efficient Function, Constraint, and Classification Learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991.

[40] A.G. Pedersen, P. Baldi, S. Brunak, and Y. Chauvin. Characterization of prokaryotic and eukaryotic promoters using hidden markov models. In *Proceedings of Intelligent Systems for Molecular Biology '96*, Washington University, St. Louis, MI, 1996.

[41] M. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association*, 1999. Forthcoming.

[42] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. IEEE, 1989. IEEE Log Number 8825949.

[43] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. In *IEEE ASSP Magazine*, 1986.

[44] D.B. Rubin. Using the SIR algorithm to simulate posterior distributions. In M.H. Bernardo, K.M. an DeGroot, D.V. Lindley, and A.F.M. Smith, editors, *Bayesian Statistics 3*. Oxford University Press, Oxford, UK, 1988.

[45] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[46] L.K. Saul and M.I. Jordan. Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems 8 (NIPS)*, pages 486–492, Cambridge, MA, 1996. MIT Press.

[47] H. Shatkay. *Learning Models for Robot Navigation*. PhD thesis, Computer Science Department, Brown University, Providence, RI, 1998.

[48] H Shatkay and L. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of IJCAI-97*. IJCAI, Inc., 1997.

[49] A.F.M. Smith and A.E. Gelfand. Bayesian statistics without tears: a sampling-resampling perspective. *American Statistician*, 46:84–88, 1992.

[50] C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability 1*, pages 197–206. University of California Press, 1955.

[51] M.A. Tanner. *Tools for Statistical Inference*. Springer Verlag, New York, 1993. 2nd edition.

[52] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998. also appeared in Autonomous Robots 5, 253–271.

[53] V. Vapnik. *Estimations of dependences based on statistical data*. Springer Publisher, 1982.

[54] A. Waibel and K.-F. Lee, editors. *Readings in Speech Recognition*. Morgan Kaufmann Publishers, San Mateo, Californien, 1990.

[55] A. S. Weigend and N. A. Gershenfeld, editors. *Time Series Prediction*. Addison-Wesley, Reading, MA, September 1993.

[56] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina, Department of Computer Science, 1995.

[57] G. Winkler. *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods*. Springer Verlag, Berlin, 1995.

[58] S. Zilberstein and S. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, Dordrecht, 1995.