Markovian Models for Sequential Data

Yoshua Bengio †
Dept. Informatique et Recherche Opérationnelle
Université de Montréal, Montreal, Qc H3C-3J7, Canada

Abstract

Hidden Markov Models (HMMs) are statistical models of sequential data that have been used successfully in many machine learning applications, especially for speech recognition. Furthermore, in the last few years, many new and promising probabilistic models related to HMMs have been proposed. We first summarize the basics of HMMs, and then review several recent related learning algorithms and extensions of HMMs, including in particular hybrids of HMMs with artificial neural networks, Input-Output HMMs (which are conditional HMMs using neural networks to compute probabilities), weighted transducers, variable-length Markov models and Markov switching state-space models. Finally, we discuss some of the challenges of future research in this very active area.

1 Introduction

Hidden Markov Models (HMMs) are statistical models of sequential data that have been used successfully in many applications in artificial intelligence, pattern recognition, speech recognition, and modeling of biological sequences. The focus of this paper is on learning algorithms which have been developed for HMMs and many related models, such as hybrids of HMMs with artificial neural networks [1, 2, 3], Input-Output HMMs [4, 5, 6, 7], weighted transducers [8, 9, 10, 11], variable-length Markov models [12, 13], Markov switching models [14] and switching state-space models [15, 16]. Of course, there is a lot more litterature on HMMs and their applications than can be covered here, but this survey wants to be representative of the issues addressed here, mainly concerning learning algorithms and extensions of HMMs and related models.

Note that what we call learning here is also called parameter estimation in statistics and system identification in control and engineering. The models and the probability distributions that we talk about in this paper are not assumed to represent necessarily the true relations between the variables of interest. Instead, they are viewed as *tools for taking decisions* about the data, in particular about new data. It is also important to distinguish between the model classes, which correspond to assumptions about a good set of solutions, and the training algorithms, such as the EM (Expectation-Maximization) algorithm, which are used to pick one solution in such as set to optimize a data-dependent criterion.

The models discussed here, which we call **Markovian models**, can be applied to sequential data which have a certain property described here. First let us remind the reader that the joint probability distribution[1] of a sequence of observations $y_1^T = \{y_1, y_2, \ldots, y_T\}$ can always be factored as

$$P(y_1^T) = P(y_1) \prod_{t=2}^{T} P(y_t | y_1^{t-1}).$$

[0] † This work is supported by the National Science and Engineering Research Council of Canada and the Institute of Robotics and Intelligent Systems. Some of the work was performed while Yoshua Bengio was at AT&T Laboratories . Updates, corrections, and comments should be sent to Yoshua Bengio at bengioy@iro.umontreal.ca.

Neural Computing Surveys 2, 129–162, 1999, *http://www.icsi.berkeley.edu/~jagota/NCS*

[1] In this paper, we will use the notation $P(x)$ to mean the probability $P(X = x)$ that random variable X takes the value x, unless the context would make the notation ambiguous.

It would be intractable in general to model sequential data in which the conditional distribution $P(y_t|y_1^{t-1})$ of an observed variable y_t at time t depends on all the details of the previous values y_1^{t-1}. However, the models discussed in this paper share the property that they assume that the past sequence can be summarized concisely, often using an unobserved random variable called a **state variable**, which carries all the information from y_1^{t-1} that is useful to describe the distribution of the next observation y_t.

The most common of these models are the HMMs, which are best known for their contribution to advances in automatic speech recognition in the last two decades. A good tutorial on HMMs in the context of speech recognition is [17]. See also this book [3], which focusses on the applications to bioinformatics. Algorithms for estimating the parameters of HMMs have been developed in the 60's and 70's [18, 19, 20]. The application of HMMs to speech was independently proposed by [21] and [22], and popularized by [23], [24], and [17]. An early review of alternative methods based on HMMs or related to HMMs, also for speech recognition, can be found in the collection of papers [25]. Recently, HMMs have been applied to a variety of applications outside of speech recognition, such as handwriting recognition [26, 27, 28, 29, 30, 31, 32], pattern recognition in molecular biology [33, 34, 35, 36, 3], and fault-detection [37]. The variants and extensions of HMMs discussed here also include language models [38, 39, 13], econometrics [14, 15, 40], time series [41], and signal processing. An analysis of the sample and computational complexity of approximating a distribution using an HMM or a probabilistic automaton has been done [42] using tools from the PAC-learning paradygm [43]. See also [44] for an analysis of the case of hidden Markov chains with deterministic emissions, which shows that some classes of Markovian learning problems are hard while others are polynomial in the number of samples required. In the case of probabilistic finite state automata, [45] show an algorithm that has polynomial time complexity in the sample size for learning a subclass of acyclic probabilistic finite state automata.

The learning problem for the type of algorithms discussed here can be framed as follows. Given a training set $\mathcal{D} = \{d_1, \ldots, d_N\}$ of N sequences of data and a criterion C for the quality of a model on a set of data (mapping $\mathcal{D}$ and a model to a real-valued scalar), choose a model from a certain set of models, in such a way as to maximize (or minimize) the expected value of this criterion on new data (assumed to be sampled from the same unknown distribution from which the training data was sampled). For a general mathematical analysis of the learning theory behind learning algorithms such as those discussed here, see for example [46]. In some applications there is only one sequence of observations $d = y_1^T = \{y_1, y_2, \ldots, y_T\}$, and the new data is simply a continuation of the training data (e.g., time-series prediction, econometry). In other applications there is a very large number of training sequences of different lengths, (e.g., thousands or tens of thousands of sequences, as in speech recognition databases). In some applications, the objective is to model the distribution of a sequence variables, e.g., $P(y_1^T)$. In other applications, the data consists of sequences of "output" variables y_1^T given "input" variables x_1^L, and the objective is to model the conditional distribution $P(y_1^T|x_1^L)$. In some of these applications the input and output sequences do not have the same length. For example, in speech recognition, we are interested in the distribution of word sequences given an acoustic sequence.

The next two sections of this paper review the basic elements of traditional HMMs (section 2) and their application to speech recognition (section 3). The remaining sections describe extensions of HMMs and Markovian models related to HMMs, i.e., hybrids with Artificial Neural Networks in section 4, Input-Output HMMs in section 5 (including Markov switching models in section 5.1, asynchronous Input-Output HMMs in section 5.3), generalizations of HMMs called weighted transducers in section 6 (useful to combine many Markovian models), and finally, state space models (Markovian models with continuous state) in section 7.

2 Hidden Markov Models

In this section we remind the reader of the basic definition of an HMM in a tutorial-like way. We formalize the assumptions that are made, and describe the basic elements of algorithms for HMMs. The algorithms to estimate the parameters for HMMs will be discussed in more details in section 5.2 after we have generalized HMMs to Input-Output or conditional HMMs.

A Markov model [47] of order k is a probability distribution over a sequence of variables $q_1^t = \{q_1, q_2, \ldots, q_t\}$

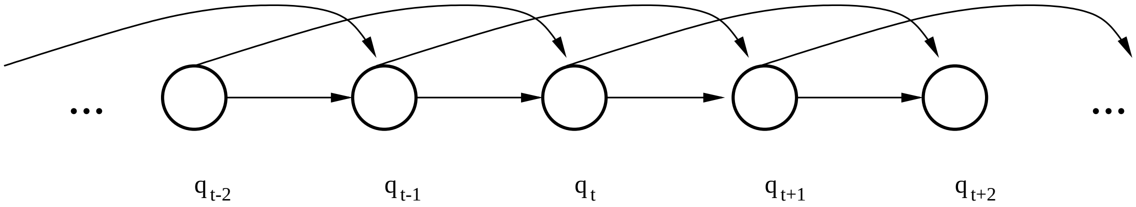

Figure 1: Bayesian network representing the independence assumptions of a Markov model of order 2: $P(q_t|q_1^{t-1}) = P(q_t|q_{t-1}, q_{t-2})$, where $q_1^{t-1} = \{q_1, q_2, \ldots, q_{t-1}\}$
.

with the following conditional independence property:

$$P(q_t|q_1^{t-1}) = P(q_t|q_{t-k}^{t-1}).$$

Since q_{t-k}^{t-1} summarizes all the relevant past information, q_t is generally called a **state variable**. Because of the above conditional independence property, the joint distribution of a whole sequence can be decomposed into the product

$$P(q_1^T) = P(q_1^k) \prod_{t=k+1}^{T} P(q_t|q_{t-k}^{t-1}).$$

The special case of a Markov model **of order 1** is the one found in most of the models described in this paper. In this case, the distribution is even simpler,

$$P(q_1^T) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}),$$

and it is completely specified by the so-called **initial state probabilities** $P(q_1)$ and **transition probabilities** $P(q_t|q_{t-1})$.

To illustrate the concept of a Markov model of order k, let us consider the following simple example. A robot can be at each discrete time step in one of n discrete positions on a grid: these are the different states, or values of the state variable at time t. At each time step, it jumps from one position to another position or stays at the same position, with probabilities that depend on the k last positions in which it was. For example, if $k = 1$, the probability distribution for the next position only depends on its current position. A plausible dependency is that it may have a high probability of jumping to a nearby position (in the grid) or to stay where it is, and a low probability of jumping to a remote position. The model is completely characterized by these conditional probabilities, and it can be estimated by simply observing a long trajectory of the robot and counting how many times it jumps from one position to another.

A **Bayesian network** [48] is a graphical representation of conditional independencies between random variables. A Bayesian network for a Markov model of order 2 is shown in Figure 1. The figure shows a directed acyclic graph (DAG), in which each node corresponds to a random variable. An edge from variable A to variable B implies a causal and direct influence of A on B. The absence of an edge between A and B implies a conditional independence between variables A and B, even though there may exist a path between A and B. Conditioning on intermediate variables on paths between A and B can make A and B independent. More specifically, the joint probability distribution of the set of random variables $V = \{A_1, \ldots, A_n\}$ represented in the graph (with arbitrary connectivity) is given by the product

$$P(A_1, \ldots, A_n) = \prod_{i=1}^{n} P(A_i|\texttt{parents}(A_i))$$

where $\texttt{parents}(A) = \{B \in V|$ there is an edge from B to $A\}$. See [48, 49, 50, 51] for more formal definitions, and pointers to related literature on graphical probabilistic models and inference algorithms for them. See [52, 53] for relations between HMMs and other graphical models such as Markov random fields and Kalman filters, and see [54] for an application of these ideas to Turbo-decoding. Other formulations of HMMs as graphical models have been proposed and applied successfully to speech recognition [55, 56]. As also exploited in the review presented here, the graphical model formulation of [55, 56] lends itself naturally to extensions such as the introduction of context variables or factoring of the state variable. Note that almost all the probabilistic models described in this paper can be cast in the framework of these Bayesian networks.

In many Markovian models, the transition probabilities are assumed to be homogeneous, i.e., the same for all time steps. For example, for Markov models of order 1, $P(q_t|q_{t-1}) = P(q_2|q_1)$, $\forall t$. With homogeneous models, the number of parameters is much reduced, and the model can be trained on sequences of certain lengths and generalize to sequences of different lengths. It makes sense to use such models on sequential data which shows temporal translation invariance. Other models, such as Input-Output (or conditional) HMMs (section 5), are inhomogeneous: different transition probabilities are used at different time steps. However, since the transition probabilities are not directly the parameters of the model but are instead obtained as a parameterized function of the previous state and other conditioning variables, the same advantages stated above apply, with more ability to deal with some of the changes in dynamics observed in different parts of the sequences.

In most applications, the state variable is discrete and the conditional distribution of the state variable at time t is given by a multinomial distribution. An exception to this approach is briefly discussed in section 7, with "continuous-state HMMs" (more generally called state-space models).

2.1 Hidden State

One problem with Markov models of order k is that they quickly become intractable for large k. For example, for a multinomial state variable $q_t \in \{1, \ldots, n\}$, the number of required parameters for representing the transition probabilities is $O(n^{k+1})$. This necessarily restricts one to using a small value of k. However, most observed sequential data of interest do not satisfy the Markov assumption for k small. As stated above, it may however be that the sequential data to be modeled warrants the hypothesis that at time t, past data in the sequence can be summarized concisely by a state variable. This is precisely what *Hidden* Markov Models embed: we do not assume that the observed data sequence has a Markov property (of low order); however, another, unobserved but related variable (the state variable) is assumed to exist and to have the Markov property (with low order, typically $k = 1$). HMMs are generally taken to be of order 1 because an HMM of order 1 can emulate an HMM of any higher order by increasing the number of values that the state variable can take.

To return to our robot example, suppose that when the robot is in a particular position it can do different actions, with different probabilities. The problem now is that we do not observe the trajectory of the robot, only the actions that it performs at each time step. From this we will try to infer both transition probabilities from position to position, and emission probabilities, which characterize the distribution over actions at each position.

The relation between the observed sequence $y_1^t = \{y_1, \ldots, y_t\}$ and the hidden state sequence q_1^t is shown graphically in the Bayesian network of Figure 2 and by these conditional independence assumptions (for the case of order 1):

$$\boxed{P(y_t|q_1^t, y_1^{t-1}) = P(y_t|q_t)} \tag{1}$$

$$\boxed{P(q_{t+1}|q_1^t, y_1^t) = P(q_{t+1}|q_t)} \tag{2}$$

In simple terms, the state variable q_t summarizes all the relevant past values of the observed and hidden variables when one tries to predict the value of the observed variable y_t, or of the next state q_{t+1}.

Because of the above independence assumptions, the joint distribution of the hidden and observed vari-

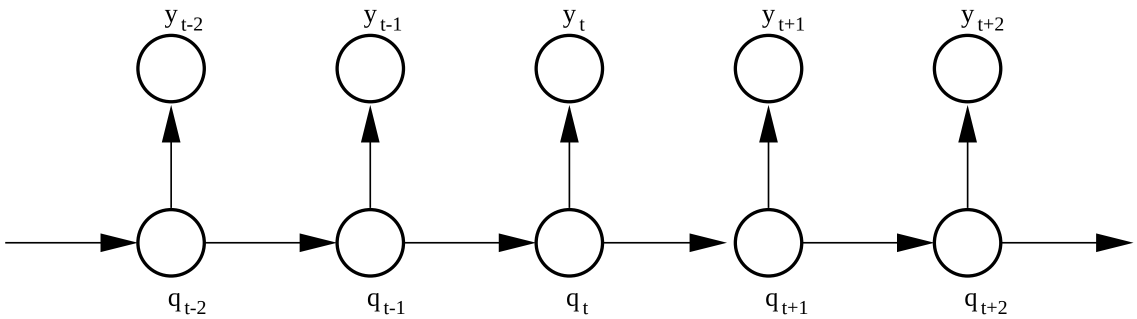

Figure 2: Bayesian network representing graphically the independence assumptions of a Hidden Markov Model (order 1). The state sequence is $q_1, \ldots, q_t, \ldots$, and the output (or observation) sequence is $y_1, \ldots, y_t, \ldots$.

ables can be much simplified, as follows:

$$\boxed{P(y_1^T, q_1^T) = P(q_1) \prod_{t=1}^{T-1} P(q_{t+1}|q_t) \prod_{t=1}^{T} P(y_t|q_t)} \tag{3}$$

The joint distribution is therefore completely specified in terms of

1. the **initial state probabilities** $P(q_1)$,
2. the **transition probabilities** $P(q_t|q_{t-1})$ and,
3. the **emission probabilities** $P(y_t|q_t)$.

In many applications in which the state variable is a discrete variable, all the state sequences are forced to start from a common initial state (i.e., $P(q_1)$ is 1 for this value of the state and 0 for the other values) and end in a common final state, and many transition probabilities are forced to have the value 0, using prior knowledge to structure the model. In the speech recognition literature, one often talks of *states* to mean the different values of the state random variable, and of a *transition* between two states (for which the transition probability is non-zero). To represent the structure imposed by the choice of zero on non-zero transition probabilities (i.e., the existence of transitions), one talks of the **topology** of an HMM. Such a topology is represented in a graph such as the one of Figure 3, in which nodes represent values of the state variable (i.e., *states*), and arcs represent *transitions* (i.e., with non-zero probability). Such a graph should not be confused with the graph of Bayesian networks introduced earlier, in which each node represents a random variable.

In a common variant of the above model, the emissions are not dependent only on the current state but also on the previous state (i.e., on the transitions):

$$P(y_1^T, q_1^T) = P(q_1)P(y_1|q_1) \prod_{t=2}^{T} P(q_t|q_{t-1})P(y_t|q_t, q_{t-1}).$$

The computation of $P(y_1^T, q_1^T)$ is therefore straightforward (done in time $O(T)$). However, q_1^T is not observed, and we are really interested in representing the distribution $P(y_1^T)$. Simply marginalizing the joint distribution yields an exponential number of terms (here when q is discrete):

$$P(y_1^T) = \sum_{q_1^T} P(y_1^T, q_1^T)$$

In the case of discrete states, there is fortunately an efficient recursive way to compute the above sum, based on a factorization of the probabilities that takes advantage of the Markov property of order 1. The recursion

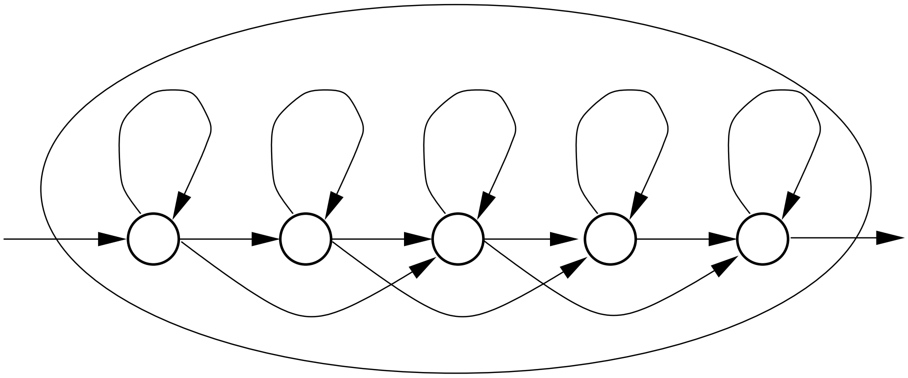

Figure 3: Example of a **left-to-right** topology for an HMM which may be used in a speech recognition system to represent the distribution of acoustic sequences associated with of a unit of speech (e.g., phoneme, word). A node represents a value of the state variable q_t. An arc represents a transition with non-zero probability between two values of the state variable. The oval in this picture corresponds to a symbolic meaning (e.g., a word) associated to the group of states within the oval.

is not on $P(y_1^t)$ itself but on $P(y_1^t, q_t)$, i.e., the probability of observing a certain subsequence while the state takes a particular value at the end of that subsequence:

$$\begin{aligned} P(y_1^t, q_t) &= P(y_t | y_1^{t-1}, q_t) P(y_1^{t-1}, q_t) \\ &= P(y_t|q_t) \sum_{q_{t-1}} P(y_1^{t-1}, q_t, q_{t-1}) \\ &= P(y_t|q_t) \sum_{q_{t-1}} P(q_t|q_{t-1}, y_1^{t-1}) P(y_1^{t-1}, q_{t-1}) \end{aligned} \tag{4}$$

$$\boxed{P(y_1^t, q_t) = P(y_t|q_t) \sum_{q_{t-1}} P(q_t|q_{t-1}) P(y_1^{t-1}, q_{t-1})} \tag{5}$$

where we used the two Markov assumptions (on the observed variable and on the state) respectively to obtain the second and last equation above. The recursion can be initialized with $P(y_1, q_1) = P(y_1|q_1)P(q_1)$, using the initial state probabilities $P(q_1)$. This recursion is true whether the model is homogeneous or not (and the probabilities can be conditioned on other variables). This recursion is central to many algorithms for HMMs, and is often called the **forward** phase. Let us note $y_1^{T_p}(p)$ for the p-th sequence of a training data set, of length T_p. The above recursion allows to compute the **likelihood** function $l(\theta) = \prod_p P(y_1^{T_p}(p)|\theta)$, where θ are parameters of the model which can be tuned in order to maximize the likelihood over the training sequences $y_1^{T_p}(p)$. The computational cost of this recursion is $O(Tm)$ when T is the length of a sequence and m is the number of non-zero transition probabilities at each time step, i.e., $m \leq n^2$ (where n is the number of values that the state variable q_t can take). Note that in many applications $m \ll n^2$ because prior knowledge imposes a structure on the HMM, in the form of zero probability for most transitions.

Once $P(y_1^T, q_T|\theta)$ is obtained, one can readily compute the likelihood $P(y_1^T|\theta)$ for each sequence as follows:

$$P(y_1^T|\theta) = \sum_{q_T} P(y_1^T, q_T|\theta).$$

Note that we sometimes drop the conditioning of probabilities on the parameters θ unless the context would make that notation ambiguous.

2.2 Choice of Distributions

2.2.1 Transition Probabilities

HMMs are conventionally taken to have a discrete-valued hidden state, with a multinomial distribution for q_t (given the previous values of the state). In this paper, we sometimes use the shortcut often found in papers on HMMs for speech recognition and talk about *different states* instead of *different values of the state random variable.*

In the discrete state case, if the model is homogeneous, the transition parameters (for Markov models of order 1) can be represented by a matrix of transition probabilities $A_{i,j} = P(q_t = i|q_{t-1} = j)$. In section 7 we discuss continuous state models, also called state-space models, in which the next-state probability distribution is usually a Gaussian whose mean is a linear function of the previous state.

2.2.2 Discrete Emissions

There are two types of emission probabilities: discrete, for **discrete HMMs**, and continuous, for **continuous HMMs**. In the first case, y_t is a discrete variable, and $P(y_t|q_t)$ is generally taken to be multinomial. If the model is homogeneous in the output distributions, its parameters are given by a matrix with elements $B_{i,j} = P(y_t = i|q_t = j)$. However, in many applications of interest, y_t is multivariate and continuous. To obtain a discrete distribution, two approaches are common:

1. Perform a vector quantization [57] in order to map each vector-valued y_t to a discrete value $\mathit{quantize}(y_t)$, and use $P(\mathit{quantize}(y_t)|q_t)$ as emission probability, or more generally.

2. Use multiple *codebooks* [58], i.e., split the vector variable y_t in sub-vectors y_{ti} which are assumed to be approximately independent, quantize them separately (with maps $\mathit{quantize}_i(y_{ti})$), and use $\prod_i P(\mathit{quantize}_i(y_{ti})|q_t)$ as emission probability. For example, in many speech recognition systems, y_{t1} represents spectral information at time t, y_{t2} represents changes in spectrum, y_{t3} represents the local average of the signal energy at time t, and y_{t4} its time derivative.

2.2.3 Continuous Emissions

For continuous HMMs, the two most commonly used emission distributions are the Gaussian distribution, and the Gaussian mixture,

$$P(y_t|q_t = i) = \sum_j w_{ji} N(y_t; \mu_{ij}, \Sigma_{ij})$$

where $w_{ji} \geq 0$, $\sum_j w_{ji} = 1$, and $N(x; \mu, \Sigma)$ is the probability of observing the vector x under the Gaussian distribution with mean vector μ and covariance matrix Σ. A variant of the continuous HMM with Gaussian mixtures is the so-called semi-continuous HMM [59, 60], in which the Gaussians are shared and the parameters specific to each state are only the mixture weights:

$$P(y_t|q_t = i) = \sum_j w_{ji} N(y_t; \mu_j, \Sigma_j).$$

where the mixture weights play a role that is similar to the multinomial coefficients of the discrete emission HMMs described above.

As in many modeling approaches, there are many discrete features of an HMM which have to be selected by the modeler, based on prior knowledge and/or the data, e.g., the number of values of the state variable, the topology of the HMM (forcing some transition probabilities to zero), the type of distribution for the emissions, which includes such choices as the number of Gaussians in a mixture, etc... In this paper we will basically not address this model selection question and restrict the discussion to the general use of prior knowledge in the topology of speech recognition HMMs, and to the numerical free parameters, i.e., those that are chosen numerically with a learning or parameter estimation algorithm.

2.2.4 Parameter Estimation

Up to this point we have only discussed the HMM as a model. Here we briefly discuss the issue of learning the parameters of the model. More details on learning algorithms will be given in sections 5.2 and 3.4.

For all the above distributions, the EM (Expectation-Maximization) algorithm [61, 18, 19, 20] can be used to estimate the parameters of the HMM in order to maximize the likelihood function $l(\theta) = P(\mathcal{D}|\theta) = \prod_p P(y_1^{T_p}(p)|\theta)$ over the set $\mathcal{D}$ of training sequences $y_1^{T_p}(p)$ of length T_p (indiced by the letter p). The EM algorithm itself is introduced below and discussed more formally in section 5.2. The EM algorithm can also used for some graphical models (Bayesian networks) [62].

When training with the EM algorithm, in addition to the emission distributions described in the previous subsection, other emission distributions of the exponential family (or mixtures thereof) could also be used. In speech and other sequence recognition applications, this algorithm can also be used when the HMM is conditioned by the sequence of correct labels $w_1^L = \{w_1, \ldots, w_L\}$, i.e., one chooses θ which maximizes the product of the class conditional likelihoods $P(y_1^T|w_1^L, \theta)$ over the training sequences.

It should be noted that other criteria than the maximum likelihood criterion can be used to train HMMs, for example to incorporate a prior on parameters, or to make the training more discriminant (focus more on doing the classification correctly). For more complex distributions than those described above or for several learning criteria other than maximizing the likelihood of the data, numerical optimization methods other than the EM algorithm are often used, usually based on the derivative of the learning criterion with respect to the parameters. When possible, the EM algorithm is generally preferred because of its faster convergence properties.

In section 5.2 we will discuss EM more formally and in a general framework that includes both HMMs and Input/Output HMMs. The basic principle of the EM algorithm is the following. Let us return to our robot example, which jumps from position to position at each time step and takes an observed action each time. If we knew the trajectory of the robot (in addition to the sequence of its actions), it would be easy to estimate the transition probabilities (by counting the relative frequencies of jumps: how many times it goes from position i to position j, divided by how many times it is in position i) and the emissition probabilities (similarly, by counting the frequencies of the different actions in each position). The estimated parameters would therefore be normalized frequencies. However, we do not know the robot's trajectory. One way to understand the EM algorithm is that we will consider all the possible trajectories, but with a weight for each trajectory that is the posterior probability of following that trajectory, given the observed actions and the current values of the parameters. For each time step, these weights give a different posterior probability for being in one of the positions (one of the states). Instead of obtaining normalized frequency counts for the re-estimated parameters, we will obtain weighted frequency counts, with the weights being the posterior probability of being in a particular state (or going from a particular state to the next state). Because these re-estimation formulae used the incorrect parameter values in the posterior weights, the new parameters may not be exactly those that maximize the likelihood, but it can be shown that they will bring the likelihood closer to a maximum. By iterating this procedure, we rapidly converge to a maximum of the likelihood.

2.3 The Viterbi Algorithm

Once a model class has been defined and the parameters of the model have been learned, there are many things can be done with the model. In this subsection, we discuss how we can infer the state sequence given the observation sequence (i.e., in the robot example, infer the most likely trajectory of the robot given the sequence of its actions).

In several applications of HMMs (as in speech recognition [17] and molecular biology [3] applications, for example), the hidden state variable is associated with a particular meaning (e.g., phonemes and words, for speech recognition). To each state corresponds a classification label (and several states are grouped together with the same label, as in Figure 4). To each state sequence corresponds a sequence of classification labels (e.g., words, characters, phonemes). It is therefore useful, given an observed sequence y_1^T, to infer the most likely state sequence q_1^T corresponding to it. This is achieved with algorithms that perform the following

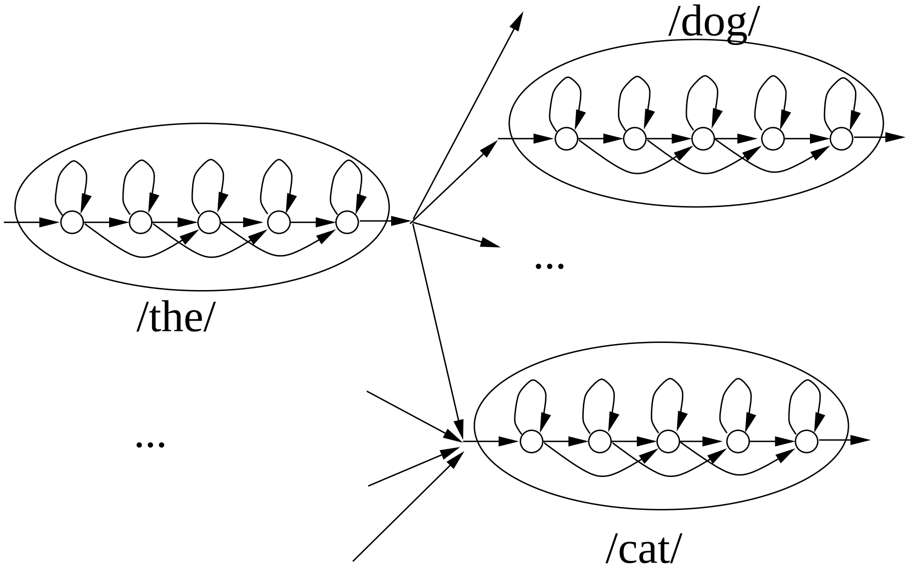

Figure 4: This figure shows part of the topology of an HMM which may be used for recognizing connected words, with groups of state values (represented by nodes here) associated with a meaning, e.g., a word label. A word HMM is represented by an oval that groups the corresponding set of states. A state sequence also corresponds to a sequence of words. Transition probabilities between word models are given by the language model.

maximization:

$$\boxed{q_1^{T*} = \text{argmax}_{q_1^T} P(q_1^T|y_1^T) = \text{argmax}_{q_1^T} P(q_1^T, y_1^T)}$$

The Viterbi algorithm [63] finds the above maximum with a relatively efficient recursive solution (of computational cost proportional to the number of non-zero transitions probabilities times the sequence length). This is in fact an application of Bellman's dynamic programming algorithm [64]. First let us define

$$V(i,t) = \max_{q_1^{t-1}} P(y_1^t, q_1^{t-1}, q_t = i)$$

which can be computed as follows, using the Markov conditional independence assumptions (equations 1 and 2):

$$\begin{aligned} V(i,t) &= P(y_t|q_t = i) \max_j P(q_t = i|q_{t-1} = j) V(j, t-1) \\ j^*(i,t) &= P(y_t|q_t = i) \text{argmax}_j P(q_t = i|q_{t-1} = j) V(j, t-1) \end{aligned} \qquad (6)$$

with the initialization $V(i,1) = \max_{q_1} P(y_1|q_1)P(q_1)$, and $j^*(i,t)$ recording the "best previous state" from state i at time t. We therefore obtain at the end of the sequence $\max_{q_1^T} P(y_1^T, q_1^T) = \max_i V(i,T)$. The optimal q_1^{T*} can be obtained using j^* in a backward recursion, starting from $q_T^* = \text{argmax}_i V(i,T)$, with $q_{t-1}^* = j^*(q_t^*, t)$. Like the forward phase, the computation cost of the Viterbi algorithm is $O(Tm)$ (where m is the number of non-zero transition probabilities at each time step).

When the number of non-zero transition probabilities m is large, other graph search algorithms may be used in order to look for the optimal state sequence. Some are optimal (e.g., the A^* *search* [65]) and others are approximate but faster (e.g., the *beam search* [66]). For very large HMMs (e.g., for speech recognition with several tens of thousands of words), even these methods are not efficient enough. The methods that are employed for such large HMMs are based on progressive search, performing multiple passes. See [67, 68, 69, 70] for more details.

3 Speech Recognition with HMMs

Because speech recognition has been the most common application of HMMs, we will discuss here some of the issues this involves, although this discussion is relevant to many other applications. See [71] for a recent survey of statistical methods for speech recognition. The basic speech recognition problem can be stated as follows: given a sequence of acoustic descriptors (obtained by pre-processing the speech signal, e.g., spectral information represented by a vector of between approximately 10 and 40 numbers, obtained at a rate of around 10 millisecond per time step), find the sequence of words intended by the speaker who pronounced those words.

3.1 Isolated Speech Recognition

Let us first consider the case of isolated word recognition, which is simpler than connected speech recognition. For isolated word recognition, a single HMM can be built for each word w within a preset vocabulary. With these models one can compute $P(y_1^T|w)$ for each word w within the vocabulary, when an acoustic sequence y_1^T is given. When an a priori distribution $P(w)$ on the words is also given, the most likely word w^* given the acoustic sequence can be obtained by picking the model which maximizes both the acoustic probability and the prior:

$$w^* = \text{argmax}_w P(w|y_1^T) = \text{argmax}_w P(y_1^T|w)P(w) \tag{7}$$

The computational cost for recognition is simply the number of words times the computation of the acoustic probability for a word (forward computation, equation 5). The recognition time can however be significantly reduced by using search techniques mentioned in the previous section.

This application of HMMs examplifies a typical situation in several applications: many observation sequences are given, each with a single class label (here the correct word), and we would like to use the model to classify new sequences. This is a simple extension of the usual machine learning classification framework to the case of sequential inputs (rather than fixed-size vectors). In the next section we look at a case in which the class labels also have a sequential structure and we which to take advantage of it in the model.

3.2 Connected Speech Recognition

A more interesting task is that of recognizing connected speech, since users do not like to pause between words. In that case, we can generalize the isolated speech recognition system by considering the mapping from an acoustic sequence to a sequence of words:

$$w_1^{l*} = \text{argmax}_{w_1^L} P(w_1^L|y_1^T) = \text{argmax}_{w_1^L} P(y_1^T|w_1^L)P(w_1^L) \tag{8}$$

The high-level task (which can occur in applications than speech recognition) is therefore to learn to map sequences of symbols or vectors (acoustic observations) into shorter sequences of symbols (phonemes).

In the above equation we introduced a **language model** $P(w_1^L)$, which characterizes the structural dependencies in sequences of symbols (e.g., phonemes, words). See [72] for a collection of review papers on this subject. The language model is a crucial element of modern speech recognition systems (and speech understanding systems, which translate speech into actions), because most word sequences are very unlikely in a particular language, and in a particular semantic context. The quality of the language model of humans may be one of the most important factors in the superiority of speech recognition by humans over machines.

It is clearly not computationally practical to directly enumerate the word sequences w_1^L above. The solutions generally adopted are based on representing the language model in a graphical form and using search techniques to combine the constraints from the acoustic model ($P(y_1^T|w_1^L)$) with those from the language model ($P(w_1^L)$). A very common type of language model is based on restricting the context to word bigrams ($P(w_i|w_{i-1})$) or trigrams ($P(w_i|w_{i-1}, w_{i-2})$). Such language models have a simple Markovian interpretation and can be combined with the acoustic HMMs to build a large HMM in which transition probabilities between HMMs representing a word (possibly in the context of other words) are obtained from

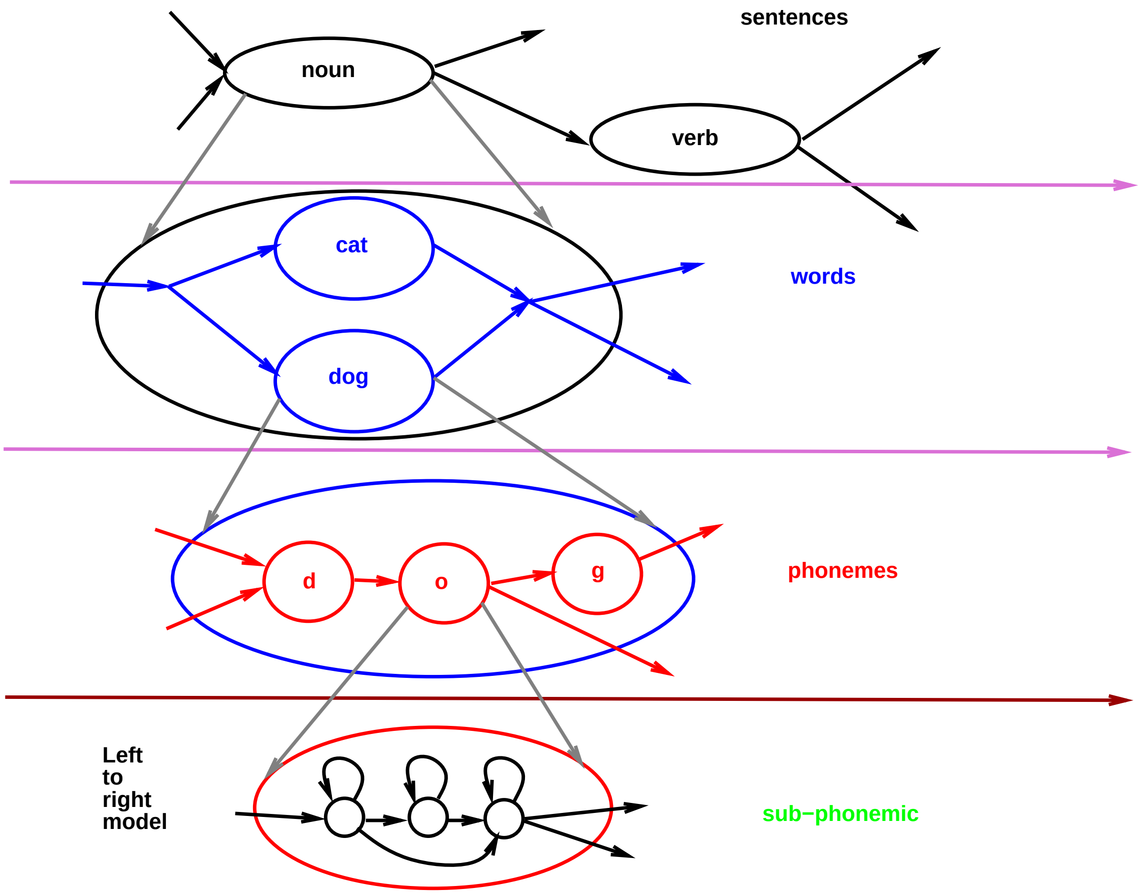

Figure 5: Example of hierarchical organization of speech knowledge in the topology of an HMM. Each level can be represented by a different weighted transducer or acceptor. Arcs represent transitions between HMM states at a certain level (or groups of states at a lower level). Low-level models (e.g. phoneme models) are shared in many places in the overall HMM.

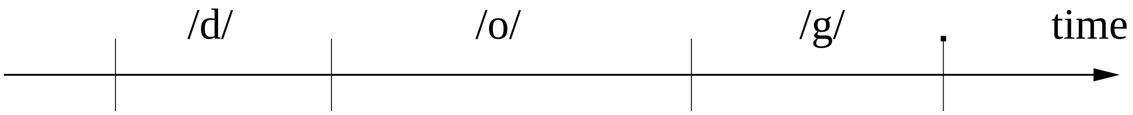

Figure 6: The Viterbi or other search algorithms give a segmentation of the observation sequence in consecutive temporal segments associated with a classification label. Here for example, the temporal segments are associated with speech units for phonemes /d/, /o/, and /g/ (part of a word /dog/, as in Figure 5).

the language model. For example, $P(w_i|w_{i-1})$ may be used as a transition probability between the final state of a HMM for word w_{i-1} and the initial state of a HMM for word w_i, as illustrated in Figure 4. When more context is used, different instantiations of each word may exist corresponding to different contexts, making the overall HMM (representing the joint probability $P(y_1^T, w_1^L)$) very large. Such large HMMs are often not represented explicitly in a computer but instead particular instances of a word HMM are "created" when needed by a search algorithm that traverses the large "virtual" HMM. Transducers (see section 6) are an elegant way to represent such complicated data structures in a uniform and mathematically well-grounded framework. The Viterbi or other search algorithms may be used to look for the optimal state sequence. In turn this state sequence corresponds to a sequence of words. Note that the most likely sequence of words may be different from the one obtained from the most likely state sequence, but it would be computationally much more expensive to compute. In practice very good results are obtained with this approximation.

The most likely state sequence also gives a **segmentation** of the speech, i.e., a partition of the observed sequence in consecutive temporal segments, as illustrated in Figure 6.

For a given sequence of **speech units** (e.g., words, phonemes), this segmentation therefore gives an alignment between the observed sequence and the "template" that the sequence of speech units represents.

3.3 HMM Topology from A Priori Knowledge

A priori knowledge about an application (such as speech and language) may be used to impose a structure on an HMM and a meaning for the values of the state variable. We have already seen that each state may be associated with a certain label. Furthermore, the topology of the HMM can be strongly constrained: most transition probabilities are forced to be zero. Since the number of free parameters and the amount of computation are directly dependent on the number of non-zero transition probabilities, imposing such structure is very useful. Furthermore, imposing such structure can almost completely answer one of the most difficult questions in constructing an HMM (including not only its parameters but also its structure): what should the hidden state represent? The most basic structure that is often imposed on speech HMMs is the **left-to-right** structure: states (e.g. within a word HMM) are ordered sequentially and transitions go from the "left" to the "right", or from a state to itself, as in Figures 3, 4 and 5.

The set of states is generally partitioned in subsets to which a particular linguistic meaning is attached (e.g., phoneme or word, in a particular context). An example of the topology of a part of an HMM for speech recognition is shown in Figure 5. To reduce the number of free parameters and help generalize, designers of speech recognition HMMs use the notion of a **speech unit** [58] (representing a particular linguistic meaning and the associated distribution on acoustic subsequences) which can be re-used (or shared) in many different places of an HMM. The simplest set of speech unit one can think of is simply the phoneme. For example, a speech unit for phoneme /a/ may be used in several higher-level units such as words that contain an /a/ in their linguistic definition. More complex speech units are **context-dependent**: they represent the acoustic realization of a linguistic unit in the context (left or right) of other linguistic units. As illustrated in Figure 5, each word can be pronounced as a sequence of phonemes: each word HMM can be built as a concatenation of corresponding speech units. If there are multiple pronunciations for a word, then a word HMM would be made of several of these concatenations in parallel.

The constraints on the topology and sharing of parameters across states associated to different instances of the same speech unit represent very strong assumptions on the relation between the speech signal and the

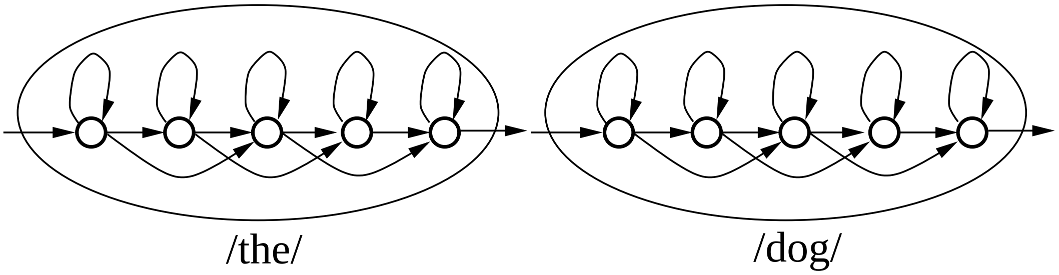

Figure 7: Example of a constrained HMM, representing the conditional distribution $P(y_1^T|w_1^L)$. Here y_1^T is an acoustic sequence and w_1^L is the sequence of two words /the/ and /dog/.

word sequence, and these assumptions are known to be too strong. The focus of much current research in this field is therefore to build more faithful models (while keeping them tractable), or make sure that the imperfections of the model do not hurt too much the final decision taking. These assumptions, however, have been found very useful in practice, in order to build the current state-of-the-art speech recognition systems [17] and applications to bioinformatics [36, 3]. See also in [73] the use of a prior topology to discover sequential clusters in data. More generally, much research in learning language models addresses this issue of learning a structure for the model, in addition to the parameters. See for example [74, 75] and [76] for polynomial-time algorithms that constructively learn a probabilistic structure for the language by merging states, and [45] on the learnability of acyclic probabilistic finite automata.

3.4 Learning Criteria

According to what learning criterion should we choose the parameters of the HMM? In this subsection we discuss two types of criteria: those aimed at fitting the observed sequences, and those aimed at discriminating among the observation sequences associated with different classes (word sequences).

In many applications of HMMs such as speech recognition, there are actually two sequences of interest: the observation (e.g., acoustic) sequence, y_1^T, and the classification (e.g. correct word) sequence, w_1^L. The traditional approach to HMM speech recognition is to consider independently a different HMM for each word (or speech unit) and to learn the distribution of acoustic subsequences associated with each word or speech unit. By concatenating the speech units associated with the correct word sequence w_1^L, one can represent the conditional acoustic probability $P(y_1^T|w_1^L)$. An HMM that is constrained by the knowledge of the correct word sequence is called a constrained model, and is illustrated in Figure 7. On the other hand, during speech recognition, the correct word sequence is unknown, and all the word sequences allowed by the language model must be taken into account. An HMM that allows all these word sequences is called a **recognition model** (and it is generally much larger than a constrained model). It represents the joint distribution $P(y_1^T, w_1^L)$ (or, when summing over all possible state paths, the unconditional probability $P(y_1^T)$).

A **maximum likelihood criterion** can then be applied to estimate the parameters of the speech units that maximize the constrained acoustic likelihood, $l(\theta) = \prod_p P(y_1^{T_p}(p)|w_1^L(p), \theta)$, over all the training sequences (indiced by p above). For this purpose, the EM or GEM algorithms described in section 5.2 are often used.

The above approach is called **non-discriminant** because it is based on learning the acoustic distribution associated with each speech unit (i.e., class-conditional density functions), rather than learning how the various linguistic classes differ acoustically. When trying to discriminate between different interpretations w_1^L (i.e., different classes), it is sufficient to know about these differences, e.g, using $P(w_1^L|y_1^T)$ or even directly describing the decision surface in the space of acoustic sequences. The non-discriminant models contain the additional information $P(y_1^T)$, since they learn $P(y_1^T, w_1^L) = P(y_1^T|w_1^L)P(w_1^L) = P(w_1^L|y_1^T)P(y_1^T)$. That includes both $P(w_1^L|y_1^T)$ and $P(y_1^T)$ but $P(w_1^L|y_1^T)$ is sufficient to perform a classification, so non-discriminant approaches do more work: they learn $P(y_1^T)$. Furthermore, non-discriminant approaches strongly rely on the assumptions made on the form of the probability density of the acoustic data. Since the models chosen to represent the data are generally imperfect, it has been found that better classification results can often be obtained when the objective of learning is closer to the reduction of the number of classification errors. Several approaches have been proposed to train HMMs with a discriminant criterion. The most common are the maximum a posteriori criterion, to maximize $P(w_1^L|y_1^T)$, and the maximum mutual information criterion [77, 78, 79], to maximize $\log \frac{P(y_1^T|w_1^L)}{P(y_1^T)}$. The maximum mutual information criterion is therefore obtained by comparing the log-probability of the constrained model, $\log P(y_1^T|w_1^L)$, with the log-probability the unconstrained recognition model, $\log P(y_1^T) = \log \sum_{w_1^L} P(y_1^T, w_1^L)$ allowing all the possible interpretations (word sequences). Maximizing this criterion attempts to increase the likelihood of the correct (i.e., constrained) model while decreasing the likelihood of all the other models. Other criteria have been proposed to approximate the minimization of the number of classification errors [80, 81].

A gradient-based numerical optimization method is generally used with these criteria: the EM algorithm cannot be used in general (an exception is the synchronous or asynchronous Input-Output HMM with discrete observations, described in section 5).

3.5 Performance

The performance of speech recognition systems based on HMMs varies a lot depending on the difficulty of the task. Benchmarks to compare speech recognition systems have been set up by ARPA [82] in the U.S.A.. The difficulty increases with the size of the vocabulary, the variability of the speech among the speakers, and other factors. For example, on the ATIS benchmark (where the task is to provide airline information to users, and the vocabulary has around 2000 words), laboratory experiments yielded around 5% of incorrectly answered queries [83]. This task involves not only recognition but also understanding. On a large vocabulary task set up by ARPA with around 60000 words (no understanding, only recognition), the word error rates reported are below 11%. However, performance of speech recognition systems is often worse in the field than in the laboratories. Speech recognition is now used in commercial applications, as in the AT&T telephone network. This system looks for one of five keywords. It makes an error in less than 5% of the calls and processes around one billion calls per year [83].

3.6 Imbalance Between Emission and Transition Probabilities

One problem that is faced in classical applications of HMMs is that, on a logarithmic scale, the range of values that emission probabilities $P(y_t|q_t)$ can take is much larger from that of transition probabilities $P(q_{t+1}|q_t)$, because typically there are only a few allowed transitions from a state to the next state, whereas the space of observations is very large (e.g., continuous).

As a consequence, the path chosen by the Viterbi (or other search) algorithm,

$$q_1^{T*} = \text{argmax}_{q_1^T} P(q_1) \prod_{t=1}^{T-1} P(q_{t+1}|q_t) \prod_{t=1}^{T} P(y_t|q_t), \tag{9}$$

is mostly influenced by the emission probabilities. When comparing two paths with equation 9, what makes the most difference is whether the emissions are well modeled by the sequence of states that is compatible with the topology of the HMM, i.e., with the choice of the existence or non-existence of transitions (obtained by forcing some transitions to have zero probability). In the extreme case, if the numerical value of non-zero transition probabilities are completely ignored, the Viterbi algorithm only does a "dynamic time-warping" match [84] between the observation sequence and the sequence of probabilistic prototypes associated (through the emission distributions) with a sequence of state values in the HMM. Some operational speech recognition models actually ignore transition probabilities altogether, because of this problem. Many others introduce a so-called "fudge factor" to adjust the weighing of acoustic and language model probabilities. This is usually achieved by taking a linear combination of log-probabilities from the acoustic model (i.e., emissions) and language model (i.e. transitions). This was used for example in [32]. These heuristic solutions point out a weakness in our current probabilistic framework for modeling speech.

The notion of **duration** in speech models refers to the segmentation of the acoustic sequence into intervals associated to different speech units (e.g. phonemes). Some phonemes tend to last longer than others (e.g., plosives are very short, the duration of vowels can vary a lot, etc...). The distribution of durations associated with each speech unit can in principle be represented by multiple states with a left-to-right structure and appropriate transition probabilities between them. However, because of the imbalance problem, the only constraint on durations that is really effective is the one obtained from the topology of the HMM, i.e., by forcing some transition probabilities to zero. Note that learning algorithms for parametric models, such as the EM algorithm, cannot be used to learn such discrete structure: instead the topology of the HMM is often decided a priori. Ignoring the value of non-zero transition probability corresponds to assigning a uniform probability for the duration within certain intervals and zero outside these intervals.

In section 5 we discuss the recently proposed asynchronous Input-Output HMMs, which could significantly alleviate this problem.

4 Integrating Artificial Neural Networks and HMMs

Artificial Neural Networks (ANNs) or connectionist models have been successfully used in several pattern recognition and sequential data processing problems. Multi-layered ANNs [85] can represent a non-linear regression or classification model. Different researchers have proposed different ways to combine ANNs with HMMs, in particular for automatic speech recognition. What most of the proposed hybrids have in common is that the neural networks are used to capture temporally local, but possibly complex and non-linear dependencies, while the HMM is used to handle the temporal structure and the elastic time alignment which is one of the strong and useful a-priori in speech HMMs. Put simply (but the details differ), the neural networks have been generally used within the part of the model that represents the emissions (the type of observations associated to each state). How do these hybrids mainly differ? there are several mathematical formulations that have been proposed to integrate into a HMM the output of a neural network that takes as input local speech observations. In some cases, the neural network and the HMM are trained separately, whereas in some cases the training of both is with respect to a single criterion. This is normally the case when the ANN is part of a Markovian probabilistic model (e.g., to compute emission probabilities). In some approaches, the ANN is seen more as a preprocessor for the HMM, to transform the raw observations in a form that is easier to model (but even in this case it is possible to train both with respect to a single, discriminant criterion). Finally, in some approaches, the ANN is used in a post-processing phase to refine the scores that the HMM associates to each segment of speech.

The proposed advantages of such systems include more discriminant training, the ability to represent the data with richer, non-linear models (in comparison to Gaussian or discrete models) and the improved incorporation of context (by using as input multiple lagged values of the input variable). Some models (such as the Input-Output HMMs described in the next section) are also designed to learn long-term dependencies better and to eliminate the problem of imbalance between emission and transition probabilities, therefore yielding more effective models of duration. Several new variants of HMMs such as the ANN/HMM hybrids attempt to address some of the modeling weaknesses in HMMs as they are used for speech recognition, such as the incorrectness of the two Markov assumptions (with respect to the interpretation of state values that is made in these models), the poor modeling of phoneme duration (as discussed in section 3.6), and the poor use of some of the contextual information (including both short-term acoustic context and long-term context such as prosody).

A left-to-right HMM can be seen as a flexible template and the Viterbi algorithm as a sophisticated way to align that template to the observed speech. Since ANNs were successful at classifying individual phonemes, initial research focused on using the dynamic programming tools of HMMs in order to go from the recognition of individual phonemes (or other local classification) to the recognition of whole sequences [86, 87, 88, 89, 90, 91, 92, 93, 94, 2]. In some cases [95, 92, 2, 93, 94], the ANN outputs are not interpreted as probabilities, but are rather used as scores and generally combined with a dynamic programming algorithm akin to the

Viterbi algorithm to perform the alignment and segmentation.

In some cases the dynamic programming algorithm is embedded in the ANN itself [93, 96]. Alternatively, the ANN can be used to re-score the N-best hypotheses of phoneme segmentation produced with an HMM [97], by assigning posterior probabilities to the phonemes for each of the phonetic segments hypothesized with the HMM. An HMM can also be viewed as a particular kind of recurrent [85] ANN [98, 99]. Although the ANN and the HMM are sometimes trained separately, most researchers have proposed schemes in which both are trained together, or at least the ANN is trained in a way that depends on the HMM. The models proposed by Bourlard et al. rely on a probabilistic interpretation of the ANN outputs [87, 88, 1, 100]. The ANN is trained to estimate posterior probabilities of HMM states, given a context of observation vectors, $P(q_t|y_{t-k},\ldots,y_{t-1},y_t,y_{t+1},y_{t+k})$, centered on the current time step. By normalizing these posteriors with state priors $P(q_t)$, one obtains scaled emission probabilities $\frac{P(y_t|q_t)}{P(y_t|y_1^{t-1})}$. These scaled emission probabilities are used in the usual Viterbi algorithm for recognition. Training of the ANN is based on the optimal state sequence obtained from the constrained HMM (with knowledge of the correct word sequence). For each time step, the ANN is supervised with a target value of 1 for the correct state and a target value of 0 for the other states. This procedure has been found to converge and yield speech recognition performance at the level of state-of-the-art systems [1, 100]. Bourlard et al. draw links between this procedure and the EM algorithm; however, this procedure does not optimize a well-defined criterion during training: training is based on the local targets provided by the constrained Viterbi alignment algorithm.

Another approach [2, 101, 32] uses the ANN to transform the observation sequence into a form that is easier to model for an HMM that has simple (but continuous) emission models (e.g., Gaussian or Gaussian mixture). The ANN is used as a non-linear trainable pre-processor or feature extractor for the HMM. In that case, the objective of learning for the combined ANN/HMM system is given by a single criterion defined at the level of the whole sequence, rather than at the level of individual observations or segments (e.g., for phonemes or characters). In some applications of this idea, the ANN is viewed as an "object" spotter (e.g., a phoneme or a character, for speech or handwriting recognition), and the HMM as a post-processor that can align the sequence of outputs from the ANN with a higher-level (e.g., linguistic and lexical) model of the temporal structure of the observed sequences. This model was introduced in [101] for phoneme recognition. It is also described in [2], and was extended to character recognition in [32]. The ANN transforms an input sequence u_1^T into an intermediate observation sequence y_1^T, with a parameterized function $y_1^T = f(u_1^T, \theta)$. For example, this function may capture some of the contextual influences, and transform the input in a way that makes it more invariant with respect to the classifications of interest. A basic idea of the implementation of this model is that the optimization criterion C used to train the HMM is a continuous and differentiable function of the intermediate observations y_t. Therefore, the gradients $\frac{\partial C}{\partial y_t}$ can be used to train the parameters θ of the ANN: gradient descent using the chain rule for derivatives (also called **back-propagation** [85]) yields the parameter gradients

$$\frac{\partial C}{\partial \theta} = \sum_t \frac{\partial C}{\partial y_t}\frac{\partial y_t}{\partial \theta}.$$

for a single sequence. Two criteria have been considered: the maximum likelihood criterion and the maximum mutual information criterion. In both cases the derivatives $\frac{\partial C}{\partial y_t}$ can be obtained from the state posteriors $P(q_t|y_1^T)$ which would have to be computed for the EM algorithm.

In some cases of ANN/HMM hybrids, it is possible with an a priori idea of what the ANN should accomplish to train the ANN and the HMM separately. However, it has been shown experimentally with the above ANN/HMM hybrid how training the ANN jointly with the HMM improves performance on a speech recognition problem [101, 2], bringing down the error rate on a plosive recognition task from 19% to 14%. It has later been shown how using joint training with respect to a discriminant criterion on a handwriting recognition problem [32] reduced the character error rate from 12.4% down to 8.2% (no dictionary), or from 2% down to 1.4% (with a 350-word dictionary). The idea of training a set of modules together (rather than separately) with respect to a global criterion with gradient-based algorithms was proposed several years ago [102, 103, 92].

Another way to integrate ANNs with HMMs in a mathematically clear way is based on the idea of

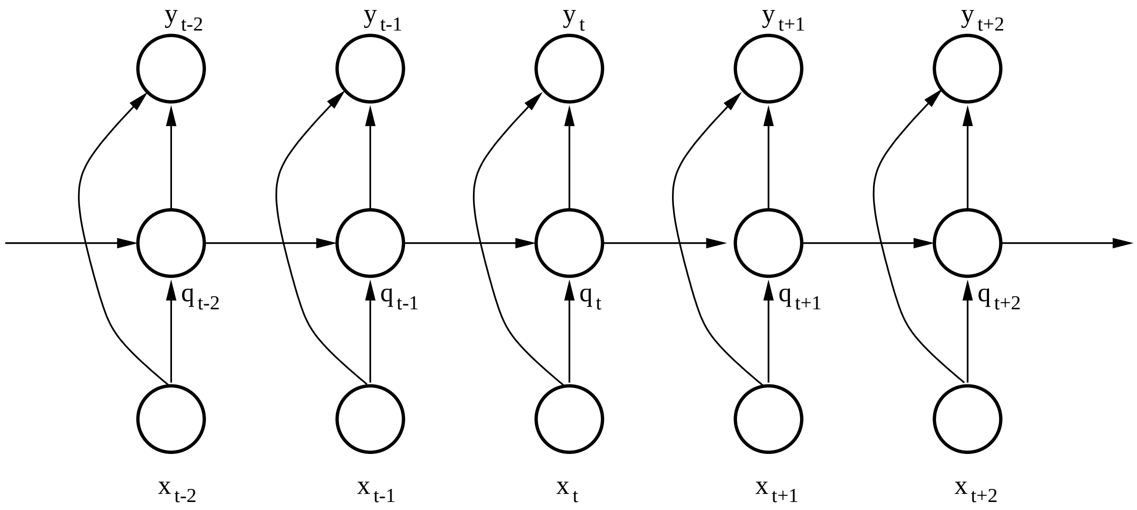

Figure 8: Bayesian network representing graphically the independence assumptions of a synchronous Input-Output Hidden Markov Model. The state sequence is $q_1, \ldots, q_t, \ldots$, the output sequence is $y_1, \ldots, y_t, \ldots$, and the input sequence is $x_1, \ldots, x_t, \ldots$.

Input-Output HMMs described in the next section.

5 Input-Output HMMs

Input-Output Hidden Markov Models (IOHMMs) [6] (or Conditional HMMs) are simply HMMs for which the emission and transition distributions are conditional on another sequence, called the input sequence, and noted x_1^L. In that case, the observations modeled with the emission distributions are called outputs, and the model represents not the distribution of sequences $P(y_1^T)$ but instead the conditional distribution $P(y_1^T|x_1^L)$. In the simpler models first presented here, the input and output sequences have the same length, but a recent extension (section 5.3) allows input and output sequences of different lengths. Transducers (section 6) which can be seen as generalizations of such conditional distributions, also allow input and output sequences of different lengths. The conditional independence assumption of a synchronous IOHMM are represented in the Bayesian network of Figure 8.

We say that HMMs are homogeneous (in the sense that the transition and emission probability distributions do not depend on t) whereas IOHMMs are inhomogeneous (in the sense that the transition and emission probability distribution change according to the input x_t, so they change with time t).

In the simpler case in which the input and output sequences are synchronous, the mathematics of IOHMMs is very similar to that of HMMs but more general. For this reason we will explain the EM algorithm used to train HMMs (and IOHMMs) in this section (in section 5.2). Whereas in ordinary HMMs the emission distributions are given by a homogeneous model $P(y_t|q_t)$, in IOHMMs, they are given by a time-varying conditional model $P(y_t|q_t, x_t)$. Similarly, instead of time-invariant transition probabilities $P(q_t|q_{t-1})$, in IOHMMs we have $P(q_t|q_{t-1}, x_t)$. More generally values of the inputs $x_{t-k}, \ldots, x_t, \ldots, x_{t+k}$ at different time steps around x_t can be used to condition these distributions. Whereas HMMs used for pattern recognition are often trained by choosing parameters θ to maximize the likelihood of the observations given the correct classification sequence, $P(y_1^T|w_1^L, \theta)$, IOHMMs may be trained to maximize the likelihood $P(y_1^T|x_1^L, \theta)$ of decision variables y_1^T given the actually observed variables x_1^L.

In the literature on learning algorithms [5, 6, 7], IOHMMs have been proposed for sequence processing tasks, with complex emission and transition models based on ANNs. In the control and reinforcement learning literature, similar models have been called Partially Observable Markov Decision Processes [104, 105, 4]. In this case, the objective is not to model an output sequence given an input sequence, but rather, to find the strategy (a sequence of actions which can be seen as the inputs for the model) which minimizes a cost function defined on the sequence of outputs (which are observed). In this case the IOHMM represents the probabilistic relationship between the actions and the observations, with a hidden state variable. Although the HMMs such as those described in section 3 for speech recognition do represent the conditional probability $P(y_1^T|w_1^L)$ of an observation sequence given a classification sequence, this is achieved by deterministically attaching a symbolic meaning to the different values of the state variable. Instead, in IOHMMs, the state variable is stochastically related to the output variable, and for classification problems, one can view the output sequence as the classification sequence and the input (observed) sequence as the conditioning sequence. Recent work related to IOHMMs is the research on modeling tree-shaped data structures (instead of sequences), but using a similar probabilistic framework [106].

In some applications, it is conceivable that either HMMs or IOHMMs be used. For example, in speech recognition we use HMMs to model $P(acoustics|words)$, but we could instead use (asynchronous) IOHMMs to model $P(words|acoustics)$. In econometrics, we can use HMMs to model a univariate or multivariate time-series y_1^T with a model for $P(y_1^T)$. Instead we could consider that the dependent variable y_1^T should be predicted using both its past value and independent variables x_1^T, with an IOHMM $P(y_1^T|x_1^T)$. Even if we believe the x_1^T to be relevant, we could still use an HMM instead, to model the joint sequence $P(x_1^T, y_1^T)$.

Potential advantages of IOHMMs over HMMs are the following:

- When the output sequence is discrete, the training criterion is discriminant, since we use the maximum a posteriori criterion. Furthermore, when the input sequence is also discrete, the EM algorithm can be used even though the training criterion is discriminant.

- The local models (emission and transitions) can be represented by complex models such as ANNs, which are flexible non-linear models more powerful and yet more parsimonious than the Gaussian mixtures often used in HMMs. Furthermore, these ANNs can take into account a wide context (not just the observations at time t but also neighboring observations in the sequence), without violating the Markov assumptions, because there are no independence assumptions on the conditioning input variable. The ANN can even be recurrent [85, 2] (to take into account arbitrarily distant past contexts).

- Let us compare an IOHMM $P(w_1^L|y_1^T)$ and an HMM representing $P(y_1^T|w_1^L)$. In this case, the output variable of the IOHMM (w's) is discrete and takes relatively few values (e.g., phonemes), whereas the output variable of the HMM (acoustic vector) can take many more values. As explained in section 3.6, having less output values reduces the problem of imbalance between transition probabilities and emission probabilities (and would make the effect of transition probabilities more significant).

- We expect long-term dependencies to be more easily learned in IOHMMs than in HMMs, because the transition probabilities are less ergodic (i.e., the state variable does not "mix" and forget past contexts as quickly). See [107] for a development of this argument and an analysis of the difficulty of learning to represent long-term dependencies in Markovian models in general.

- Even where there is no obvious input/output relation to model (e.g., when modeling a time-series y_1^T), and it would seem that HMMs would be more appropriate, IOHMMs could offer something particular. With an IOHMM, one could use prior knowledge about what kind of transformations $f(y_1^t)$ of the past series y_1^t could be useful as summarizing informations used to condition the distribution of a future value y_{t+1}. These summarizing functions could be added to the ordinary state variable q_{t+1} as conditionning variables in "input" of an IOHMM, so the emission distribution would have the form $P(y_{t+1}|q_{t+1}, f(y_1^t))$, with $f(y_1^t)$ playing the role of an input variable for an IOHMM. In that case, we would need less values for the state variable q_t since $f(y_1^t)$ already provides much information about the past sequence.

In the next section we describe particular kinds of IOHMMs which have been proposed in the econometrics literature. In the following section, we present the EM algorithm which can be used for training both HMMs and synchronous HMMs.

5.1 Markov Switching Models

Markov Switching Models have been introduced in the econometrics literature [108, 109, 110, 14, 15] for modeling non-stationarities due to abrupt changes of regime in the economy [111, 112, 113, 114, 115, 40].

The point of view taken by most econometricians is to extend time-series regression models by the addition of a discrete hidden state variable, which allows changing the parameters of the regression models when the state variable changes its value.

Consider for example the time-series regression model

$$y_t = \beta_{q_t} x_t + e_t \tag{10}$$

where y_t is the observation (or output) variable at time t, e_t is a random variable with a zero-mean Gaussian distribution, and x_t is a vector of input variables (e.g., past values of y, as in [14], or present and past values of other observed variables). There are different sets of parameters β_{q_t} for the different (discrete) values of the hidden state variable q_t. This basically specifies a particular form for the emission distribution $P(y_t|q_t, x_t)$ of a IOHMM: a Gaussian distribution whose mean is a linear function of x_t, with different parameters for the different values of q_t. The conditional mean can also be a non-linear function of x_t, e.g., using a neural network, as in the applications of IOHMMs to modeling the distribution of future returns described in [41] (which shows that these non-linear models can yield to improved modeling of the distribution of stock market indices future returns).

To obtain a complete picture of the joint distribution of y_1^T and q_1^T (given past observed values), one then needs to specify the distribution of the state variable. In most of the cases described in the econometrics literature, this distribution is assumed to be time-invariant, and it is specified by a matrix of transition probabilities (as in ordinary HMMs), although more complicated specifications have been suggested [116, 117, 118, 119, 120].

The representation of the variance of e_t in equation 10 can be made more complex than a single constant parameter: variance can also be a function of the state variable as well as of the input variables. See for example [114, 115] for Markov-switching ARCH models applied to analyzing respectively the changes in variance of stock returns and interest rates.

The parameters of Markov switching models can generally be estimated using the EM algorithm [121, 109, 111, 122] to maximize the likelihood $P(y_1^T|\theta)$ (see next section). Other inference algorithms are used in econometrics applications [123], for filtering, smoothing, and prediction. A **filtering** algorithm is used to compute an estimate of the current distribution $P(q_t|y_1^t, x_1^t)$ for the state given past inputs and outputs. A **smoothing** algorithm is used to compute an a-posteriori estimate of the distribution $P(q_t|y_1^T, x_1^T)$ for the state path, given the whole sequence of inputs and outputs. Finally a **prediction** algorithm allows one to compute the distribution of future states and outputs given past input and output observations.

In section 7, we consider state-space models (in which the hidden state variable is continuous) and hybrids with both discrete and continuous state variables, which have been used in similar time-series modeling applications.

5.2 EM for HMMs and IOHMMs

In this section we will sketch the application of the EM (Expectation-Maximization) algorithm [61] to HMMs [18, 19, 20] and IOHMMs. The papers by Baum et al. present a special case of the EM algorithm applied to discrete emissions HMMs, but were written before the general version of the EM algorithm was described [61].

The basic idea of the EM algorithm is to use a hidden variable whose joint distribution with the observed variable is "simpler" than the marginal distribution of the observed variable itself. In HMMs and IOHMMs, the hidden variable is the state path q_1^T. We have already seen that $P(y_1^T, q_1^T)$ is simpler to compute and represent than $P(y_1^T) = \sum_{q_1^T} P(y_1^T, q_1^T)$. Because the hidden variable is not given, the EM algorithm looks at the expectation (over all values of the hidden variable) of the log-probability of the joint distribution. This expectation, called the auxiliary function, is conditioned on the previous values of the parameters, θ^k, and on the training observations. The E-Step of the algorithm consists in forming this conditional expectation:

$$F(\theta|\theta^k) = E_{\mathcal{Q}}[\log P(\mathcal{Y}, \mathcal{Q}|\mathcal{X}, \theta) \;\; | \, \mathcal{Y}, \mathcal{X}, \theta^k] \tag{11}$$

where $E_{\mathcal{Q}}$ is the expectation over $\mathcal{Q}$, $\mathcal{Y} = \{y_1^{T_1}(1), \ldots, y_1^{T_N}(N)\}$ is the set of N output sequences, and similarly $\mathcal{X}$ and $\mathcal{Q}$ are respectively the sets of N input and N state sequences. The EM algorithm is an iterative algorithm successively applying the E-Step and the M-step. The M-Step consists in finding the parameters θ which maximize the auxiliary function. At the k^{th} iteration,

$$\theta^{k+1} = \text{argmax}_\theta F(\theta|\theta^k). \tag{12}$$

It can be shown [61] that an increase of F brings an increase of the likelihood, and this algorithm converges to a local maximum of the likelihood, $P(\mathcal{Y}|\mathcal{X}, \theta)$. When the above maximization cannot be done exactly (but F increases at each iteration), we have a GEM (Generalized EM) algorithm. The maximization can in general be done by solving the system of equations

$$\frac{\partial F(\theta|\theta^k)}{\partial \theta} = 0 \tag{13}$$

For HMMs, IOHMMs and state space models with simple enough emission and transition distributions, this can be done analytically. We will discuss here the case of discrete states, where the expectation in equation 11 corresponds to a sum over the values of the state variable, and the solution of equation 13 can be obtained efficiently with recursive algorithms. To see this, we will first rewrite the joint probability of states and observations by introducing indicator variables $z_{i,t}$ with value 1 when $q_t = i$ and 0 otherwise:

$$\log P(y_1^T, q_1^T | x_1^T, \theta) = \sum_{t,i} z_{i,t} \log P(y_t | q_t{=}i, x_t, \theta) + \sum_{t,i,j} z_{i,t} z_{j,t-1} \log P(q_t{=}i | q_{t-1}{=}j, x_t, \theta)$$

The overall joint log-probability for the whole training set is a sum over the training sequences of the above sum. Moving the expectation in equation 11 inside these sums, and ignoring the p indices for sequences within the training set (which would make the notation very heavy):

$$\begin{aligned} F(\theta|\theta^k) \quad = \quad & \sum_{p,t,i} E_{\mathcal{Q}}[z_{i,t} | x_1^T, y_1^T, \theta^k] \log P(y_t | q_t{=}i, x_t, \theta) \\ & + \sum_{p,t,i,j} E_{\mathcal{Q}}[z_{i,t}, z_{j,t-1} | x_1^T, y_1^T, \theta^k] \log P(q_t{=}i | q_{t-1}{=}j, x_t, \theta) \end{aligned}$$

Note how in this expression the maximization of F with respect to the parameters θ of the emission and transition probabilities have been completely decoupled in two separate sums. To simplify the notation (and because they are often ignored in practice by forcing all state sequences to start from the same state) we have ignored the initial state probabilities. In the M-Step, the problem becomes one of simple likelihood maximization for each of the different types of distributions, *but with weights* for each the probabilities in the above sums. These weights are the state posterior probabilities

$$P(q_t{=}i | x_1^T, y_1^T, \theta^k) = E_{\mathcal{Q}}[z_{i,t} | x_1^T, y_1^T, \theta^k],$$

and the transition posterior probabilities

$$P(q_t{=}i, q_{t-1}{=}j | x_1^T, y_1^T, \theta^k) = E_{\mathcal{Q}}[z_{i,t}, z_{j,t-1} | x_1^T, y_1^T, \theta^k].$$

Let us now see how these posterior probabilities, which we will note $P(q_t | x_1^T, y_1^T)$ and $P(q_t, q_{t-1} | x_1^T, y_1^T)$ to lighten the notation, can be computed with the **Baum-Welch** forward and backward recursions [18, 19, 20].

We have already introduced the **forward recursion** (equation 5), which yields $P(y_1^t, q_t|x_1^T)$ recursively. Note that $P(y_1^t, q_t|x_1^T)$ can be normalized to perform the filtering operation, i.e., to obtain $P(q_t|y_1^t, x_1^t)$.

Using the two Markov assumptions (the equivalent of equations 1 and 2 conditioned on the input sequence), the Baum-Welch **backward recursion** can be obtained:

$$P(y_{t+1}^T|q_t, x_1^T) = \sum_{q_{t+1}} P(y_{t+1}|q_{t+1}, x_{t+1})P(q_{t+1}|q_t, x_{t+1})P(y_{t+2}^T|q_{t+1}, x_1^T)$$

By multiplying the results of the forward and backward recursion and normalizing by the output sequence probability, we obtain the state posteriors (i.e., the smoothed estimates of the state distribution):

$$P(q_t|x_1^T, y_1^T) = \frac{P(y_1^t, q_t|x_1^T)P(y_{t+1}^T|q_t, x_1^T)}{P(y_1^T)}.$$

Similarly, the transition posteriors can be obtained from these two recursions and from the emission and transition probabilities as follows:

$$P(q_t, q_{t-1}|x_1^T, y_1^T) = \frac{P(y_t|q_t, x_t)P(y_1^{t-1}, q_{t-1}|x_1^T)P(y_{t+1}^T|q_t, x_1^T)P(q_t|q_{t-1}, x_t)}{P(y_1^T)}$$

Some care must be taken in performing the forward and backward recursions in order to avoid numerical over or under flow (usually this is accomplished by performing the computation in a logarithmic scale with a small base).

The details of the parameter update algorithm depends on the particular form of the emission and transition distributions. If they are discrete, in the exponential family, or a mixture thereof, then exact (and simple) solutions for the M-Step exist (by using a weighted form of the maximum likelihood solutions for these distributions). For other distributions such as those incorporating an artificial neural network to compute conditional discrete probabilities or the conditional mean of a Gaussian, one can use a GEM algorithm or the maximization of the observations likelihood by numerical methods such as gradient ascent. Note that maximizing F by gradient ascent is equivalent to maximizing the likelihood by gradient ascent. This can be shown by noting that the quantities computed in the backward pass are in fact gradients of the likelihood with respect to the quantities computed in the forward pass:

$$P(y_{t+1}^T|q_t, x_1^T) = \frac{\partial P(y_1^T)}{\partial P(y_1^t, q_t|x_1^T)}.$$

When the representation of the state variable is more complicated than in ordinary HMMs (e.g., with multiple state variables), performing the E-Step exactly becomes difficult. See for example the models discussed in section 7.1.

5.3 Asynchronous IOHMMs

In a recent paper on asynchronous HMMs [124], it is shown how to extend the IOHMM formalism to the case of output sequences shorter than input sequences, which is normally the case in speech recognition (where the output sequence would typically be a phoneme sequence, and the input sequence a sequence of acoustic vectors). For this purpose the states can either emit or not emit an output at each time step, according to a certain probability (which can also be conditioned on the current input).

When conceived as a generative model of the output (given the input), an asynchronous IOHMM works as follows. At time $t = 0$, an initial state q_0 is chosen according to the distribution $P(q_0)$, and the length of the output sequence l is initialized to 0. At other time steps $t > 0$, a state q_t is first picked according to the *transition distribution* $P(q_t|q_{t-1}, x_t)$, using the state at the previous time step q_{t-1} and the current input x_t. A decision is then taken as to whether or not an output y_l will be produced at time t or not, according to the *emit-or-not distribution*.

In the positive case, an output y_l is then produced according to the *emission distribution* $P(y_l|q_t, x_t)$. The length of the output sequence is increased from $l-1$ to l. The parameters of the model are thus the initial state probabilities, $P(q_0 = i)$, and the parameters of the emit-or-not, emission and transition conditional distribution models, $P(\text{emit} - \text{or} - \text{not at } t|q_t, x_t)$, $P(y_l|q_t, x_t)$ and $P(q_t|q_{t-1}, x_t)$.

The application of the EM algorithm to this model is similar to the one already outlined for HMMs and synchronous IOHMMs, but the forward and backward recurrences require amounts of storage and computation that are proportional to the product of the input and output lengths, times the number of non-zero transition probabilities (whereas ordinary HMMs and synchronous IOHMMs only require resources proportional to the product of the input length times the number of transitions).

A recognition algorithm (which looks for the most likely output and state sequence) can also be derived, similarly to the Viterbi algorithm for HMMs. This recognition algorithm has the same computational complexity as the recognition algorithm for ordinary HMMs, i.e., the number of transitions times the length of the input sequence.

Asynchronous IOHMMs have been proposed for speech recognition [124] but could be used in other applications to map input sequences to output sequences of a different length. They represent a particular type of probabilistic transducers, discussed in the next section.

6 Acceptors and Transducers

One way to view an HMM is as a way to weigh various hypotheses. For example, in speech recognition HMMs, different sequences of speech units (corresponding to a subset of the possible state sequences) are associated with different weights (in fact the joint probability of these state sequences and the acoustic sequence). More generally, weighted acceptors and transducers [8, 9, 10, 11] can be used to assign a weight to a sequence (or a pair of input/output sequences). Weighted acceptors and transducers are attractive in applications such as speech recognition and language processing because they can conveniently and uniformly represent and integrate different types of knowledge about a sequence processing task. Another advantage of this framework is that it deals easily with sequences of different lengths. Furthermore, algorithms for transducers and acceptors can be applied to weight structures which include but are not limited to probabilities (and this can be useful when the sequence processing task involves the processing of numbers which do not necessarily have a probabilistic interpretation).

A weighted **acceptor** maps a sequence into a scalar (which may be a probability, for example). A weighted **transducer** maps a pair of sequences into a scalar (which may be interpreted as a conditional probability of one sequence given another one).

Weighted acceptors and transducers can be represented by labeled weighted directed graphs. The label on arcs of an acceptor graph can be an element of the set of "output" values or it can be the special "null symbol". Two labels are associated with the arcs of a transducer graph: the input label and the output label, both of which can take the special "null symbol" value. The output sequence associated with a path of a graph associated with an acceptor or transducer is obtained from the sequence of non-null output values along that path. Because of the null symbol, the input and output sequences need not have the same length.

A speech recognition HMM for which labels are associated with subsets of state values (i.e., speech units) is in fact a transducer, with weights that are probabilities. It represents the joint distribution of speech unit label sequences and acoustic observations sequences. Transducers are convenient to represent the hierarchical structure of linguistic units that designers of speech recognition systems usually embed in HMMs. A language model $P(w_1^L)$ for speech recognition is in fact an acceptor that assigns a probability to every possible sequence of labels in a certain language. An acoustic transducer $P(y_1^t|u)$ assigns a probability to each speech unit u (e.g. a phoneme in a particular context), for a subsequence of acoustic data y_1^t. Intermediate transducers represent the mapping between sequences of speech units and sequences of words, e.g., $P(u_1^N|w_1^L)$.

A generic composition operation [8, 9, 10, 11] allows to combine a cascade of transducers and acceptors, e.g., the joint distribution over acoustics, phonetic speech units, and words (with conditional independence between the different levels),

$$P(w_1^L, u_1^N, y_1^T) = P(y_1^T|u_1^N)P(u_1^N|w_1^L)P(w_1^L), \tag{14}$$

integrates different levels of knowledge about the data (e.g., as in the hierarchical representation of speech shown in Figure 5).

Search algorithms (like the Viterbi algorithm, beam search, A^*, etc...) can be used to look for the most likely sequence of values for all the intermediate variables (e.g., states in HMMs, speech units, words).

6.1 Generalized Transducers

A way to generalize transducers was recently proposed [125, 126] which allows any kind of data structure to be used as "labels" (instead of discrete symbols) in the sequences to be processed, generalizes the transducers to parameterized operations on weighted graphs, and allows a cascade of these generalized transducers and acceptors to be jointly trained with respect to a global criterion.

In this framework, data processing is viewed as a transformation of directed acyclic weighted graphs into other directed acyclic weighted graphs, which we will call **hypotheses graphs**. These graphs are different from the graphs which may be used to represent transducers and acceptors. They have a start node and an end node, and they typically represent a set of hypotheses about the input data: each path from an initial node to a final node corresponds to a distinct hypothesis, with a weight that is the sum (or the product) of the weights on the individual arcs. For example, let us consider the special case of IOHMMs: when given the input sequence x_1^T, we are able to compute with the IOHMM a hypothesis graph that contains all the possible sequences y_1^T, with a weight $P(y_1^T|x_1^T)$ for each path. In this case, the graph might simply be a regular lattice with $T \times m$ nodes when $y_t \in \{1, \ldots, m\}$. More generally, a hypotheses graph may have a different structure, and this structure may be data-dependent. This feature is used profitably in the document recognition applications described in [125, 126]. When normalized over all the paths, these path weights can be formally interpreted as probabilities for different hypotheses (conditional on the assumption that the correct hypothesis is one of those represented in the graph). Note again that although these weights can be formally interpreted as probabilities, they should be viewed as tools for decision-taking, rather than the actual and true probabilities that certain events would take place.

An object that maps one such graph to another one is called a **transformer** and can be viewed as a generalization of a transducer. In our example, the IOHMM that represents the model $P(y_1^T|x_1^T)$ can be used as a transformer. As in equation 14, many transformers can be stacked on top of each other, in a processing style that resembles the multi-layer neural networks, but in which the intermediate variables are not simple numeric vectors but instead graphs representing a set of sequential interpretations for some data, with arbitrary data structures attached to the arcs of the graph.

Whereas training algorithms for IOHMMs or HMMs require that both input and output sequences be observed and each IOHMM in a composition is trained separately, in this framework the different transformers can be trained jointly with respect to a single performance criterion, usually computed at the last stage of the transformers cascade. As in multi-layer neural networks, the parameters of a transformer can be learned by propagating gradients with respect to this criterion in the reverse direction.

This approach was successfully used as part of a document analysis system [125, 126] that reads amounts from check images. It is used by customers of NCR to process millions of checks per day. The transducers cascade incorporates a sequence of processing stages, such as generating field location hypotheses, segmentation hypotheses, isolated character recognition hypotheses, and a language model.

6.2 Variable Length Markov Models

In this section we will briefly mention some constructive learning algorithms for acceptors and transducers, which learn to process discrete sequences (e.g., for language modeling tasks).

A Variable Length Markov Model [12] is a probability model over strings in which the state variable is not hidden: its value is a deterministic function of the past observation sequence. However, this function uses more or less of the past sequence for different contexts, hence the name, variable length Markov model. For subsequences which are frequent, a deeper context is maintained. The probability of a sequence has the form

$$P(y_1^T) = \prod_t P(y_t | y_{t-d(y_1^{t-1})}^{t-1}),$$

where $d(y_1^{t-1})$ is the depth of context when all the preceding symbols are y_1^{t-1}. When $d = 0$ the next output distribution is unconditional. A tree of suffixes for past contexts is used to efficiently represent this distribution, with each node representing a particular context y_{t-d}^{t-1}, and the children of a node representing contexts that are deeper by one time step. A constructive, on-line (one-pass), learning algorithm was proposed to adaptively grow this tree [12]. Each node of the tree at depth d represents a particular value y_{t-d}^{t-1} of the context of depth d, and may be associated with a distribution over the next symbol y_t. The basic idea is to add a child to a node (i.e., deeper context for certain values of the context) when one measures a sufficiently large Kullback-Liebler divergence (or relative entropy) of the next-output distribution of the child from that of the parent node. The potential branching factor of the tree is equal to the size of the alphabet for y_t, but most nodes may have much fewer children.

More recently, an extension of this idea to probabilistic but synchronous transducers was proposed [13]. The conditional distribution of an output sequence y_1^T given an input sequence x_1^T has the form

$$P(y_1^T | x_1^T) = \prod_t P(y_t | x_{t-d(x_1^{t-1})}^{t})$$

and it can also be represented by a similar tree, where each node represents a particular input context, associated with a distribution on the next output given that input context, and the root is associated with the unconditional distribution of the next output. An on-line, one-pass, constructive learning algorithm for suffix tree transducers is proposed that adaptively grows the tree when new contexts are encountered (possibly up to a maximum depth D). A simple pruning algorithm can be used to discard deep nodes with low posterior probability (i.e., the normalized product of the probability of emitting the right data, times a prior which depends exponentially on the depth). Using these posteriors, a mixture over a very large family of such trees can be formed, whose generalization performance tracks that of the best tree in that family [13]. These algorithms were used in language modeling [38, 13] and handwritten character recognition [39].

7 State Space Models

In this section we draw a few connections between HMMs (which traditionally are based on a discrete hidden state) and state space models, which can be seen as HMMs with a continuous vector state variable.

To keep the mathematics tractable, most state space models are restricted to a transition model which is Gaussian with a mean vector that is a linear function of the previous state (and possibly of the current inputs, for input/output models):

$$P(q_t | q_{t-1}, x_t) = N(q_t; Aq_{t-1} + Bx_t, \Sigma(q_{t-1}, x_t))$$

where $N(x; \mu, \Sigma)$ is the probability of observing vector x under a Gaussian distribution with mean μ and covariance matrix Σ. A and B are matrices which are parameters of the model. Various models for the covariance $\Sigma(q_{t-1}, x_t)$ have been proposed: it may be constant, or it may depend on the previous state and the current input. Like the Markov switching models introduced earlier, state space models are more generally expressed functionally:

$$q_t = Aq_{t-1} + Bx_t + v_t,$$

where v_t is a zero-mean Gaussian random variable. Similarly, a Gaussian emission model can be expressed as in equation 10.

The **Kalman filter** [127] is in fact such a model, and the associated algorithms allow to compute $P(q_t | x_1^t, y_1^t)$ in a forward recursion (thus solving the filtering problem). Similarly to Markov switching models, a backward recursion (the Rauch equations [128]) allows to compute the posterior probabilities $P(q_t|x_1^T, y_1^T)$ for $T > t$ (thus solving the smoothing problem).

In the context of real-time control and other applications where learning must be on-line, numerical maximization of the likelihood can be performed recursively with a second-order method which requires only gradients [129]. For off-line applications, the EM algorithm can also be used [130], with a backward pass that is equivalent to the Rauch equations.

7.1 Hybrids of Discrete and Continuous State

One disadvantage of the discrete representation of the state is that it is an inefficient representation in comparison to a distributed representation with multiple state variables. When the state variable can take n values, only $O(logn)$ bits of information about the past of the observation sequence are carried by its value. For example, if instead n binary variables were used, exponentially more bits would be available. Unfortunally, the types of algorithms presented in this paper would also require exponentially more computation, but so-called factorial HMMs [131] have been proposed with such properties, and approximations to the EM algorithm whose cost does not grow exponentially. Models with such a factorial (or distributed) state are very appealing for their expressive power, and there has recently been a lot of research on trying to make computationally efficient learning algorithms for them (see for example [132, 133, 134, 135, 136, 131]).

On the other hand, models with a continuous-valued state have been typically restricted to a linear-Gaussian model, again for reasons of computational tractability of the learning algorithm. In this case, the problem is that the EM algorithm requires computing integrals (rather than sums), which can be done analytically in the Gaussian case, but would have to be done numerically in general.

To model both the abrupt and gradual changes in time series, several researchers have in fact proposed hybrids of state space models and discrete-state HMMs (or IOHMMs), also known as state space models with switching, or jump-linear systems. See [137] and [16] for a review of such models. Many early models assume that some of the parameters of the distribution are known a-priori, and others [15] approximate the EM algorithm with a heuristic, because the E-step would require exponential computations. Others [138, 139] used expensive Monte-Carlo simulations to address this problem. Instead, in [16], a function that is a lower bound on the log likelihood is maximized with a tractable algorithm. This paper uses the idea of variational approximation that has already been proposed in [136] for other intractable models. A simpler version of this idea used in physics is the mean-field approximation [140] for statistical mechanics systems.

Note how this kind of hybrid model again underlines the trade-off that may occur between the choice of a model class that fits well to the data distribution and the efficiency of training such a model. Similar trade-offs (between generality of the model and intractability of the learning algorithm) have been described for variants HMMs and other finite-state learning algorithms in [42, 44, 45].

8 Challenges for Future Research

Hidden Markov models are powerful models of sequential data which have already been successfully used in several applications, notably speech recognition. They could be applied in many other domains. Many extensions and related models have been proposed in recent years, making such models applicable to an even wider range of learning tasks. Many interesting questions remain unanswered, but recent research suggests several promising directions.

- Much research focuses on designing models that better reflect the data, for example trying to remedy the discrepancy between the Markov assumptions (which simplify the mathematics and the algorithms) and the interpretations forced on the state variable (e.g., in speech recognition). In this context, hybrids of HMMs and ANNs and other recent models such as asynchronous Input-Output HMMs are promising but a clear superiority in performance with respect to ordinary HMMs remains to be shown.

- One important other issue that was not yet directly discussed in this paper is that of learning an appropriate representation for the hidden state in Markovian models. In most current applications (such as speech recognition, and econometric applications of IOHMMs and state space models), a lot of prior knowledge must be applied to the definition of what the hidden state represents in order to successfully learn what remains to be learned.

What happens when we try to learn what the hidden state should represent? The state variable keeps some informations about the past sequence and discards others. It therefore captures the temporal dependencies. In [107], it was shown that, for Markovian models (including HMMs, IOHMMs, Markov switching models and Partially Observable Markov Decision Processes), learning of long-term dependencies in sequential data becomes exponentially more difficult as the span of these dependencies increases. However, it was found that this problem is not as bad for conditional models (such as IOHMMs, conditional Markov switching models and Partially Observable Markov Decision Processes) because the state to next-state transformation, being conditioned with extra information, is generally more deterministic.

One promising direction that was proposed to manage this problem is to split the state variable in multiple sub-state variables [131], which may operate at different time scales [141], since the "slow" variables can more easily represent longer-term context. See also related work using Bayesian networks to factor the state variable to represent different types of contexts [55, 56, 131].

- The above models raise the general problem of intractability of the computation of the likelihood (or of the E-Step of the EM algorithm). To address such problems, [136] recently introduced a promising methodology of variational approximation based on tractable substructures in the Bayesian network. This idea was applied to hybrids of continuous and discrete state variables [16].

- Transducers offer a generalization of Markovian models that can be applied to a wide range of learning tasks in which complex a priori structural knowledge about the task is to be smoothly integrated with learning from examples. Local probabilistic assumptions and interpretations of the numbers that are processed by the learning algorithm may be wrong (inconsistent with the data), and the normalization imposed by probabilities may correspond to too strong assumptions about the correct solution. Some of the difficulties inherent in making such probabilistic assumptions and interpretations can be avoided by removing the local probabilistic assumptions and delaying the probabilistic interpretation to the final level of decision.

- The problem of non-stationary time-series is addressed to a certain extent by IOHMMs and Markov switching models, as long as the new regimes in the time series resemble already seen regimes. However, models that can constructively add new states and new distributions (to the extent that the amount of information in the data permits it) would better reflect many time series (such as those studied by econometricians). In this vein, we have briefly mentioned variable-length Markov models (section 6.2) that add more context to the state variable as more training data is encountered. With such constructive algorithms even more than with parametric models, a careful balance between fitting the data and allowing more capacity for representing it must of course be found to avoid overfitting.

- An interesting direction of research, in particular for speech and language processing applications, concerns the higher-level tasks of understanding and man-machine dialogue. Some advocate a complete integration of the recognition task with the understanding and decision-taking modules, to drive the learning with the effect of the actions taken by the machine, using for example methodologies developed in the reinforcement learning community.

Acknowledgements

The author would like to thank Léon Bottou, Patrick Haffner, Yoram Singer, Fernando Pereira, Chris Burges, and Craig Nohl for helpful discussions and useful comments.

References

[1] H. Bourlard and N. Morgan, *Connectionist Speech Recognition. A Hybrid Approach*, vol. 247 of *The Kluwer international series in engineering and computer science*. Boston: Kluwer Academic Publishers, 1993.

[2] Y. Bengio, *Neural Networks for Speech and Sequence Processing*. International Thomson Computer Press, 1996.

[3] P. Baldi and S. Brunak, *Bioinformatics, the Machine Learning Approach*. MIT Press, 1998.

[4] L. Chrisman, "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach," in *Proceedings of the 12th National Conference on Artificial Intelligence*, pp. 183–188, 1992.

[5] T. W. Cacciatore and S. J. Nowlan, "Mixtures of controllers for jump linear and non-linear plants," in *Advances in Neural Information Processing Systems 6* (J. Cowan, G. Tesauro, and J. Alspector, eds.), (San Mateo, CA), Morgan Kaufmann, 1994.

[6] Y. Bengio and P. Frasconi, "An input/output HMM architecture," in *Advances in Neural Information Processing Systems 7* (G. Tesauro, D. Touretzky, and T. Leen, eds.), pp. 427–434, MIT Press, Cambridge, MA, 1995.

[7] M. Meila and M. Jordan, "Learning fine motion by markov mixtures of experts," in *Advances in Neural Information Processing Systems 8* (M. Mozer, D. Touretzky, and M. Perrone, eds.), MIT Press, Cambridge, MA, 1996.

[8] M. Riley and F. Pereira, "Weighted-finite-automata tools with applications to speech and language processing," Tech. Rep. Technical Memorandum 11222-931130-28TM, AT&T Bell Laboratories, 1994.

[9] F. Pereira, M. Riley, and R. Sproat, "Weighted rational transductions and their application to human language processing," in *ARPA Natural Language Processing Workshop*, 1994.

[10] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 20, no. 1, pp. 1–33, 1996.

[11] F. Pereira and M. Riley, "Speech recognition by composition of weighted finite automata," in *Finite-State Language Processing* (E. Roche and Y. Schabes, eds.), pp. 431–453, MIT Press, Cambridge, Massachussetts, 1997.

[12] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: Learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, 1996.

[13] Y. Singer, "Adaptive mixtures of probabilistic transducers," *Neural Computation*, vol. 9, no. 8, 1997.

[14] J. Hamilton, "A new approach to the economic analysis of non-stationary time series and the business cycle," *Econometrica*, vol. 57, pp. 357–384, March 1989.

[15] R. Shumway and D. Stoffer, "Dynamic linear models with switching," *J. Amer. Stat. Assoc.*, vol. 86, pp. 763–769, 1991.

[16] Z. Ghahramani and G. Hinton, "Switching state-space models," Tech. Rep. Technical Report CRG-TR-91-3, University of Toronto, 1996.

[17] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[18] L. E. Baum and J. Eagon, "An inequality with applications to statistical prediction for functions of Markov processes and to a model of ecology," *Bull. Amer. Math. Soc.*, vol. 73, pp. 360–363, 1967.

[19] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Statistic.*, vol. 41, pp. 164–171, 1970.

[20] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process," *Inequalities*, vol. 3, pp. 1–8, 1972.

[21] J. Baker, "Stochastic modeling for automatic speech understanding," in *Speech Recognition* (D. Reddy, ed.), pp. 521–542, New York: Academic Press, 1975.

[22] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, vol. 64, pp. 532–556, 1976.

[23] S. Levinson, L. Rabiner, and M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, vol. 64, no. 4, pp. 1035–1074, 1983.

[24] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 257–285, January 1986.

[25] A. Waibel and K. Lee, *Readings in Speech Recognition.* Morgan Kaufmann, 1990.

[26] R. Nag, K. Wong, and F. Fallside, "Script recognition using hidden Markov models," in *International Conference on Acoustics, Speech and Signal Processing*, (Tokyo), pp. 2071–2074, 1986.

[27] A. Kundu and L. Bahl, "Recognition of handwritten script: a hidden Markov model based approach," in *International Conference on Acoustics, Speech and Signal Processing*, (New-York, NY), pp. 928–931, 1988.

[28] O. Matan, C. Burges, Y. LeCun, and J. Denker, "Multi-digit recognition using a space displacement neural network," in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lipmann, eds.), (San Mateo CA), pp. 488–495, Morgan Kaufmann, 1992.

[29] J. Ha, S. Oh, J. Kim, and Y. Kwon, "Unconstrained handwritten word recognition with interconnected hidden Markov models," in *Third International Workshop on Frontiers in Handwriting Recognition*, (Buffalo), pp. 455–460, IAPR, May 1993.

[30] M. Schenkel, H. Weissman, I. Guyon, C. Nohl, and D. Henderson, "Recognition-based segmentation of on-line hand-printed words," in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), (Denver, CO), pp. 723–730, 1993.

[31] M. Schenkel, I. Guyon, and D. Henderson, "On-line cursive script recognition using time delay neural networks and hidden Markov models," *Machine Vision and Applications*, pp. 215–223, 1995.

[32] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "Lerec: A NN/HMM hybrid for on-line handwriting recognition," *Neural Computation*, vol. 7, no. 5, pp. 1289–1303, 1995.

[33] A. Krogh, M. Brown, I. Mian, K. Sjlander, and D. Haussler, "Hidden markov models in computational biology: Applications to protein modeling," *Journal Molecular Biology*, vol. 235, pp. 1501–1531, 1994.

[34] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure, "Hidden markov models of biological primary sequence information," *Proc. Nat. Acad. Sci. (USA)*, vol. 91, no. 3, pp. 1059–1063, 1995.

[35] Y. Chauvin and P. Baldi, "Hidden markov models of the g-protein-coupled receptor family," *Journal of Computational Biology*, vol. to appear, 1995.

[36] K. Karplus, K. Sjolander, C. Barrett, M. Cline, D. Haussler, R. Hughey, L. Holm, and C. Sander, "Predicting protein structure using hidden markov models," *Proteins: Structure, Function and Genetics*, vol. Supp. 1, no. 1, pp. 134–139, 1997.

[37] P. Smyth, “Hidden markov models for fault detection in dynamic systems,” *Pattern Recognition*, vol. 27, no. 1, pp. 149–164, 1994.

[38] I. Guyon and F. Pereira, “Design of a linguistic postprocessor using variable memory length Markov models,” in *International Conference on Document Analysis and Recognition*, (Montreal, Canada), pp. 454–457, IEEE Computer Society Press, August 1995.

[39] I. Guyon, M. Schenkel, and J. Denker, “Overview and synthesis of on-line cursive handwriting recognition techniques,” in *Handbook on Optical Character Recognition and Document Image Analysis* (P. Wang and H. Bunke, eds.), World Scientific, 1996.

[40] R. Garcia and P. Perron, “An analysis of the real interest rate under regime shift,” *The Review of Economics and Statistics*, 1996.

[41] Y. Bengio, V.-P. Lauzon, and R. Ducharme, “Experiments on the application of IOHMMs to model financial returns series,” Tech. Rep. 1146, Département d’informatique et recherche opérationnelle, Université de Montréal, 1999.

[42] N. Abe and M. Warmuth, “On the computational complexity of approximating distributions by probabilistic automata,” *Machine Learning*, vol. 9, July 1992.

[43] L. G. Valiant, “A theory of the learnable,” *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[44] D. Gillman and M. Sipser, “Inference and minimization of hidden marko chains,” in *proceedings of the 7th annual ACM conference on Computational learning theory (COLT’94)*, pp. 147–158, ACM, 1994.

[45] N. T. Dana Ron, Yoram Singer, “On the learnability and usage of acyclic probabilistic finite automata,” *Journal of Computer and System Sciences*, vol. 56, no. 2, pp. 133–152, 1998.

[46] V. Vapnik, *The Nature of Statistical Learning Theory.* New-York: Springer, 1995.

[47] A. Markov, “An example of statistical investigation in the text of ‘eugene onyegin’ illustrating coupling of ‘tests’ in chains,” *Proceedings of the Academy of Science, St. Petersburg*, vol. 7, pp. 153–162, 1913.

[48] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[49] D. Spiegelhalter, A. Dawid, S. Lauritzen, and R. Cowell, “Bayesian analysis in expert systems,” *Statistical Science*, vol. 8, pp. 219–283, 1993.

[50] W. Buntine, “Operations for learning with graphical models,” *Journal of Artificial Intelligence Research*, vol. 2, pp. 159–225, 1994.

[51] D. Heckerman, “A tutorial on learning with bayesian networks,” Tech. Rep. TR-95-06, Microsoft Research, ftp://ftp.research.microsoft.com/pub/Tech-Reports/Winter94-95/TR-95-06.PS, January 1996.

[52] P. Smyth, D. Heckerman, and M. Jordan, “Probabilistic independence networks for hidden markov probability models,” *Neural Computation*, vol. 9, no. 2, pp. 227–269, 1997.

[53] P. Smyth, “Belief networks, hidden markov models, and markov random fields: a unifying view,” *Pattern Recognition Letters*, 1998.

[54] D. MacKay, R. McEliece, and J.-F. C. (in press), “Turbo-decoding as an instance of pearl’s belief propagation algorithm,” *IEEE Journal on Selected Areas in Communications*, 1998.

[55] G. Zweig and S. Russel, “Speech recognition with dynamic Bayesian networks,” in *Proceedings of the AAAI Conference*, (Madison, Wisconsin), AAAI Press, 1998.

[56] G. Zweig and S. Russel, “Probabilistic modeling with Bayesian networks for ASR,” in *Proceedings of the International Conference on Statistical Language Processing*, (Sidney, Australia), 1998.

[57] R. Gray, “Vector quantization,” *IEEE ASSP Magazine*, pp. 4–29, April 1984.

[58] K.-F. Lee, *Automatic Speech Recognition: the development of the SPHINX system.* Kluwer Academic Publ., 1989.

[59] X. Huang, K.-F. Lee, and H.-W. Hon, “On semi-continuous hidden Markov modeling,” in *International Conference on Acoustics, Speech and Signal Processing*, pp. 689–692, 1990.

[60] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition.* Edinburgh: University Press, 1990.

[61] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum-likelihood from incomplete data via the EM algorithm,” *Journal of Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.

[62] S. L. Lauritzen, “The EM algorithm for graphical association models with missing data,” *Computational Statistics and Data Analysis*, vol. 19, pp. 191–201, 1995.

[63] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, pp. 260–269, 1967.

[64] R. Bellman, *Dynamic Programming.* NJ: Princeton University Press, 1957.

[65] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

[66] H. Ney, D. Mergel, A. Noll, and A. Paesler, “Data driven search organization for continuous speech recognition,” *IEEE Transactions on Signal Processing*, vol. 40, pp. 272–281, February 1992.

[67] F. Alleva, X. Huang, and M. Hwang, “An improved search algorithm using incremental knowledge for continuous speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing*, (Minneapolis, Minnesota), pp. 307–310, 1993.

[68] H. Murveit, J. Butzberger, V. Digilakis, and M. Weintraub, “Large-vocabulary dictation using SRI’s DECIPHER speech recognition system: Progressive search techniques knowledge for continuous speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing*, (Minneapolis, Minnesota), pp. 319–322, 1993.

[69] X. Aubert, C. Dugast, H. Ney, and V. Steinbiss, “Large vocabulary continuous speech recognition of Wall Street journal data,” in *International Conference on Acoustics, Speech and Signal Processing*, (Adelaide, Australia), pp. 129–132, 1994.

[70] F. Kubala, A. Anastasakos, J. Makhoul, L. Nguyen, R. Schwartz, and G. Zavaliagkos, “Comparative experiments on large vocabulary speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing*, (Adelaide, Australia), pp. 561–564, 1994.

[71] F. Jelinek, *Statistical Methods for Speech Recognition.* Cambridge, Massachussetts: MIT Press, 1998.

[72] R. Cole, J. Mariani, H. Uszkoriet, A. Zaenen, and V. Zue, *Survey of the State of the Art in Human Language Technology*
. http://www.cse.ogi.edu/CSLU/HLTsurvey/HLTsurvey.html: Cambridge University Press, 1996.

[73] P. Smyth, “Clustering sequences with hidden markov models,” in *Advances in Neural Information Processing 9* (M. Mozer, M. Jordan, and T. Petsche, eds.), MIT Press, 1997.

[74] A. Stolcke and S. Omohundro, “Hidden Markov model induction by bayesian model merging,” in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), (San Mateo, CA), pp. 11–18, Morgan Kaufmann, 1993.

[75] A. Stolcke and S. Omohundro, “Best-first model merging for hidden Markov model induction,” Tech. Rep. TR-94-003, International Computer Science Institute, Berkeley, CA, January 1994.

[76] R. Carrasco and J. Oncina, “Learning regular grammars by means of a state merging method,” in *Grammatical Inference and Applications Proc. of the 2nd International Colloquium on Grammatical Inference ICGI94*, (Alicante (Spain)), Lecture Notes in Artificial Intelligence 862, September 1994.

[77] P. Brown, *The Acoustic-Modeling problem in Automatic Speech Recognition.* PhD thesis, Dept. of Computer Science, Carnegie-Mellon University, 1987.

[78] L. R. Bahl, P. Brown, P. V. de Souza, and R. L. Mercer, “Speech recognition with continuous-parameter hidden Markov models,” *Computer, Speech and Language*, vol. 2, pp. 219–234, 1987.

[79] A. Nadas, D. Nahamoo, and M. Picheny, “On a model-robust training method for speech recognition,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-36, no. 9, pp. 1432–1436, 1988.

[80] B. Juang and S. Katagiri, “Discriminative learning for minimum error classification,” *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043–3054, 1992.

[81] H. Leprieur and P. Haffner, “Discriminant learning with minimum memory loss for improved non-vocabulary rejection,” in *EUROSPEECH’95*, (Madrid, Spain), 1995.

[82] Advanced Research Projects Agency, *Proceedings of the 1994 ARPA Human Language Technology Workshop (Princeton, New Jersey, March 1994).* Morgan Kaufmann, 1994.

[83] S. Levinson, “Statistical modeling and classification,” in *Survey of the State of the Art in Human Language Technology* (R. Cole, J. Mariani, H. Uszkoriet, A. Zaenen, and V. Zue, eds.), pp. 395–401, http://www.cse.ogi.edu/CSLU/HLTsurvey/HLTsurvey.html: Cambridge University Press, 1996.

[84] H. Sakoe and C. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP–26, pp. 43–49, February 1978.

[85] D. Rumelhart, G. Hinton, and R. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing* (D. Rumelhart and J. McClelland, eds.), vol. 1, ch. 8, pp. 318–362, Cambridge: MIT Press, 1986.

[86] R. P. Lippmann and B. Gold, “Neural classifiers useful for speech recognition,” in *IEEE Proc. First Intl. Conf. on Neural Networks*, vol. IV, (San Diego, CA), pp. 417–422, 1987.

[87] H. Bourlard and C. Wellekens, “Links between hidden Markov models and multilayer perceptrons,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 1167–1178, 1990.

[88] N. Morgan and H. Bourlard, “Continuous speech recognition using multilayer perceptrons with hidden Markov models,” in *International Conference on Acoustics, Speech and Signal Processing*, (Albuquerque, NM), pp. 413–416, 1990.

[89] H. Bourlard and C. Wellekens, “Speech pattern discrimination and multi-layered perceptrons,” *Computer Speech and Language*, vol. 3, pp. 1–19, 1989.

[90] M. Franzini, K. Lee, and A. Waibel, "Connectionist Viterbi training: a new hybrid method for continuous speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, (Albuquerque, NM), pp. 425–428, 1990.

[91] A. J. Robinson and F. Fallside, "A recurrent error propagation network speech recognition system," *Computer Speech and Language*, vol. 5, pp. 259–274, July 1991.

[92] X. Driancourt, L. Bottou, and P. Gallinari, "Learning vector quantization, multi-layer perceptron and dynamic programming: Comparison and cooperation," in *International Joint Conference on Neural Networks*, vol. 2, pp. 815–819, 1991.

[93] P. Haffner, M. Franzini, and A. Waibel, "Integrating time alignment and neural networks for high performance continuous speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, (Toronto), pp. 105–108, 1991.

[94] J. Tebelskis, A. Waibel, B. Petek, and O. Schmidbauer, "Continuous speech recognition using linked predictive networks," in *Advances in Neural Information Processing Systems 3* (R. P. Lippman, R. Moody, and D. S. Touretzky, eds.), (Denver, CO), pp. 199–205, Morgan Kaufmann, San Mateo, 1991.

[95] L. Bottou, F. Fogelman-Soulié, P. Blanchet, and J. S. Lienard, "Speaker independent isolated digit recognition: multilayer perceptrons vs dynamic time warping," *Neural Networks*, vol. 3, pp. 453–465, 1990.

[96] E. Levin, R. Pieraccini, and E. Bocchieri, "Time-warping network: a hybrid framework for speech recognition," in *Advances in Neural Information Processing Systems 4* (J. Moody, S. Hanson, and R. Lipmann, eds.), (Denver, CO), pp. 151–158, 1992.

[97] G. Zavaliagkos, S. Austin, J. Makhoul, and R. Schwartz, "A hybrid continuous speech recognition system using segmental neural nets with hidden Markov models," *Int. Journal of Pattern Recognition and Artificial Intelligence*, pp. 305–319, 1993. Special Issue on Applications of Neural Networks to Pattern Recognition (I. Guyon Ed.).

[98] J. Bridle, "Alphanets: a recurrent 'neural' network architecture with a hidden Markov model interpretation," *Speech Communication*, vol. 9, no. 1, pp. 83–92, 1990.

[99] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems 2* (D. Touretzky, ed.), pp. 211–217, Morgan Kaufmann, 1990.

[100] N. Morgan, Y. Konig, S. Wu, and H. Bourlard, "Transition-based statistical training for ASR," in *Proceedings of IEEE Automatic Speech Recognition Workshop (Snowbird)*, pp. 133–134, 1995.

[101] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[102] L. Bottou, *Une approche théorique de l'apprentissage connexioniste; applications à la reconnaissance de la parole.* PhD thesis, Université de Paris XI, 1991.

[103] Y. Bengio, *Artificial Neural Networks and their Application to Sequence Recognition.* PhD thesis, McGill University, (Computer Science), Montreal, Qc., Canada, 1991.

[104] E. Sondik, "The optimal control of partially observable markov processes over the finite horizon," *Operations Research*, vol. 11, pp. 1071–1088, 1973.

[105] E. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: discounted case," *Operations Research*, vol. 26, pp. 282–304, 1978.

[106] P. Frasconi, M. Gori, and A. Sperduti, "On the efficient classification of data structures by neural networks," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 1997.

[107] Y. Bengio and P. Frasconi, "Diffusion of context and credit information in markovian models," *Journal of Artificial Intelligence Research*, vol. 3, pp. 223–244, 1995.

[108] S. Goldfeld and R. Quandt, "A markov model for switching regressions," *Journal of Econometrics*, vol. 1, pp. 3–16, 1973.

[109] R. Shumway and D. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[110] S. Cosslett and L.-F. Lee, "Serial correlation in discrete variable models," *Journal of Econometrics*, vol. 27, pp. 79–97, 1985.

[111] J. Hamilton, "Analysis of time series subject to changes in regime," *Journal of Econometrics*, vol. 45, pp. 39–70, 1990.

[112] M. Bonomo and R. Garcia, "Can a well-fitted equilibrium asset-pricing model produce mean reversion?," *Journal of Applied Econometrics*, vol. 9, pp. 19–29, 1994.

[113] M. Sola and J. Driffill, "Testing the term structure of interest rates using a stationary vector autoregression with regime switching," *Journal of Economic Dynamics and Control*, vol. 18, pp. 601–628, 1994.

[114] J. Hamilton and R. Susmel, "Autoregressive conditional heteroskedasticity and changes in regime," *Journal of Econometrics*, vol. 64, no. 1-2, pp. 307–33, 1994.

[115] J. Cai, "A markov model of unconditional variance in ARCH," *Journal of Business and Economic Statistics*, 1994.

[116] D. Sichel, "Business cycle duration dependence: a parametric approach," *Review of Economics and Statistics*, vol. 71, pp. 245–260, 1991.

[117] F. Diebold, G. Rudebusch, and E. Sichel, "Further evidence on business-cycle duration dependence," in *Business Cycles, Indicators, and Forecasting* (J. Stock and M. Watson, eds.), Chicago: University of Chicago Press, 1993.

[118] E. Ghysel, "A time series model with periodic stochastic regime switching," Tech. Rep. C.R.D.E. Discussion paper 1093, C.R.D.E., Universite de Montreal, Montreal, Quebec, Canada, 1993.

[119] R. Garcia and H. Schaller, "Are the effects of monetary policy asymmetric," Tech. Rep. 95s-6, CIRANO, Montreal, Quebec, Canada, 1995.

[120] F. Diebold, J. Lee, and G. Weinbach, "Regime switching with time-varying transition probabilities," in *Nonstationary Time Series Analysis and Cointegration* (C. Hargreaves, ed.), Oxford: Oxford University Press, 1993.

[121] N. Kiefer, "A note on switching regressions and logistic discrimination," *Econometrica*, vol. 48, pp. 1065–1069, 1980.

[122] C. Kim, "Dynamical linear models with markov-switching," *Journal of Econometrics*, vol. 60, pp. 1–22, 1994.

[123] J. Hamilton, "State-space models," in *Handbook of Econometrics* (R. Engle and D. McFadden, eds.), North Holland, New York, 1993.

[124] S. Bengio and Y. Bengio, "An EM algorithm for asynchronous input/output hidden Markov models," in *International Conference On Neural Information Processing* (L. Xu, ed.), (Hong-Kong), pp. 328–334, 1996.

[125] L. Bottou, Y. Bengio, and Y. Le Cun, "Document analysis with generalized transduction," Tech. Rep. HA6156000-960701-01TM, AT&T Laboratories, Holmdel, New-Jersey, July 1996.

[126] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[127] R. Kalman and R. Bucy, "New results in linear filtering and prediction," *Journal of Basic Engineering (ASME)*, vol. 83D, pp. 95–108, 1961.

[128] H. Rauch, "Solutions to the linear smoothing problem," *IEEE Transactions on Automatic Control*, vol. 8, pp. 371–372, 1963.

[129] L. Ljung and T. Soderstrom, *Theory and Practice of recursive identification.* MIT Press, 1983.

[130] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," Tech. Rep. Technical Report CRG-TR-91-1, University of Toronto, 1996.

[131] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," in *Advances in Neural Information Processing Systems 8* (M. Mozer, D. Touretzky, and M. Perrone, eds.), MIT Press, Cambridge, MA, 1996.

[132] R. Neal, "Connectionist learning of belief networks," *Artificial Intelligence*, vol. 56, pp. 71–113, 1992.

[133] G. Hinton, P. Dayan, B. Frey, and R. Neal, "The wake-sleep algorithm for unsupervised neural networks," *Science*, vol. 268, pp. 1558–1161, 1995.

[134] P. Dayan, G. Hinton, R. Neal, and R. Zemel, "The Helmholtz machine," *Neural Computation*, vol. 7, pp. 889–904, 1995.

[135] L. Saul, T. Jaakola, and M. Jordan, "Mean field theory for sigmoid belief networks," *Journal of Artificial Intelligence Research*, vol. 4, pp. 61–76, 1996.

[136] L. Saul and M. Jordan, "Exploiting tractable substructures in intractable networks," in *Advances in Neural Information Processing Systems 8* (M. Mozer, D. Touretzky, and M. Perrone, eds.), MIT Press, Cambridge, MA, 1996.

[137] Y. Bar-Shalom and X. Li, *Estimation and Tracking.* Boston, MA: Artech House, 1993.

[138] C. Carter and R. Kohn, "On gibbs sampling for state space models," *Biometrika*, vol. 81, pp. 541–553, 1994.

[139] C. Athaide, *Likelihood estimation and state estimation for nonlinear state space models.* PhD thesis, Graduate Group in Managerial Science and Applied Economics, University of Pennsylvania, Philadelphia, PA, 1995.

[140] G. Parisi, *Statistical Field Theory.* Redwood City, CA: Addison-Wesley, 1988.

[141] S. ElHihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Advances in Neural Information Processing Systems 8* (M. Mozer, D. Touretzky, and M. Perrone, eds.), pp. 493–499, MIT Press, Cambridge, MA, 1996.