LETTER **Communicated by Christopher Williams**

Hidden Neural Networks

Anders Krogh
Center for Biological Sequence Analysis, Building 208, Technical University of Denmark,
2800 Lyngby, Denmark

Søren Kamaric Riis
Department of Mathematical Modeling, Section for Digital Signal Processing, Technical University of Denmark, Building 321, 2800 Lyngby, Denmark

A general framework for hybrids of hidden Markov models (HMMs) and neural networks (NNs) called hidden neural networks (HNNs) is described. The article begins by reviewing standard HMMs and estimation by conditional maximum likelihood, which is used by the HNN. In the HNN, the usual HMM probability parameters are replaced by the outputs of state-specific neural networks. As opposed to many other hybrids, the HNN is normalized globally and therefore has a valid probabilistic interpretation. All parameters in the HNN are estimated simultaneously according to the discriminative conditional maximum likelihood criterion. The HNN can be viewed as an undirected probabilistic independence network (a graphical model), where the neural networks provide a compact representation of the clique functions. An evaluation of the HNN on the task of recognizing broad phoneme classes in the TIMIT database shows clear performance gains compared to standard HMMs tested on the same task.

1 Introduction

Hidden Markov models (HMMs) is one of the most successful modeling approaches for acoustic events in speech recognition (Rabiner, 1989; Juang & Rabiner, 1991), and more recently they have proved useful for several problems in biological sequence analysis like protein modeling and gene finding (see, e.g., Durbin, Eddy, Krogh, & Mitchison, 1998; Eddy, 1996; Krogh, Brown, Mian, Sjölander, & Haussler, 1994). Although the HMM is good at capturing the temporal nature of processes such as speech, it has a very limited capacity for recognizing complex patterns involving more than first-order dependencies in the observed data. This is due to the first-order state process and the assumption of state-conditional independence of observations. Multilayer perceptrons are almost the opposite: they cannot model temporal phenomena very well but are good at recognizing complex pat-

terns. Combining the two frameworks in a sensible way can therefore lead to a more powerful model with better classification abilities.

The starting point for this work is the so-called class HMM (CHMM), which is basically a standard HMM with a distribution over classes assigned to each state (Krogh, 1994). The CHMM incorporates conditional maximum likelihood (CML) estimation (Juang & Rabiner, 1991; Nádas, 1983; Nádas, Nahamoo, & Picheny, 1988). In contrast to the widely used maximum likelihood (ML) estimation, CML estimation is a discriminative training algorithm that aims at maximizing the ability of the model to discriminate between different classes. The CHMM can be normalized globally, which allows for nonnormalizing parameters in the individual states, and this enables us to generalize the CHMM to incorporate neural networks in a valid probabilistic way.

In the CHMM/NN hybrid, which we call a hidden neural network (HNN), some or all CHMM probability parameters are replaced by the outputs of state-specific neural networks that take the observations as input. The model can be trained as a whole from observation sequences with labels by a gradient-descent algorithm. It turns out that in this algorithm, the neural networks are updated by standard backpropagation, where the errors are calculated by a slightly modified forward-backward algorithm.

In this article, we first give a short introduction to standard HMMs. The CHMM and conditional ML are then introduced, and a gradient descent algorithm is derived for estimation. Based on this, the HNN is described next along with training issues for this model, and finally we give a comparison to other hybrid models. The article concludes with an evaluation of the HNN on the recognition of five broad phoneme classes in the TIMIT database (Garofolo et al., 1993). Results on this task clearly show a better performance of the HNN compared to a standard HMM.

2 Hidden Markov Models

To establish notation and set the stage for describing CHMMs and HNNs, we start with a brief overview of standard hidden Markov models. (For a more comprehensive introduction, see Rabiner, 1989; Juang & Rabiner, 1991.) In this description we consider discrete first-order HMMs, where the observations are symbols from a finite alphabet $\mathcal{A}$. The treatment of continuous observations is very similar (see, e.g., Rabiner, 1989).

The standard HMM is characterized by a set of N states and two concurrent stochastic processes: a first-order Markov process between states modeling the temporal structure of the data and an emission process for each state modeling the locally stationary part of the data. The state process is given by a set of transition probabilities, θ_{ij}, giving the probability of making a transition from state i to state j, and the emission process in state i is described by the probabilities, $\phi_i(a)$, of emitting symbol $a \in \mathcal{A}$ in state i. The ϕ's are usually called emission probabilities, but we use the term

match probabilities here. We observe only the sequence of outputs from the model, and not the underlying (hidden) state sequence, hence the name *hidden* Markov model. The set Θ of all transition and emission probabilities completely specifies the model.

Given an HMM, the probability of an observation sequence, $x = x_1, \ldots, x_L$, of L symbols from the alphabet $\mathcal{A}$ is defined by

$$P(x|\Theta) = \sum_{\pi} P(x, \pi\,|\Theta) = \sum_{\pi} \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l} \phi_{\pi_l}(x_l). \tag{2.1}$$

Here $\pi = \pi_1, \ldots, \pi_L$ is a state sequence; π_i is the number of the ith state in the sequence. Such a state sequence is called a *path* through the model. An auxiliary start state, $\pi_0 = 0$, has been introduced such that θ_{0i} denotes the probability of starting a path in state i. In the following we assume that state N is an end state: a nonmatching state with no outgoing transitions.

The probability 2.1 can be calculated efficiently by a dynamic programming-like algorithm known as the forward algorithm. Let $\alpha_i(l) = P(x_1, \ldots, x_l, \pi_l = i \mid \Theta)$, that is, the probability of having matched observations $x_1, \ldots, x_l$ and being in state i at time l. Then the following recursion holds for $1 \leq i \leq N$ and $1 < l \leq L$,

$$\alpha_i(l) = \phi_i(x_l) \sum_j \alpha_j(l-1)\theta_{ji}, \tag{2.2}$$

and $P(x|\Theta) = \alpha_N(L)$. The recursion is initialized by $\alpha_i(1) = \theta_{0i}\phi_i(x_1)$ for $1 \leq i \leq N$.

The parameters of the model can be estimated from data by an ML method. If multiple sequences of observations are available for training, they are assumed independent, and the total likelihood of the model is just a product of probabilities of the form 2.1 for each of the sequences. The generalization from one to many observation sequences is therefore trivial, and we will consider only one training sequence in the following. The likelihood of the model, $P(x|\Theta)$, given in equation 2.1, is commonly maximized by the Baum-Welch algorithm, which is an expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) guaranteed to converge to a local maximum of the likelihood. The Baum-Welch algorithm iteratively reestimates the model parameters until convergence, and for the transition probabilities the reestimation formulas are given by

$$\theta_{ij} \leftarrow \frac{\sum_l n_{ij}(l)}{\sum_{j'l'} n_{ij'}(l')} = \frac{n_{ij}}{\sum_{j'} n_{ij'}}, \tag{2.3}$$

where $n_{ij}(l) = P(\pi_{l-1} = i, \pi_l = j \mid x, \Theta)$ is the expected number of times a transition from state i to state j is used at time l. The reestimation equations

for the match probabilities can be expressed in a similar way by defining $n_i(l) = P(\pi_l = i \mid x, \Theta)$ as the expected number of times we are in state i at time l. Then the reestimation equations for the match probabilities are given by

$$\phi_i(a) \leftarrow \frac{\sum_l n_i(l)\delta_{x_l,a}}{\sum_{la'} n_i(l)\delta_{x_l,a'}} = \frac{n_i(a)}{\sum_{a'} n_i(a')} . \tag{2.4}$$

The expected counts can be computed efficiently by the forward-backward algorithm. In addition to the forward recursion, a similar recursion for the backward variable $\beta_i(l)$ is introduced. Let $\beta_i(l) = P(x_{l+1}, \ldots, x_L \mid \pi_l = i, \Theta)$, that is, the probability of matching the rest of the sequence $x_{l+1}, \ldots, x_L$ given that we are in state i at time l. After initializing by $\beta_N(L) = 1$, the recursion runs from $l = L - 1$ to $l = 1$ as

$$\beta_i(l) = \sum_{j=1}^{N} \theta_{ij}\beta_j(l+1)\phi_j(x_{l+1}), \tag{2.5}$$

for all states $1 \le i \le N$. Using the forward and backward variables, $n_{ij}(l)$ and $n_i(l)$ can easily be computed:

$$n_{ij}(l) = P(\pi_{l-1} = i, \pi_l = j \mid x, \Theta) = \frac{\alpha_i(l-1)\theta_{ij}\phi_j(x_l)\beta_j(l)}{P(x|\Theta)} \tag{2.6}$$

$$n_i(l) = P(\pi_l = i \mid x, \Theta) = \frac{\alpha_i(l)\beta_i(l)}{P(x|\Theta)} . \tag{2.7}$$

2.1 Discriminative Training. In many problems, the aim is to predict what class an input belongs to or what sequence of classes it represents. In continuous speech recognition, for instance, the object is to predict the sequence of words or phonemes for a speech signal. To achieve this, a (sub)model for each class is usually estimated by ML independent of all other models and using only the data belonging to this class. This procedure maximizes the ability of the model to reproduce the observations in each class and can be expressed as

$$\hat{\Theta}^{\mathrm{ML}} = \operatorname*{argmax}_{\Theta} P(x, y|\Theta) = \operatorname*{argmax}_{\Theta} [P(x|\Theta_y)P(y|\Theta)], \tag{2.8}$$

where y is the class or sequence of class labels corresponding to the observation sequence x and Θ_y is the model for class y or a concatenation of submodels corresponding to the observed labels. In speech recognition, $P(x|\Theta_y)$ is often denoted the acoustic model probability, and the language model probability $P(y|\Theta)$ is usually assumed constant during training of the acoustic models. If the true source producing the data is contained in

the model space, ML estimation based on an infinite training set can give the optimal parameters for classification (Nádas et al., 1988; Nádas, 1983), provided that the global maximum of the likelihood can be reached. However, in any real-world application, it is highly unlikely that the true source is contained in the space of HMMs, and the training data are indeed limited. This is the motivation for using discriminative training.

To accommodate discriminative training, we use one big model and assign a label to each state; all the states that are supposed to describe a certain class C are assigned label C. A state can also have a probability distribution $\psi_i(c)$ over labels, so that several labels are possible with different probabilities. This is discussed in Krogh (1994) and Riis (1998a), and it is somewhat similar to the input / output HMM (IOHMM) (Bengio & Frasconi, 1996). For brevity, however, we here limit ourselves to consider only one label for each state, which we believe is the most interesting for many applications. Because each state has a class label or a distribution over class labels, this sort of model was called a class HMM (CHMM) in Krogh (1994).

In the CHMM, the objective is to predict the labels associated with x, and instead of ML estimation, we therefore choose to maximize the probability of the correct labeling,

$$\hat{\Theta}^{\mathrm{CML}} = \operatorname*{argmax}_{\Theta} P(y|x, \Theta) = \operatorname*{argmax}_{\Theta} \frac{P(x, y|\Theta)}{P(x|\Theta)}, \tag{2.9}$$

which is also called conditional maximum likelihood (CML) estimation (Nádas, 1983). If the language model is assumed constant during training, CML estimation is equivalent to maximum mutual information estimation (Bahl, Brown, de Souza, & Mercer, 1986).

From equation 2.9, we observe that computing the probability of the labeling requires computation of (1) the probability $P(x, y|\Theta)$ in the clamped phase and (2) the probability $P(x|\Theta)$ in the free-running phase. The term *free running* means that the labels are not taken into account, so this phase is similar to the decoding phase, where we wish to find the labels for an observation sequence. The constraint by the labels during training gives rise to the name *clamped phase*; this terminology is borrowed from the Boltzmann machine literature (Ackley, Hinton, & Sejnowski, 1985; Bridle, 1990). Thus, CML estimation adjusts the model parameters so as to make the free-running recognition model as close as possible to the clamped model. The probability in the free-running phase is computed using the forward algorithm described for standard HMMs, whereas the probability in the clamped phase is computed by considering only paths $\mathcal{C}(y)$ that are consistent with the observed labeling,

$$P(x, y|\Theta) = \sum_{\pi \in \mathcal{C}(y)} P(x, \pi|\Theta). \tag{2.10}$$

This quantity can be calculated by a variant of the forward algorithm to be discussed below.

Unfortunately the Baum-Welch algorithm is not applicable to CML estimation (see, e.g., Gopalakrishnan, Kanevsky, Nádas, & Nahamoo, 1991). Instead, one can use a gradient-descent-based approach, which is also applicable to the HNNs discussed later. To calculate the gradients, we switch to the negative log-likelihood, and define

$$\mathcal{L} = -\log P(y|x,\Theta) = \mathcal{L}_c - \mathcal{L}_f \tag{2.11}$$

$$\mathcal{L}_c = -\log P(x,y|\Theta) \tag{2.12}$$

$$\mathcal{L}_f = -\log P(x|\Theta). \tag{2.13}$$

The derivative of $\mathcal{L}_f$ for the free-running model with regard to a generic parameter $\omega \in \Theta$ can be expressed as,

$$\begin{aligned}\frac{\partial \mathcal{L}_f}{\partial \omega} &= -\frac{1}{P(x|\Theta)}\frac{\partial P(x|\Theta)}{\partial \omega} \\ &= -\sum_\pi \frac{1}{P(x|\Theta)}\frac{\partial P(x,\pi|\Theta)}{\partial \omega} \\ &= -\sum_\pi \frac{P(x,\pi|\Theta)}{P(x|\Theta)}\frac{\partial \log P(x,\pi|\Theta)}{\partial \omega} \\ &= -\sum_\pi P(\pi|x,\Theta)\frac{\partial \log P(x,\pi|\Theta)}{\partial \omega}\,. \end{aligned} \tag{2.14}$$

This gradient is an expectation over all paths of the derivative of the complete data log-likelihood $\log P(x,\pi|\Theta)$. Using equation 2.1, this becomes

$$\frac{\partial \mathcal{L}_f}{\partial \omega} = -\sum_{l,i}\frac{n_i(l)}{\phi_i(x_l)}\frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{l,i,j}\frac{n_{ij}(l)}{\theta_{ij}}\frac{\partial \theta_{ij}}{\partial \omega}\,. \tag{2.15}$$

The gradient of the negative log-likelihood $\mathcal{L}_c$ in the clamped phase is computed similarly, but the expectation is taken only for the allowed paths $\mathcal{C}(y)$,

$$\frac{\partial \mathcal{L}_c}{\partial \omega} = -\sum_{l,i}\frac{m_i(l)}{\phi_i(x_l)}\frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{l,i,j}\frac{m_{ij}(l)}{\theta_{ij}}\frac{\partial \theta_{ij}}{\partial \omega}\,, \tag{2.16}$$

where $m_{ij}(l) = P(\pi_{l-1} = i, \pi_l = j \,|\, x, y, \Theta)$ is the expected number of times a transition from state i to state j is used at time l for the allowed paths. Similarly, $m_i(l) = P(\pi_l = i \,|\, x, y, \Theta)$ is the expected number of times we are in state i at time l for the allowed paths. These counts can be computed using the modified forward-backward algorithm, discussed below.

For a standard model, the derivatives in equations 2.15 and 2.16 are simple. When ω is a transition probability, we obtain

$$\frac{\partial \mathcal{L}}{\partial \theta_{ij}} = -\frac{m_{ij} - n_{ij}}{\theta_{ij}} . \tag{2.17}$$

The derivative $\frac{\partial \mathcal{L}}{\partial \phi_i(a)}$ is of exactly the same form, except that m_{ij} and n_{ij} are replaced by $m_i(a)$ and $n_i(a)$, and θ_{ij} by $\phi_i(a)$.

When minimizing $\mathcal{L}$ by gradient descent, it must be ensured that the probability parameters remain positive and properly normalized. Here we use the same method as Bridle (1990) and Baldi and Chauvin (1994) and do gradient descent in another set of unconstrained variables. For the transition probabilities, we define

$$\theta_{ij} = \frac{e^{z_{ij}}}{\sum_{j'} e^{z_{ij'}}} , \tag{2.18}$$

where z_{ij} are the new unconstrained auxiliary variables, and θ_{ij} always sum to one by construction. Gradient descent in the z's by $z_{ij} \leftarrow z_{ij} - \eta \frac{\partial \mathcal{L}}{\partial z_{ij}}$ yields a change in θ given by

$$\theta_{ij} \leftarrow \frac{\theta_{ij} \exp(-\eta \frac{\partial \mathcal{L}}{\partial z_{ij}})}{\sum_{j'} \theta_{ij'} \exp(-\eta \frac{\partial \mathcal{L}}{\partial z_{ij'}})} . \tag{2.19}$$

The gradients with respect to z_{ij} can be expressed entirely in terms of θ_{ij} and $m_{ij} - n_{ij}$,

$$\frac{\partial \mathcal{L}}{\partial z_{ij}} = -[m_{ij} - n_{ij} - \theta_{ij} \sum_{j'} (m_{ij'} - n_{ij'})], \tag{2.20}$$

and inserting equation 2.20 into 2.19 yields an expression entirely in θs. Equations for the emission probabilities are obtained in exactly the same way. This approach is slightly more straightforward than the one proposed in Baldi and Chauvin (1994), where the auxiliary variables are retained and the parameters of the model calculated explicitly from equation 2.18 after updating the auxiliary variables. This type of gradient descent is very similar to the exponentiated gradient descent proposed and investigated in Kivinen and Warmuth (1997) and Helmbold, Schapire, Singer, and Warmuth (1997).

2.2 The CHMM as a Probabilistic Independence Network. A large variety of probabilistic models can be represented as graphical models (Lauritzen, 1996), including the HMM and its variants. The relation between HMMs and probabilistic independence networks is thoroughly described

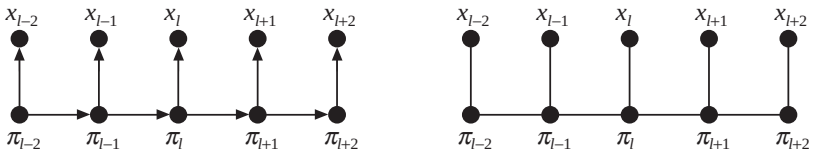


Figure 1: The DPIN (left) and UPIN (right) for an HMM.

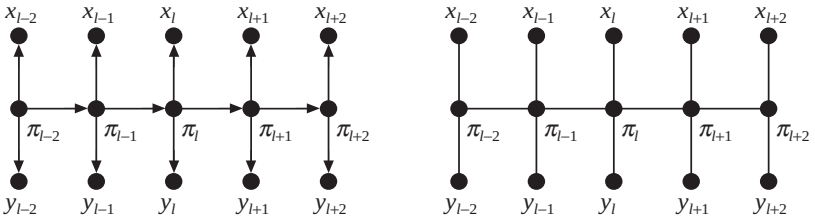


Figure 2: The DPIN (left) and UPIN (right) for a CHMM.

in Smyth, Heckerman, and Jordan (1997), and here we follow their terminology and refer the reader to that paper for more details.

An HMM can be represented as both a directed probabilistic independence network (DPIN) and an undirected one (UPIN) (see Figure 1). The DPIN shows the conditional dependencies of the variables in the HMM—both the observable ones (x) and the unobservable ones (π). For instance, the DPIN in Figure 1 shows that conditioned on π_l, x_l is independent of $x_1, \ldots, x_{l-1}$ and $\pi_1, \ldots, \pi_{l-1}$, that is, $P(x_l|x_1, \ldots, x_{l-1}, \pi_1, \ldots, \pi_l) = P(x_l|\pi_l)$. Similarly, $P(\pi_l|x_1, \ldots, x_{l-1}, \pi_1, \ldots, \pi_{l-1}) = P(\pi_l|\pi_{l-1})$. When "marrying" unconnected parents of all nodes in a DPIN and removing the directions, the moral graph is obtained. This is a UPIN for the model. For the HMM, the UPIN has the same topology as shown in Figure 1.

In the CHMM there is one more set of variables (the y's), and the PIN structures are shown in Figure 2. In a way, the CHMM can be seen as an HMM with two streams of observables, x and y, but they are usually not treated symmetrically. Again the moral graph is of the same topology, because no node has more than one parent.

It turns out that the graphical representation is the best way to see the difference between the CHMM and the IOHMM. In the IOHMM, the output y_l is conditioned on both the input x_l and the state π_l, but more important, the state is conditioned on the input. This is shown in the DPIN of Figure 3 (Bengio & Frasconi, 1996). In this case the moral graph is different, because π_l has two unconnected parents in the DPIN.

It is straightforward to extend the CHMM to have the label y conditioned on x, meaning that there would be arrows from x_l to y_l in the DPIN for the

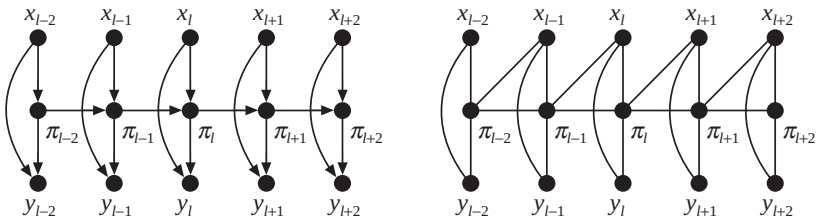


Figure 3: The DPIN for an IOHMM (left) is adapted from Bengio and Frasconi (1996). The moral graph to the right is a UPIN for an IOHMM.

CHMM. Then the only difference between the DPINs for the CHMM and the IOHMM would be the direction of the arrow between x_l and π_l. However, the DPIN for the CHMM would still not contain any "unmarried parents" and thus their moral graphs would be different.

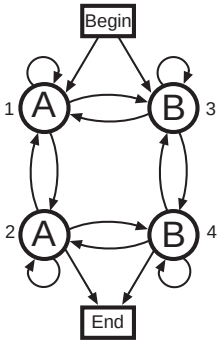
2.3 Calculation of Quantities Consistent with the Labels. Generally there are two different types of labeling: incomplete and complete labeling (Juang & Rabiner, 1991). We describe the modified forward-backward algorithm for both types of labeling below.

2.3.1 Complete Labels. In this case, each observation has a label, so the sequence of labels denoted $y = y_1, \ldots, y_L$ is as long as the sequence of observations. Typically the labels come in groups, that is, several consecutive observations have the same label. In speech recognition, the complete labeling corresponds to knowing which word or phoneme each particular observation x_l is associated with.

For complete labeling, the expectations in the clamped phase are averages over "allowed" paths through the model—paths in which the labels of the states agree with the labeling of the observations. Such averages can be calculated by limiting the sum in the forward and backward recursions to states with the correct label. The new forward and backward variables, $\tilde{\alpha}_i(l)$ and $\tilde{\beta}_i(l)$, are defined as $\alpha_i(l)$ (see equation 2.2) and $\beta_i(l)$ (see equation 2.5), but with $\phi_i(x_l)$ replaced by $\phi_i(x_l)\delta_{y_l,c_i}$. The expected counts $m_{ij}(l)$ and $m_i(l)$ for the allowed paths are calculated exactly as $n_{ij}(l)$ and $n_i(l)$, but using the new forward and backward variables.

If we think of $\alpha_i(l)$ (or $\beta_i(l)$) as a matrix, the new algorithm corresponds to masking this matrix such that only allowed regions are calculated (see Figure 4). Therefore the calculation is faster than the standard forward (or backward) calculation of the whole matrix.

2.3.2 Incomplete Labels. When dealing with incomplete labeling, the whole sequence of observations is associated with a shorter sequence of



Sequence		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
Labels		A	A	A	B	B	B	B	A	A	B	B	A	A	A
State	1 A				$\tilde{\alpha}$=0						$\tilde{\alpha}$=0				
	2 A														
	3 B	$\tilde{\alpha}$=0							$\tilde{\alpha}$=0				$\tilde{\alpha}$=0		
	4 B														

Figure 4: (Left) A very simple model with four states, two labeled A and two labeled B. (Right) The $\tilde{\alpha}$ matrix for an example of observations $x_1, \ldots, x_{14}$ with complete labels. The gray areas of the matrix are calculated as in the standard forward algorithm, whereas $\tilde{\alpha}$ is set to zero in the white areas. The $\tilde{\beta}$ matrix is calculated in the same way, but from right to left.

labels $y = y_1, \ldots, y_S$, where $S < L$. The label of each individual observation is unknown; only the order of labels is available. In continuous speech recognition, the correct string of phonemes is known (because the spoken words are known in the training set), but the time boundaries between them are unknown. In such a case, the sequence of observations may be considerably longer than the label sequence. The case $S = 1$ corresponds to classifying the whole sequence into one of the possible classes (e.g., isolated word recognition).

To compute the expected counts for incomplete labeling, one has to ensure that the sequence of labels matches the sequence of *groups of states with the same label*.[1] This is less restrictive than the complete label case. An easy way to ensure this is by rearranging the (big) model temporarily for each observation sequence and collecting the statistics (the m's) by running the standard forward-backward algorithm on this model. This is very similar to techniques already used in several speech applications (see, e.g., Lee, 1990), where phoneme (sub)models corresponding to the spoken word or sentence are concatenated. Note, however, that for the CHMM, the transitions between states with different labels retain their original value in the temporary model (see Figure 5).

[1] If multiple labels are allowed in each state, an algorithm similar to the forward-backward algorithm for asynchronous IOHMMs (Bengio & Bengio, 1996) can be used; see Riis (1998a).

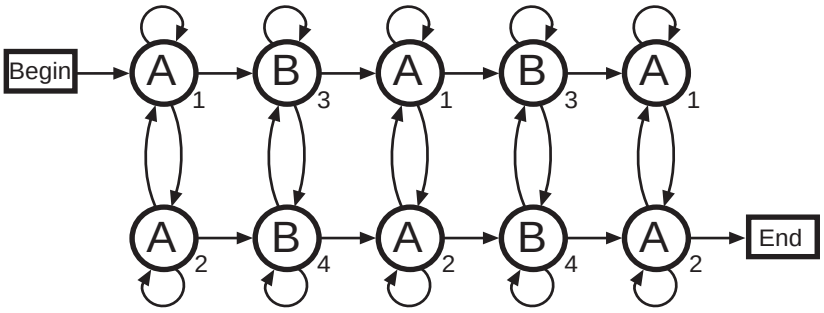


Figure 5: For the same model as in Figure 4, this example shows how the model is temporarily rearranged for gathering statistics (i.e., calculation of m values) for a sequence with incomplete labels *ABABA*.

3 Hidden Neural Networks

HMMs are based on a number of assumptions that limit their classification abilities. Combining the CHMM framework with neural networks can lead to a more flexible and powerful model for classification. The basic idea of the HNN presented here is to replace the probability parameters of the CHMM by state-specific multilayer perceptrons that take the observations as input. Thus, in the HNN, it is possible to assign up to three networks to each state: (1) a *match network* outputting the "probability" that the current observation matches a given state, (2) a *transition network* that outputs transition "probabilities" dependent on observations, and (3) a *label network* that outputs the probability of the different labels in this state. We have put "probabilities" in quotes because the output of the match and transition networks need not be properly normalized probabilities, since global normalization is used. For brevity we limit ourselves here to one label per state; the label networks are not present. The case of multiple labels in each state is treated in more detail in Riis (1998a).

The CHMM match probability $\phi_i(x_l)$ of observation x_l in state i is replaced by the output of a match network, $\phi_i(s_l; w^i)$, assigned to state i. The match network in state i is parameterized by a weight vector w^i and takes the vector s_l as input. Similarly, the probability θ_{ij} of a transition from state i to j is replaced by the output of a transition network $\theta_{ij}(s_l; u^i)$, which is parameterized by weights u^i. The transition network assigned to state i has $\mathcal{J}_i$ outputs, where $\mathcal{J}_i$ is the number of (nonzero) transitions *from* state i. Since we consider only states with one possible label, the label networks are just delta functions, as in the CHMM described earlier.

The network input s_l corresponding to x_l will usually be a window of context around x_l, such as a symmetrical context window of $2K + 1$

observations,[2] $x_{l-K}, x_{l-K+1}, \ldots, x_{l+K}$; however, it can be any sort of information related to x_l or the observation sequence in general. We will call s_l the *context* of observation x_l, but it can contain all sorts of other information and can differ from state to state. The only limitation is that it cannot depend on the path through the model, because then the state process is no longer first-order Markovian.

Each of the three types of networks in an HNN state can be omitted or replaced by standard CHMM probabilities. In fact, all sorts of combinations with standard CHMM states are possible. If an HNN contains only transition networks (that is, $\phi_i(s_l; w^i) = 1$ for all i, l) the model can be normalized locally by using a softmax output function as in the IOHMM. However, if it contains match networks, it is usually impossible to make $\sum_{x\in\mathcal{X}} P(x|\Theta) = 1$ by normalizing locally even if the transition networks are normalized. A probabilistic interpretation of the HNN is instead ensured by global normalization. We define the joint probability

$$\begin{aligned} P(x, y, \pi|\Theta) &= \frac{1}{Z(\Theta)} R(x, y, \pi|\Theta) \\ &= \frac{1}{Z(\Theta)} \prod_l \theta_{\pi_{l-1}\pi_l}(s_l; u^{\pi_{l-1}}) \phi_{\pi_l}(s_l; w^{\pi_l}) \delta_{y_l, c^{\pi_l}}, \end{aligned} \tag{3.1}$$

where the normalizing constant is $Z(\Theta) = \sum_{x,y,\pi} R(x, y, \pi|\Theta)$. From this,

$$\begin{aligned} P(x, y|\Theta) &= \frac{1}{Z(\Theta)} R(x, y|\Theta) = \frac{1}{Z(\Theta)} \sum_\pi R(x, y, \pi|\Theta) \\ &= \frac{1}{Z(\Theta)} \sum_{\pi\in\mathcal{C}(y)} R(x, \pi|\Theta), \end{aligned} \tag{3.2}$$

where

$$R(x, \pi|\Theta) = \prod_l \theta_{\pi_{l-1}\pi_l}(s_l; u^{\pi_{l-1}}) \phi_{\pi_l}(s_l; w^{\pi_l}). \tag{3.3}$$

Similarly,

$$\begin{aligned} P(x|\Theta) &= \frac{1}{Z(\Theta)} R(x|\Theta) = \frac{1}{Z(\Theta)} \sum_{y,\pi} R(x, y, \pi|\Theta) \\ &= \frac{1}{Z(\Theta)} \sum_\pi R(x, \pi|\Theta). \end{aligned} \tag{3.4}$$

[2] If the observations are inherently discrete (as in protein modeling), they can be encoded in binary vectors and then used in the same manner as continuous observation vectors.

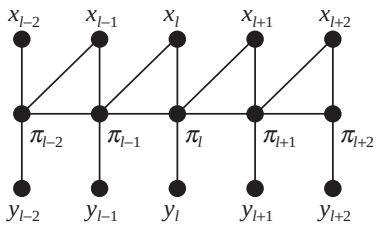


Figure 6: The UPIN for an HNN using transition networks that take only the current observation as input ($s_l = x_l$).

It is sometimes possible to compute the normalization factor Z, but not in all cases. However, for CML estimation, the normalization factor cancels out,

$$P(y|x, \Theta) = \frac{R(x, y|\Theta)}{R(x|\Theta)}. \tag{3.5}$$

The calculation of $R(x|\Theta)$ and $R(x, y|\Theta)$ can be done exactly as the calculation of $P(x|\Theta)$ and $P(x, y|\Theta)$ in the CHMM, because the forward and backward algorithms are not dependent on the normalization of probabilities.

Because one cannot usually normalize the HNN locally, there exists no directed graph (DPIN) for the general HNN. For UPINs, however, local normalization is not required. For instance, the Boltzmann machine can be drawn as a UPIN, and the Boltzmann chain (Saul & Jordan, 1995) can actually be described by a UPIN identical to the one for a globally normalized discrete HMM in Figure 1. A model with a UPIN is characterized by its clique functions, and the joint probability is the product of all the clique functions (Smyth et al., 1997). The three different clique functions are clearly seen in equation 3.1. In Figure 6 the UPIN for an HNN with transition networks and $s_l = x_l$ is shown; this is identical to Figure 3 for the IOHMM, except that it does not have edges from x to y. Note that the UPIN remains the same if match networks (with $s_l = x_l$) are used as well. The graphical representation as a UPIN for an HNN with no transition networks and match networks having a context of one to each side is shown in Figure 7 along with the three types of cliques.

A number of authors have investigated compact representations of conditional probability tables in DPINs (see Boutilier, Friedman, Goldszmidt, & Koller, 1996, and references therein). The HNN provides a similar compact representation of clique functions in UPINs, and this holds also for models that are more general than the HMM-type graphs discussed in this article.

The fact that the individual neural network outputs do not have to normalize gives us a great deal of freedom in selecting the output activation

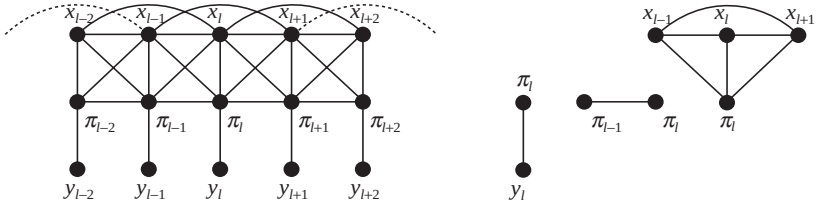


Figure 7: (Left) The UPIN of an HNN with no transition networks and match networks having a context of one to each side. (Right) The three different clique types contained in the graph.

function. A natural choice is a standard (asymmetric) sigmoid or an exponential output activation function, $g(h) = \exp(h)$, where h is the input to the output unit in question.

Although the HNN is a very intuitive and simple extension of the standard CHMM, it is a much more powerful model. First, neural networks can implement complex functions using far fewer parameters than, say, a mixture of gaussians. Furthermore, the HNN can directly use observation context as input to the neural networks and thereby exploit higher-order correlations between consecutive observations, which is difficult in standard HMMs. This property can be particularly useful in problems like speech recognition, where the pronunciation of one phoneme is highly influenced by the acoustic context in which it is uttered. Finally, the observation context dependency on the transitions allows the HNN to model the data as successive steady-state segments connected by "nonstationary" transitional regions. For speech recognition this is believed to be very important (see, e.g., Bourlard, Konig, & Morgan, 1994; Morgan, Bourlard, Greenberg, and Hermansky, 1994).

3.1 Training an HNN. As for the CHMM, it is not possible to train the HNN using an EM algorithm; instead, we suggest training the model using gradient descent. From equations 2.15 and 2.16, we find the following gradients of $\mathcal{L} = -\log P(y|x, \Theta)$ with regard to a generic weight ω^i in the match or transition network assigned to state i,

$$\frac{\partial \mathcal{L}}{\partial \omega^i} = -\sum_l \frac{m_i(l) - n_i(l)}{\phi_i(s_l; w^i)} \frac{\partial \phi_i(s_l; w^i)}{\partial \omega^i} - \sum_{lj} \frac{m_{ij}(l) - n_{ij}(l)}{\theta_{ij}(s_l; u^i)} \frac{\partial \theta_{ij}(s_l; u^i)}{\partial \omega^i}, \quad (3.6)$$

where it is assumed that networks are not shared between states. In the backpropagation algorithm for neural networks (Rumelhart, Hinton, & Williams, 1986) the squared error of the network is minimized by gradient descent. For an activation function g, this gives rise to a weight update of the form $\Delta w \propto -\mathcal{E} \times \frac{\partial g}{\partial w}$. We therefore see from equation 3.6 that the neural networks

are trained using the standard backpropagation algorithm where the quantity to backpropagate is $\mathcal{E} = [m_i(l) - n_i(l)]/\phi_i(s_l; w^i)$ for the match networks and $\mathcal{E} = [m_{ij}(l) - n_{ij}(l)]/\theta_{ij}(s_l; u^i)$ for the transition networks. The m and n counts are calculated as before by running two forward-backward passes: once in the clamped phase (the m's) and once in the free-running phase (the n's).

The training can be done in either batch mode, where all the networks are updated after the entire training set has been presented to the model, or sequence on-line mode, where the update is performed after the presentation of each sequence. There are many other variations possible. Because of the l dependence of $m_{ij}(l)$, $m_i(l)$ and the similar n's, the training algorithm is not as simple as for standard HMMs; we have to do a backpropagation pass for each l. Because the expected counts are not available before the forward-backward passes have been completed, we must either store or recalculate all the neural network unit activations for each input s_l before running backpropagation. Storing all activations can require large amounts of memory even for small networks if the observation sequences are very long (which they typically are in continuous speech). For such tasks, it is necessary to recalculate the network unit activations before each backpropagation pass. Many of the standard modifications of the backpropagation algorithm can be incorporated, such as momentum and weight decay (Hertz, Krogh, & Palmer, 1991). It is also possible to use conjugate gradient descent or approximative second-order methods like pseudo-Gauss-Newton. However, in a set of initial experiments for the speech recognition task reported in section 4, on-line gradient methods consistently gave the fastest convergence.

4 Comparison to Other Work

Recently several HMM / NN hybrids have been proposed in the literature. The hybrids can roughly be divided into those estimating the parameters of the HMM and the NN separately (see, e.g., Renals, Morgan, Bourlard, Cohen, & Franco, 1994; Robinson, 1994; Le Cerf, Ma, & Compernolle, 1994; McDermott & Katagiri, 1991) and those applying simultaneous or joint estimation of all parameters as in the HNN (see, e.g., Baldi & Chauvin, 1996; Konig, Bourlard, & Morgan, 1996; Bengio, De Mori, Flammia, & Kompe, 1992; Johansen, 1994; Valtchev, Kapadia, & Young, 1993; Bengio & Frasconi, 1996; Hennebert, Ris, Bourlard, Renals, & Morgan, 1997; Bengio, LeCun, Nohl, & Burges, 1995).

In Renals et al. (1994) a multilayer perceptron is trained separately to estimate phoneme posterior probabilities, which are scaled with the observed phoneme frequencies and then used instead of the usual emission densities in a continuous HMM. A similar approach is taken in Robinson (1994), but here a recurrent NN is used. A slightly different method is used in McDermott and Katagiri (1991) and Le Cerf et al. (1994), where the vector quantizer front end in a discrete HMM is replaced by a multilayer percep-

tron or a learning vector quantization network (Kohonen, Barna, & Chrisley, 1988). In contrast, our approach uses only one output for each match network whereby continuous and discrete observations are treated the same.

Several authors have proposed methods in which all parameters are estimated simultaneously as in the HNN. In some hybrids, a big multilayer perceptron (Bengio et al., 1992; Johansen & Johnsen, 1994) or recurrent network (Valtchev et al., 1993) performs an adaptive input transformation of the observation vectors. Thus, the network outputs are used as new observation vectors in a continuous density HMM, and simultaneous estimation of all parameters is performed by backpropagating errors calculated by the HMM into the neural network in a way similar to the HNN training. Our approach is somewhat similar to the idea of adaptive input transformations, but instead of retaining the computationally expensive mixture densities, we replace these by match networks. This is also done in Bengio et al. (1995), where a large network with the same number of outputs as there are states in the HMM is trained by backpropagating errors calculated by the HMM. Instead of backpropagating errors from the HMM into the neural network, Hennebert et al. (1997) and Senior and Robinson (1996) use a two-step iterative procedure to train the networks. In the first step, the current model is used for estimating a set of "soft" targets for the neural networks, and then the network is trained on these targets. This method extends the scaled likelihood approach by Renals et al. (1994) to use global estimation where training is performed by a generalized EM (GEM) algorithm (Hennebert et al., 1997).

The IOHMM (Bengio and Frasconi, 1996) and the CHMM/HNN have different graphical representations, as seen in Figures 2, 3, and 7. However, the IOHMM is very similar to a locally normalized HNN with a label and transition network in each state, but no match network. An important difference between the two is in the decoding, where the IOHMM uses only a forward pass, which makes it insensitive to future events but makes the decoding "real time." (See Riis, 1998a, for more details.)

5 Experiments

In this section we give an evaluation of the HNN on the task introduced in Johansen (1994) of recognizing five broad phoneme classes in continuous read speech from the TIMIT database (Garofolo et al., 1993): vowels (V), consonants (C), nasals (N), liquids (L) and silence (S) (see Table 1).

We use one sentence from each of the 462 speakers in the TIMIT training set for training, and the results are reported for the recommended TIMIT core test set containing 192 sentences. An additional validation set of 144 sentences has been used to monitor performance during training. The raw speech signal is preprocessed using a standard mel cepstral preprocessor, which outputs a 26-dimensional feature vector each 10 ms (13 mel cepstral features and 13 delta features). These vectors are normalized to zero mean

Table 1: Definition of Broad Phoneme Classes.

Broad Class	TIMIT Phoneme Label
Vowel (V)	iy ih eh ae ix ax ah ax-h uw uh ao aa ey ay oy aw ow ux
Consonant (C)	ch jh dh b d dx g p t k z zh v f th s sh hh hv
Nasal (N)	m n en ng em nx eng
Liquid (L)	l el r y w er axr
Silence (S)	h# pau

and unit variance. Each of the five classes is modeled by a simple left-to-right three-state model. The last state in any submodel is fully connected to the first state of all other submodels. (Further details are given in Riis & Krogh, 1997.)

5.1 Baseline Results. In Table 2, the results for complete label training are shown for the baseline system, which is a discrete CHMM using a codebook of 256 codebook vectors. The results are reported in the standard measure of percentage accuracy, $\%Acc = 100\% - \%Ins - \%Del - \%Sub$, where $\%Ins$, $\%Del$ and $\%Sub$ denote the percentage of insertions, deletions and substitutions used for aligning the observed and the predicted transcription.[3] In agreement with results reported in Johansen (1994), we have observed an increased performance for CML estimated models when using a forward or all-paths decoder instead of the best-path Viterbi decoder. In this work we use an N-best decoder (Schwarz & Chow, 1990) with 10 active hypotheses during decoding. Only the top-scoring hypothesis is used at the end of decoding. The N-best decoder finds (approximatively) the most probable labels, which depends on many different paths, whereas the Viterbi algorithm finds only the most probable path. For ML-trained models, the N-best and Viterbi decoder yield approximately the same accuracy (see Table 2). As shown by an example in Figure 8, several paths contribute to the optimal labeling in the CML estimated models, whereas only a few paths contribute significantly for the ML estimated models.

Table 2 shows that additional incomplete label training of a complete label trained model does not improve performance for the ML estimated model. However, for the CML estimation, there is a significant gain in accuracy by incomplete label training. The reason is that the CML criterion is very sensitive to mislabelings, because it is dominated by training sequences with an unlikely labeling. Although the phoneme segmentation (complete labeling) in TIMIT is done by hand, it is imperfect. Furthermore, it is often impossible—or even meaningless—to assign exact boundaries between phonemes.

[3] The NIST standard scoring package "sclite" version 1.1 is used in all experiments.

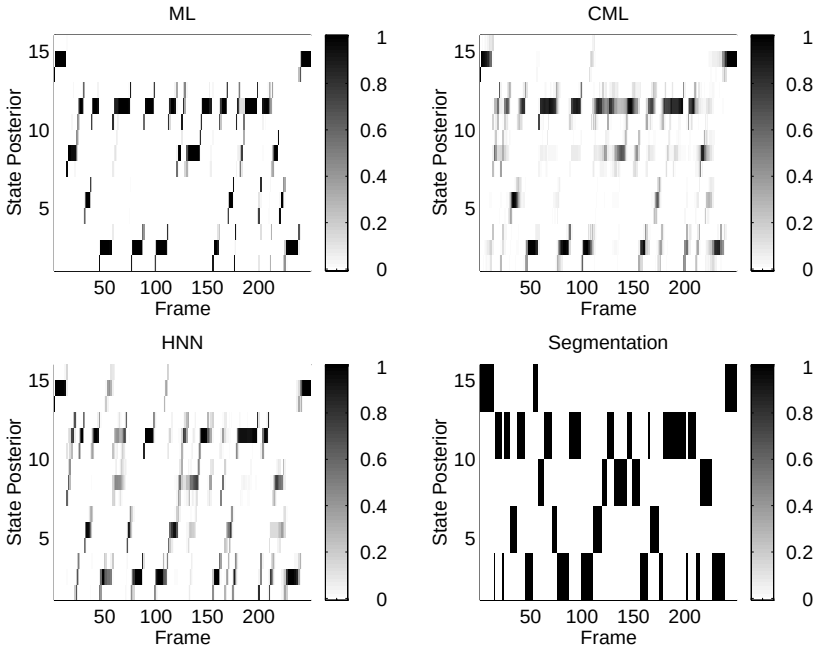


Figure 8: State posterior plots ($P(\pi_l = i \mid x, \Theta)$) for baseline and HNN for the test sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). States 1–3 belong to the consonant model, 4–6 to the nasal model, 7–9 to the liquid model, 10–12 to the vowel model, and 13–15 to the silence model. (Top left) ML-trained baseline, which yield $\%Acc = 62.5$ for this sentence. (Top right) CML-trained baseline ($\%Acc = 78.1$). (Bottom left) HNN using both match and transition networks with 10 hidden units and context $K = 1$ ($\%Acc = 93.7$). (Bottom right) The observed segmentation.

Table 2: Baseline Recognition Accuracies.

	Viterbi	N-Best
Complete labels		
ML	75.9	76.1
CML	76.6	79.0
Incomplete labels		
ML	75.8	75.2
CML	78.4	81.3

Note: The baseline system contains 3856 free parameters.

CML gives a big improvement from an accuracy of around 76% for the ML estimated models to around 81%. Statistical significance is hard to assess, because of the computational requirements for this task, but in a set of 10 CML training sessions of random initial models, we observed a deviation of no more than $\pm 0.2\%$ in accuracy. For comparison, a MMI-trained model with a single diagonal covariance gaussian per state achieved a result of 72.4% accuracy in Johansen and Johnsen (1994).

5.2 HNN Results. For the HNN, two series of experiments were conducted. In the first set of experiments, only a match network is used in each state, and the transitions are standard HMM transitions. In the second set of experiments, we also use match networks, but the match distribution *and* the standard transitions in the last state of each submodel are replaced by a transition network. All networks use the same input s_l, have the same number of hidden units, are fully connected, and have sigmoid output functions. This also applies for the transition networks; that is, a softmax output function is *not* used for the transition networks.

Although the HNN with match networks and no hidden units has far fewer parameters than the baseline system, it achieves a comparable performance of 80.8% accuracy using only the current observation x_l as input ($K = 0$) and 81.7% accuracy for a context of one left and right observation ($K = 1$) (see Table 3). No further improvement was observed for larger contexts. Note that the match networks without hidden units just implement linear weighted sums of input features (passed through a sigmoid output function). For approximately the same number of parameters as used in the baseline system, the HNN with 10 hidden units and no context ($K = 0$) yields 84.0% recognition accuracy. Increasing the context or number of hidden units for this model yields a slightly lower accuracy due to overfitting.

In Johansen (1994) a multilayer perceptron was used as a global adaptive input transformation to a continuous density HMM with a single diagonal covariance gaussian per state. Using N-best decoding and CML estimation, a result of 81.3% accuracy was achieved on the broad phoneme class task.

When using a transition network in the last state of each submodel, the accuracy increases, as shown in Table 3. Thus, for the model with context $K = 1$ and no hidden units, an accuracy of 82.3% is obtained compared to 81.7% for the same model with only match networks. The best result on the five broad class task is an accuracy of 84.4% obtained by the HNN with context $K = 1$, match and transition networks and 10 hidden units in all networks (see Table 3).

6 Conclusion

In this article we described the HNN, which in a very natural way replaces the probability parameters of an HMM with the output of state-specific

Table 3: Recognition Accuracies for HNNs.

	Context K	Number of Parameters	Accuracy
No hidden units			
HNN, match networks	0	436	80.8
HNN, match networks	1	1,216	81.7
HNN, match and transition networks	1	2411	82.3
Ten hidden units			
HNN, match networks	0	4,246	84.0
HNN, match networks	1	12,046	83.8
HNN, match and transition networks	1	12,191	84.4

Note: "HNN, match networks" are models using only match networks and standard CHMM transitions, whereas "HNN, match and transition networks" use both match and transition networks. Decoding is done by N-best.

neural networks. The model is normalized at a global level, which ensures a proper probabilistic interpretation of the HNN. All the parameters in the model are trained simultaneously from labeled data using gradient-descent-based CML estimation. The architecture is very flexible in that all combinations with standard CHMM probability parameters are possible. The relation to graphical models was discussed, and it was shown that the HNN can be viewed as an undirected probabilistic independence network, where the neural networks provide a compact representation of the clique functions.

Finally, it was shown that the HNN improves on the results of a speech recognition problem with a reduced set of phoneme classes. The HNN has also been applied to the recognition of task-independent isolated words from the PHONEBOOK database (Riis, 1998b) and preliminary results on the 39 phoneme TIMIT problem are presented in Riis (1998a).

Acknowledgments

We thank Steve Renals and Finn T. Johansen for valuable comments and suggestions to this work. We also thank the anonymous referees for pointing our attention to graphical models. This work was supported by the Danish National Research Foundation.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9*, 147–169.

Bahl, L. R., Brown, P. F., de Souza, P. V., & Mercer, R. L. (1986). Maximum mu-

tual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of ICASSP'86* (pp. 49–52).

Baldi, P., & Chauvin, Y. (1994). Smooth on-line learning algorithms for hidden Markov models. *Neural Computation, 6*(2), 307–318.

Baldi, P., & Chauvin, Y. (1996). Hybrid modeling, HMM / NN architectures, and protein applications. *Neural Computation, 8*, 1541–1565.

Bengio, S., & Bengio, Y. (1996). An EM algorithm for asynchronous input / output hidden Markov models. In *Proceedings of the ICONIP'96*.

Bengio, Y., De Mori, R., Flammia, G., & Kompe, R. (1992). Global optimization of a neural network–hidden Markov model hybrid. *IEEE Transactions on Neural Networks, 3*(2), 252–259.

Bengio, Y., & Frasconi, P. (1996). Input / output HMMs for sequence processing. *IEEE Transactions on Neural Networks, 7*(5), 1231–1249.

Bengio, Y., LeCun, Y., Nohl, C., & Burges, C. (1995). Lerec: A NN / HMM hybrid for on-line handwritting recognition. *Neural Computation, 7*(5).

Bourlard, H., Konig, Y., & Morgan, N. (1994). *REMAP: Recursive estimation and maximization of a posteriori probabilities* (Tech. Rep. TR-94-064). Berkeley, CA: International Computer Science Institute.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-specific independence in Bayesian networks. In E. Horvitz & F. V. Jensen (Eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 115–123). San Francisco: Morgan Kaufmann.

Bridle, J. S. (1990). Alphanets: A recurrent "neural" network architecture with a hidden Markov model interpretation. *Speech Communication, 9*, 83–92.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society B, 39*, 1–38.

Durbin, R. M., Eddy, S. R., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis.* Cambridge: Cambridge University Press.

Eddy, S. R. (1996). Hidden Markov models. *Current Opinion in Structural Biology, 6*, 361–365.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallet, D. S., & Dahlgren, N. L. (1993). *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM*. Gaithersburg, MD: National Institute of Standards.

Gopalakrishnan, P. S., Kanevsky, D., Nádas, A., & Nahamoo, D. (1991). An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory, 37*(1), 107–113.

Helmbold, D. P., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning, 27*(1), 97–119.

Hennebert, J., Ris, C., Bourlard, H., Renals, S., & Morgan, N. (1997). Estimation of global posteriors and forward-backward training of hybrid HMM / ANN systems. In *Proceedings of EUROSPEECH'97*.

Hertz, J. A., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

Johansen, F. T. (1994). Global optimisation of HMM input transformations. In *Proceedings of ICSLP'94* (Vol. 1, pp. 239–242).

Johansen, F. T., & Johnsen, M. H. (1994). Non-linear input transformations for discriminative HMMs. In *Proceedings of ICASSP'94* (Vol. 1, pp. 225–228).

Juang, B. H., & Rabiner, L. R. (1991). Hidden Markov models for speech recognition. *Technometrics, 33*(3), 251–272.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation, 132*(1), 1–63.

Kohonen, T., Barna, G., & Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proceedings of ICNN'88* (Vol. 1, pp. 61–68).

Konig, Y., Bourlard, H., & Morgan, N. (1996). REMAP: Recursive estimation and maximization of a posteriori probabilities—application to transition-based connectionist speech recognition. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8* (pp. 388–394). Cambridge, MA: MIT Press.

Krogh, A. (1994). Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR ICPR'94* (pp. 140–144).

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., & Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology, 235*, 1501–1531.

Lauritzen, S. L. (1996). *Graphical models*. New York: Oxford University Press.

Le Cerf, P., Ma, W., & Compernolle, D. V. (1994). Multilayer perceptrons as labelers for hidden Markov models. *IEEE Transactions on Speech and Audio Processing, 2*(1), 185–193.

Lee, K.-F. (1990). Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing, 38*(4), 599–609.

McDermott, E., & Katagiri, S. (1991). LVQ-based shift-tolerant phoneme recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing, 39*, 1398–1411.

Morgan, N., Bourlard, H., Greenberg, S., & Hermansky, H. (1994). Stochastic perceptual auditory-event-based models for speech recognition. In *Proceedings of Interternational Conference on Spoken Language Processing* (pp. 1943–1946).

Nádas, A. (1983). A decision-theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing, 31*(4), 814–817.

Nádas, A., Nahamoo, D., & Picheny, M. A. (1988). On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing, 36*(9), 814–817.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE, 77*(2), 257–286.

Renals, S., Morgan, N., Bourlard, H., Cohen, M., & Franco, H. (1994). Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing, 2*(1), 161–174.

Riis, S. K. (1998a). *Hidden Markov models and neural networks for speech recognition*. Doctoral dissertation, IMM-PHD-1998-46, Technical University of Denmark.

Riis, S. K. (1998b). Hidden neural networks: Application to speech recognition. In *In Proceedings of ICASSP'98* (Vol. 2, pp. 1117–1121).

Riis, S. K., & Krogh, A. (1997). Hidden neural networks: A framework for HMM / NN hybrids. In *Proceedings of ICASSP'97* (pp. 3233–3236).

Robinson, A. J. (1994). An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks, 5*, 298–305.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Saul, L. K., & Jordan, M. I. (1995). Boltzman chains and hidden Markov models. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 435–442). San Mateo, CA: Morgan Kaufmann.

Schwarz, R., & Chow, Y.-L. (1990). The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses. In *Proceedings of ICASSP'90* (pp. 81–84).

Senior, A., & Robinson, T. (1996). Forward-backward retraining of recurrent neural networks. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing dystems, 8* (pp. 743–749). San Mateo, CA: Morgan Kaufmann.

Smyth, P., Heckerman, D., & Jordan, M. I. (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation, 9*, 227–269.

Valtchev, V., Kapadia, S., & Young, S. (1993). Recurrent input transformations for hidden Markov models. In *Proceedings of ICASSP'93* (pp. 287–290).

Received October 21, 1997; accepted May 7, 1998.