Products of Hidden Markov Models

Andrew D. Brown[*]
Dept. of Computer Science
University of Toronto
Canada
andy@cs.utoronto.ca

Geoffrey E. Hinton
Gatsby Computational Neuroscience Unit
University College London
London, UK
hinton@gatsby.ucl.ac.uk

Abstract

We present products of hidden Markov models (PoHMM's), a way of combining HMM's to form a distributed state time series model. Inference in a PoHMM is tractable and efficient. Learning of the parameters, although intractable, can be effectively done using the Product of Experts learning rule. The distributed state helps the model to explain data which has multiple causes, and the fact that each model need only explain part of the data means a PoHMM can capture longer range structure than an HMM is capable of. We show some results on modelling character strings, a simple language task and the symbolic family trees problem, which highlight these advantages.

1 Introduction

Hidden Markov models (HMM's) have been very successful in automatic speech recognition where they are the standard method for modelling and discriminating sequences of phonemes. Using the Markov dependence of the hidden state variable, they capture the dependence of each observation on the recent history of the sequence. They also have the advantage that there is a very efficient algorithm for fitting an HMM to data: the forward-backward algorithm and the Baum-Welch re-estimation formulas. However, HMM's have been less widely applied in other areas where statistical time series are used. In statistical language modelling, for example, the most common model is a fully-observed, second-order Markov model, known as a trigram.

One limitation of HMM's that makes them inappropriate for language modelling is that they represent the recent history of the time series using a single, discrete K-state multinomial. The efficiency of the Baum-Welch re-estimation algorithm depends on this fact, but it severely limits the representational power of the model. The hidden state of a single HMM can only convey $\log_2 K$ bits of information about the recent history. If the generative model had a *distributed* hidden state representation [6] consisting of M variables each with K alternative states it could convey $M \log_2 K$ bits of information, so the information bottleneck scales linearly with the number of variables and only logarithmically with the number of alternative states of each variable.

A second limitation of HMM's is that they have great difficulty in learning to capture long range dependencies in a sequence [1]. In the case of natural language there are many examples of word agreements which span a large portion of a sentence. As we shall demonstrate, this is much easier to model in a system that has distributed hidden state since each variable in the distributed state can be concerned with a specific type of long-range regularity and does not get distracted by having to deal with all the other regularities in the time series.

2 Products of HMM's

Extending the hidden state of an HMM can be done in various ways. One is to add several hidden state variables which have a *causal* effect on the observed variables in the model. This is known as a Factorial HMM [2] and is shown in Fig. 1.

In a causal belief network each local probability distribution can be independently estimated given the posterior distribution of the hidden variables conditioned on the evidence. However, it is exponentially expensive to compute this posterior distribution exactly because observing the visible variables induces dependencies among the hidden variables. Ghahramani and Jordan handle this problem by approximating the pos-

[*]Current address: Gatsby Computational Neuroscience Unit, University College London, London, UK

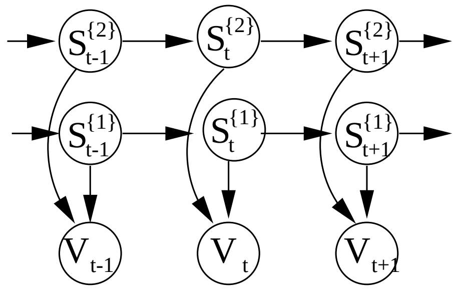

Figure 1: Factorial HMM

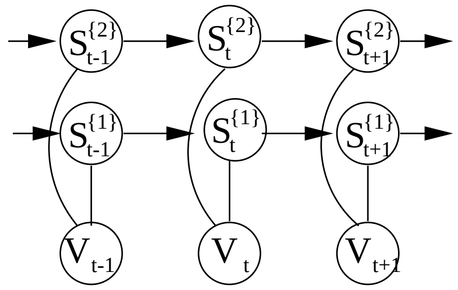

Figure 2: Product of HMM's

terior with a factored, variational distribution.

A very different way of combining multiple HMM's is to multiply their individual distributions together and renormalize (Fig. 2). This "Product of Experts" generative model can be represented as an undirected Markov network in which the hidden state variables are non-causally related to the visible variables. This PoHMM network has the opposite property of the FHMM, in that conditioned on a set of observations, the hidden state chains are independent. So exact inference can easily be performed by using the forward-backward algorithm in each chain separately. However, learning of the local probability functions is now more complex, because the local distributions are all linked by a global partition function. This may be seen in the equation for the density of a product model:

- Parameters: $\Theta = \{\theta\}_{m=1}^{M}$
- Observed Variables: $V_1^T = \{V_t\}_{t=1}^{T} \in \mathcal{V}$
- Hidden Variables: $S_1^T = \{S_t\}_{t=1}^{T} \in \mathcal{S}$

$$P(V_1^T|\Theta) = \frac{\prod_{m=1}^{M} \sum_{\mathcal{S}} P_m(V_1^T, S_1^T|\theta_m)}{Z(T,\Theta)}, \qquad (1)$$

$$Z(T,\Theta) = \sum_{\mathcal{V}} \prod_{m=1}^{M} P_m(V_1^T|\theta_m), \qquad (2)$$

where θ_m is the set of parameters for each HMM in the product. The existence of this summation over all the possible strings of a given length in the denominator of the equation makes it intractable to compute the exact gradient of the log likelihood of the observed data *w.r.t* the parameters, so it appears to be very hard to fit a PoHMM to data. Gibbs sampling can be used to estimate the derivatives of the partition function but this is very slow and noisy. Fortunately, there is an alternative objective function for learning whose gradient can be approximated accurately and efficiently [4]. It has been shown that optimizing this alternative objective function leads to good generative models for non-sequential data and we show here that the same approach works for PoHMM's.

Maximizing the log likelihood of the data is equivalent to minimizing the Kullback-Leibler divergence $KL(Q^0||Q^\infty)$ between the observed data distribution, Q^0, and the equilibrium distribution, Q^∞, produced by the generative model[1]. Instead of simply minimizing $KL(Q^0||Q^\infty)$ we minimize the "contrastive divergence" $KL(Q^0||Q^\infty) - KL(Q^1||Q^\infty)$, where Q^1 is the distribution over one-step reconstructions of the data that are produced by running a Gibbs sampler for one full step, starting at the data. The advantage of using the contrastive divergence as the objective function for learning is that the intractable derivatives of the partition function cancel out and if we are prepared to ignore a term that turns out to be negligible in practice [4] it is easy to follow the gradient of the contrastive divergence:

1. Calculate each model's gradient $\frac{\partial}{\partial \theta_m} P(V_1^T|\theta_m)$ on a data point using the forward-backward algorithm.
2. For each model take a sample from the posterior distribution of paths through state space.
3. At each time step, multiply together the distributions over symbols specified by the chosen paths in each HMM. Renormalize to get the reconstruction distribution at that time step.
4. Draw a sample from the reconstruction distribution at each time step to get a reconstructed sequence. Compute each model's gradient on the new sequence $\frac{\partial}{\partial \theta_m} P(\hat{V}_1^T|\theta_m)$
5. Update the parameters:

$$\Delta\theta_m \propto \frac{\partial \log P(V_1^T|\Theta)}{\partial \theta_m} - \frac{\partial \log P(\hat{V}_1^T|\Theta)}{\partial \theta_m}$$

[1]We call this distribution Q^∞ because one way to get exact samples from it is to run a Gibbs sampler for an infinite number of iterations

To compute the gradient of the HMM we use an EM like trick. Directly computing the gradient of an HMM is difficult due to the fact that all the parameters are coupled through their influence on the hidden states. If the HMM were visible and the hidden states were known then the gradient of the log-likelihood for each parameter would decouple into an expression involving only local variables. As in EM, we use the posterior distribution over the hidden states in place of actual values by using the identity:

$$\frac{\partial}{\partial\theta}\log P(V_1^T|\theta) = \frac{\partial}{\partial\theta}\langle\log P(V_1^T, S_1^T|\theta)\rangle_{P(S_1^T|V_1^T)} \quad (3)$$

This says that if we compute the posterior of the HMM using the forward backward algorithm we can take the gradient of the complete data log-likelihood using the sufficient statistics of the hidden variables in place of actual values.

A second optimization trick which we have used is to re-parameterize the probabilities of the HMM, using the softmax function. Working in this domain allows us to do unconstrained gradient descent over the real numbers. Doing gradient optimization directly in the probability domain would involve the more difficult proposition of constraining the parameters to the probability simplex. An added advantage of this reparamaterization is that the probabilities cannot go to zero anywhere. It is clearly desirable in the PoE framework that none of the individual experts assigns zero probability to an event.

As an example we look at the gradient rule for the transition probabilities of an HMM, $P(S_t = j|S_{t-1} = i) = A_{ij}$. If we re-parameterize using the softmax function:

$$A_{ij} = \frac{\exp(a_{ij})}{\sum_j \exp(a_{ij})}. \quad (4)$$

Taking the derivative with respect to a_{ij} yields

$$\frac{\partial}{\partial a_{ij}}\langle\log P(V_1^T, S_1^T)\rangle = \sum_{t=1}^{T}\langle S_t = j, S_{t-1} = i\rangle - (\sum_{t=1}^{T}\langle S_{t-1} = i\rangle)A_{ij}, (5)$$

As before the angle brackets indicate an expectation with respect to the posterior of the hidden states. This has the intuitive interpretation that the derivative for the softmax parameter a_{ij} regresses toward the point where A_{ij} is equal to the expected transition probability under the posterior. If we set the derivative to zero and solved this equation directly, we would recover the Baum-Welch update equation.

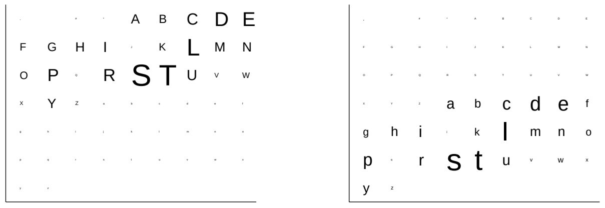

Figure 3: An 'eye-chart' diagram of the output distributions of the 2-state HMM in the PoHMM. Each chart corresponds to a single state's output distribution and the size of each symbol is proportional to the probability mass on that symbol.

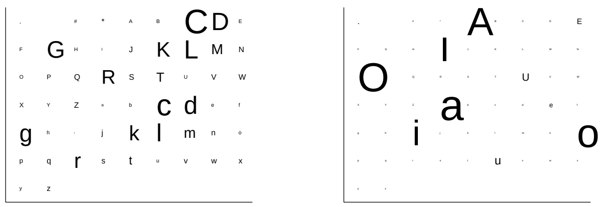

Figure 4: Eye-chart diagram of the output distributions of two of the states of the 30 state HMM

3 Results

To demonstrate the relative merits of a product of HMM's versus a single HMM, we have applied them to two problems in text and language modelling. The first of these is modelling strings of English letters, and the second is a task of discriminating sets of simple English sentences which exhibit long and short range dependencies.

3.1 Modelling Character Strings

The first experiment involved modelling character strings from a corpus of English text. The problem was slightly modified to better demonstrate the advantages of a product model. Rather than training the model on a single case, or mixed case text, we trained it on data in which the characters in a sentence were either all upper case or all lower case. Thus there really are independent factors underlying this sequence: the binary decision of upper case or lower case and the statistics of the letters.

We used 8600 sentences[2] and converted them to all upper and all lower case to yield over 17,000 training sentences. 56 symbols were allowed: 4 symbols for space and punctuation, 26 upper and 26 lower case letters. We compared a single HMM with 32 hidden

[2]from Thomas Hardy's "Tess of the d'Urbervilles" available from Project Gutenberg (http://www.gutenberg.net)

states against a product of a 2 state and a 30 state hidden Markov model. In the product model the 2 state HMM learns to differentiate upper and lower case. It 'votes' to put probability mass on the upper or lower case letters respectively (Fig. 3), and it enforces the continuity through its transition matrix. Then the 30-state HMM need only learn the case-independent statistics of the characters and the fact that the upper and lower case characters are analogous, placing proportional amounts of probability mass on the two halves of the symbol set. In Fig. 4 we see an example of two of the big HMM's 30 hidden states. Its output distributions are symmetric over the upper and lower case letters, indicating that it has left the modelling of case to the smaller 2-state HMM model.

By contrast, the single HMM has to partition its data space into two parts, one each for upper and lower case. In effect it has to model the caseless letter statistics with a much smaller number of hidden states. This can be seen in Fig. 5a) where the observation distributions of the 32 states fall into 3 categories: punctuation, upper case, and lower case. Similarly we can see in the transition matrix (Fig. 5b) that the upper case states only transition to upper case states and likewise for the lower case states.

While we cannot compute the log likelihood of a string under the PoHMM we can compute the probability of a single symbol conditioned on the other symbols in a sentence. This leads to a simple, interesting test of the models which we refer to as the "symmetric Shannon game". In the original Shannon game [5], a prediction of the next symbol in a sequence is made given the previous N symbols. In the symmetric Shannon game the model is given both past and future symbols and is asked to predict the current one. We can compute this distribution exactly since we need only normalize over the missing symbol and not all strings of symbols. For models based on directed acyclic graphs, such as an HMM, it is easy to compute the probability of the next symbol in a sequence given the symbols so far. Somewhat surprisingly, this is not true for undirected models like a PoHMM. If the data after time t is missing, the posterior distribution over paths through each HMM up to time t depends on how easily these paths can be extended in time so as to reach agreement on future data.

Table 6 shows a comparison of several PoHMM models with a single large HMM. They were scored on a set of 60 hold-out sentences with an equal number of upper and lower case. The product of a 2-state and 30-state HMM with 2728 parameters, while capturing the componential structure we were hoping for, does not outperform a single 32 state HMM which has been roughly matched for the number of parameters (2848 parameters). This is mainly an optimization problem, because if we train a 2-state model alone and a 30-state model on uni-case text, and then use their parameters to initialize the PoHMM then it does much better than the single HMM. If we use a product of many, simple HMM's then the optimization problem is eased. A product of 10, 4-state HMM's, which has still fewer parameters (2440), performs as well as a hand initialized product of 2 HMM's. Increasing, the number of HMM's in the product provides further improvements while the parameters and computation time scale linearly with the number of HMM's in the model.

Model	Sym. Shannon (bits)
PoHMM 40 x 4-states	1.96
PoHMM 20 x 4-states	2.06
PoHMM 10 x 4-states	2.13
PoHMM (2-state + 30-state, pre-initialized)	2.14
32 State HMM	2.46
PoHMM (2-state + 30-state random initialization)	2.73

Figure 6: Symmetric Shannon scores for several PoHMM models and a single large HMM

3.2 Modelling Simple Sentences

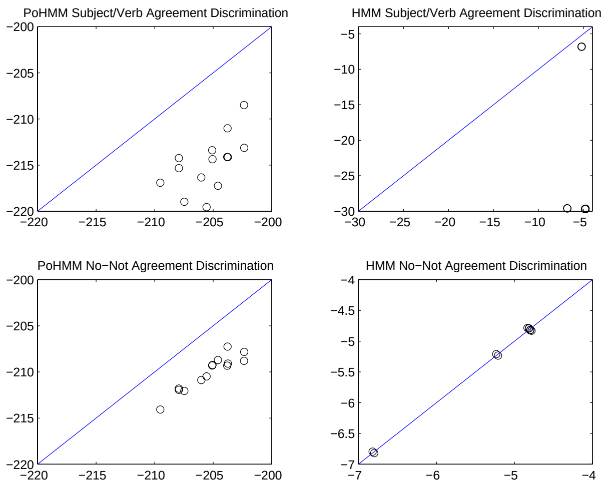

Figure 7: Discrimination diagrams of the correct and two incorrect sentence sets under each model. Circles below the line indicate that the model assigns higher probability to the correct sentence than the corrupted sentence. Circles on the line indicate that the model cannot discriminate the two. (Note there is some overlap of the circles in the HMM plots.)

In the second task, matching the models for the number of parameters, we use a single HMM with 32 states and a product of 10, 6 and 7 state HMM's to model a set of English sentences of the form, "Yes I am _" or "No she is not". There are 14 legal sentences in

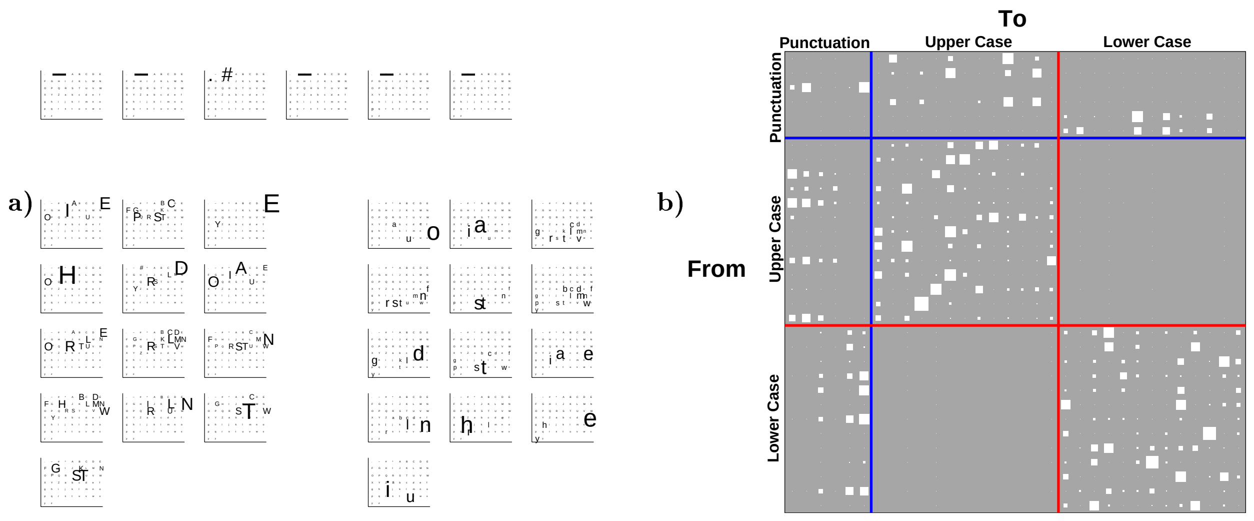

Figure 5: The 32 state HMM a) the observation probabilities of the HMM b) a diagram of the transition matrix where the area of the square indicates the probability of going to a state.

the grammar, including all combinations of yes and no with the pronouns (I,you,he,she,it,we,they) and their corresponding conjugation of the verb "to be". The sentences feature two kinds of agreement. There is short range agreement between the subject and the verb which are always adjacent, and there is longer range agreement between the "no" and "not" or "yes" and the null symbol which appear at the beginning and end of the sentence, respectively. To test whether the two types of models could capture these correlations, we created two sets of ungrammatical sentences in which either the verbs were wrong or the ending of the sentence did not match the beginning. We compared relative log-likelihoods of these sentences under each model, and the results are shown in Fig. 7. Both models can discriminate the ungrammatical sentences where short range structure is corrupted, but the single HMM cannot discriminate the cases where the longer range structure is corrupted.

3.3 Family Trees

The final example application of PoHMM's is one of symbolic inference in two family trees [3]. In the family trees problem we consider two families – one English and the other Italian. There are twelve people in each family. In addition there are twelve familial relationships such as father, daughter, uncle etc. The data set is composed of a set of triplets of the form *person relation person*. While the number of allowed triplets in the dataset covers only a small number of all the possible triplets, it is possible to generalize from training examples to unseen testing examples because there are a small number of interacting constraints on

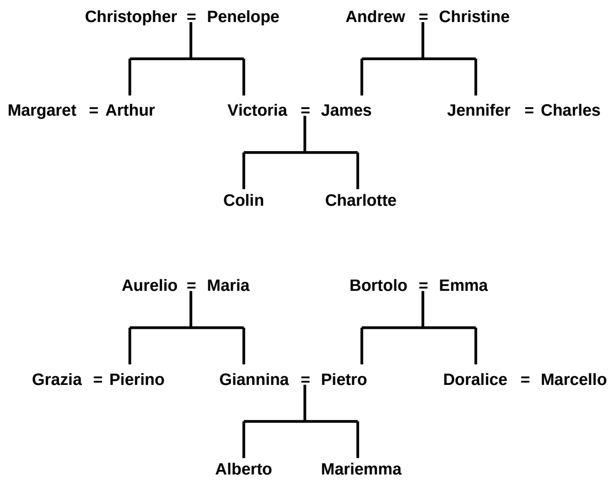

Figure 8: English and Italian family trees

the data. Fig. 8 shows the two family trees. The two families have identical structure so that relationships learned in one can be transferred by analogy to the other, in much the same way that the PoHMM learns the analogical relationship between characters in the upper and lower case text example. One can think of other rules of thumb which might be applied to this data such as only men can be husbands, or spouses must be of the same generation.

Treating each triplet as a sequence of symbols from an alphabet of 36 symbols (24 people and 12 relationships) we can train a PoHMM to learn transition probabilities and output probabilities which capture the structure in this data. Using a large number of HMM's, each with a small number of hidden states,

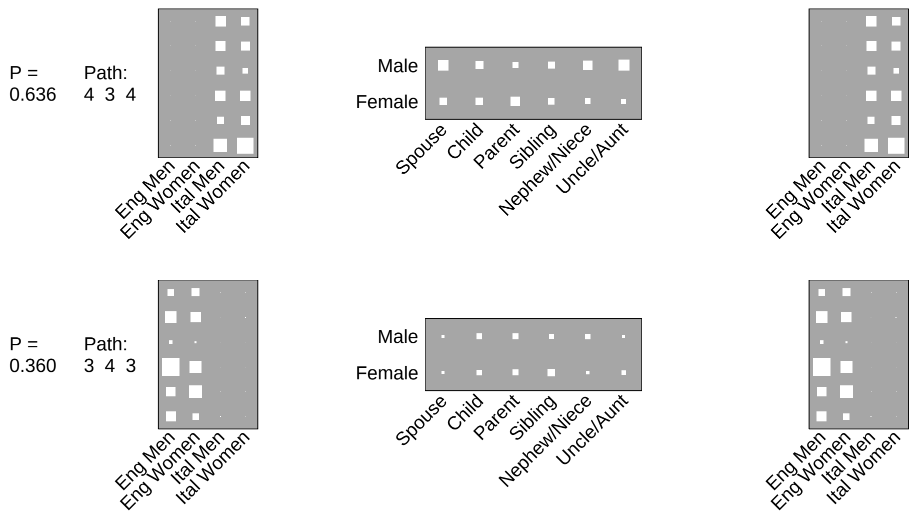

Figure 9: A 4-state HMM which encodes one rule of the family trees data – English and Italian are mutually exclusive. The display shows the path through the states and the probability of that path.

some of the models will learn to produce these rules of thumb in their transition structure. One obvious piece of structure in the triplets is that the first and third symbol always comes from the set of people and the second is always a relationship. We could construct a model which builds in this restriction, but a PoHMM easily learns this. A single model can alternate putting probability mass on the people and the relationships. The other models are then free to model other regularities in the data.

Fig. 9 shows an example of a 4-state model taken from a PoHMM trained on the family trees data. Since there are only a small number of paths through this HMM, they can all be enumerated and sorted according to their probability of occurrence. The figure shows the top two paths and their probability of occurrence. For each state in the path the output probabilities of each state have been displayed to elucidate their structure. In the first and third positions only the output probabilities over people are displayed and in the middle position only the output probabilities over relationships. The HMM uses only states 3 and 4, but it reuses them in a clever way. The most likely path is states 4-3-4, which puts high probability on an Italian, uniform probability on a relationship, and high probability on an Italian. The second most likely path, 3-4-3, shows a preference for English, followed by any relationship followed by English. Thus, this HMM has captured the mutual exclusion of nationality in the dataset. The Italian path is almost twice as probable as the English path, but this discrepancy is presumably offset by slight preferences for English over Italian in other HMM's.

While other rules are not so clear cut and easily interpretable, they express in a softer fashion similar constraints across age, and sex. When many such soft, probabilistic rules are applied they create a sharp distribution over the data.

4 Extensions

One concern that we have about the PoHMM is that each HMM has it's own output distribution over the data, which could include many parameters if there are a large number of symbols. One way to deal with this is to add an extra layer of shared hidden features between the hidden variables of the HMM and the output symbols. Sharing the output model features among the HMM's, it greatly reduces the number of free parameters in the PoHMM and it has the benefit that data regularities learned by one model do not have to be re-learned again and again in the other models. Each HMM retains it's own transition distribution and it's own weights from it's hidden states to the hidden features.

We parameterize the output model as a two layer network, with a linear hidden layer and a softmax nonlinearity in the output layer (Fig. 10). Note that we do not constrain the hidden layer values to be positive or sum to one. They may be positive or negative. If we constrained the hidden features to be a proper probability distribution then this would be equivalent to inserting a single discrete valued stochastic variable between the hidden variable and the visible variable of the HMM. This is not as powerful a representation as allowing the hidden features to take on independent

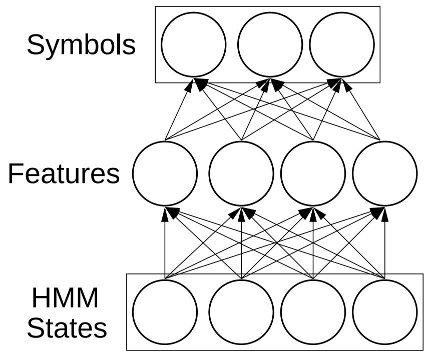

Figure 10: Output model of the HMM's

real values. The formula for such an output model is given by:

$$P(V|S;\theta_m) = \sigma(\mathbf{s}'U_m W) \tag{6}$$

Where we treat the hidden state, $\mathbf{s}$, as a column vector of indicator variables – a one in the position of the discrete state which the hidden variable takes. σ is the softmax function. U is the matrix of weights which the states of model m place on the hidden features and W is the matrix shared hidden features. Interestingly, this output distribution is also a product model. The columns of W are linearly combined in the log domain and then pushed through the softmax function to get a probability distribution. The rows of U are the weights that each state puts on these basis distributions.

There are two ways that we can regularize or constrain the output model. One way is to create a bottle neck by using a small number of hidden features. This is equivalent to decomposing the stochastic output matrix as the product of two lower rank matrices. The other way is to use a large number of hidden features, but use another regularizer on the output weights forcing them to be small. Thus, the hidden features are restricted to be soft distributions over the output symbols. We have applied this technique to the family trees problem, and it does help the generalization performance. We test the pattern completion performance of the PoHMM by clamping the first two entries of a tuple and computing the predictive distribution of the third. On fifteen learning trials, with 20 HMM's of 4 hidden states each, the PoHMM obtained perfect completion performance on the training data and 73% on the test data. This is competitive with the backpropagation solution, despite the fact that it is not directly optimized for this task. Also, as a generative model the PoHMM can be used to compute a completion distribution for any of the elements of tuple, whereas feedforward networks can only perform the completion task in the direction in which they have been trained.

5 Conclusions

Using the three datasets presented here, we have shown how to fit a PoHMM that is a better model of sequences with componential structure than a single HMM with the same number of parameters. Although the number of alternative distributed hidden states in a PoHMM grows exponentially with the number of models, the computational complexity of each approximate gradient step in the fitting only grows linearly.

On a simple language modelling problem we also show that a PoHMM can capture longer range structure in a time series because the individual models do not need to explain every observation and thus they can store information about earlier parts of the sequence in their hidden states without being distracted by other regularities that are captured by other models.

Finally, we show that the PoHMM is useful for learning the symbolic family trees problem which involves finding a set of constraints which conjunctively combine to restrict the space of allowable data points. Further, we outline some future directions for research using shared output models among the HMM's to help cope with the explosion of parameters to be estimated in problems such as large vocabulary language modelling.

Acknowledgements

We thank Zoubin Ghahramani, Sam Roweis, Brian Sallans and Chris Williams for helpful discussions. This work has been supported by the Gatsby Charitable Foundation and NSERC.

References

[1] Y. Bengio and P. Fransconi. Diffusion of context and credit information in Markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

[2] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. *Machine Learning*, 29(2/3):245–273, November–December 1997.

[3] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eight Annual Conference of the Cognitive Science Society*, pages 1–12, Hillsdale, NJ, August 1986. Lawrence Erlbaum Associates.

[4] G. E. Hinton. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, 2000.

[5] C. E. Shannon. Prediction and entropy of printed english. *Bell System Techncial Journal*, 27:623–656, July, October 1948.

[6] C. K. I. Williams and G. E. Hinton. Mean field networks that learn to discriminate temporally distorted strings. In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, editors, *Connectionst models: Proceedings of the 1990 summer school*, pages 18–22, San Francisco, CA, 1991. Morgan Kaufmann.