Mean field networks that learn to discriminate temporally distorted strings

Christopher K. I. Williams
Department of Computer Science
University of Toronto
10 King's College Road
Toronto M5S 1A4, Canada

Geoffrey E. Hinton
Department of Computer Science
University of Toronto
10 King's College Road
Toronto M5S 1A4, Canada

Abstract

Neural networks can be used to discriminate between very similar phonemes and they can handle the variability in time of occurrence by using a time-delay architecture followed by a temporal integration (Lang, Hinton and Waibel, 1990). So far, however, neural networks have been less successful at handling longer duration events that require something equivalent to "time warping" in order to match stored knowledge to the data. We present a type of mean field network (MFN) with tied weights that is capable of approximating the recognizer for a hidden markov model (HMM). In the process of settling to a stable state, the MFN finds a blend of likely ways of generating the input string given its internal model of the probabilities of transitions between hidden states and the probabilities of input symbols given a hidden state. This blend is a heuristic approximation to the full set of path probabilities that is implicitly represented by an HMM recognizer. The learning algorithm for the MFN is less efficient than for an HMM of the same size. However, the MFN is capable of using distributed representations of the hidden state, and this can make it exponentially more efficient than an HMM when modelling strings produced by a generator that itself has componential states. We view this type of MFN as a way of allowing more powerful representations without abandoning the automatic parameter estimation procedures that have allowed relatively simple models like HMM's to outperform complex AI representations on real tasks.

INTRODUCTION

Neural networks have been successful at phoneme discrimination tasks, but many researchers currently feel that the best way to deal with longer duration events is to use a neural network as a front-end to a hidden markov model. The difficult task of matching models to data across temporal distortions is then handled by the hidden markov recognizer which efficiently considers all possible matches.

Despite their powerful matching and learning procedures, HMM's have a serious drawback: They implicitly assume that the ensemble of input strings is generated by a stochastic finite-state automaton and this strongly limits the types of structure that they can efficiently represent. Suppose, for example, that 20 independent binary constraints operate between the first and second half of a string and that as a result of all these separate constraints the mutual information between the two halves is 20 bits.[1] To model these 20 constraints, a hidden markov generator would need at least 2^{20} hidden states because the only way that the first half of a string can constrain the second half is via the hidden state of the generator as it finishes generating the first half.

In a hidden markov generator, 2^{20} hidden states implies 2^{20} hidden nodes. In a neural net that uses distributed representations, 2^{20} hidden state vectors only requires 20 binary hidden units. So if the mutual information between the first and second half of each string is genuinely componential, a neural network can be exponentially more efficient in representing the constraints. In effect, the only way that a HMM can deal with a set of independent constraints is to use the cross-product of the HMM's that would be needed to capture each constraint separately, so it is unable to take advantage of the fact that the constraint structure

[1] As a concrete example, we might suppose that the first half of a sentence is singular or plural, active or passive, past or present tense, abstract or concrete, etc. etc. and that the second half "agrees" with the first half along all these dimensions.

can be factorized.

Our aim is to develop a neural network alternative to HMM's that can take advantage of constraints which can be factorized by using separate hidden units to enforce separate constraints. To achieve this we have been forced to use a matching procedure that is only a heuristic approximation to the recognition procedure used in an HMM, and a learning procedure that is considerable slower than the HMM learning procedure for a comparably sized network. Eventually, we hope to show that these disadvantages are more than offset by the ability of the neural network to use a *small* representation that captures the componential structure of the constraints efficiently. In this paper, however, we only show that the neural network *can* use distributed representations to capture the componential structure and can generalize at least as well as a comparably sized HMM.

PREVIOUS APPROACHES USING NEURAL NETS

Bridle's alpha-net (Bridle, 1989) is a translation of a HMM into a recurrent neural net framework. He shows how to implement the forward pass calculations of a HMM-based recogniser (using "sigma-pi" units with a linear output) and he presents a gradient-based back-propagation training method.

As Bridle points out, this implementation of a HMM in a neural network points the way forward to other methods of constructing and training networks which offer more general non-linear structures going beyond HMM methods. Watrous (Watrous *et al.*, 1990) and Kuhn (Kuhn *et al.*, 1989) have investigated training recurrent networks for some problems in speech recognition such as phoneme discrimination, using the back-propagation learning rule. Others such as Williams and Zipser (1988) and Cleeremans *et al.* (1989) have looked at tasks which involve learning finite state automata with recurrent nets, but from the viewpoint of predicting the next symbol given left context. However, to date the issues of "time-warping" have not been directly addressed by this work.

These recurrent nets allow a "frame-by-frame" processing of the incoming data. An alternative to this is to "spatialize" time by laying out the data in an input buffer - the method used in this paper. This has a number of disadvantages, but does mean that all of the input data is readily available, whereas in recurrent nets the information on the important properties of the input must be extracted and stored in the states of the hidden units.

THE LEARNING PROCEDURE FOR THE MEAN FIELD MODULES

We assume familiarity with mean field networks and just give a brief overview here. Hopfield (1984) or Hinton (1989) give more detailed descriptions. Mean field networks use real-valued analog units with a logistic activation function that can be viewed as a deterministic approximation to the stochastic binary units used in Boltzmann machines. Mean field networks use a parallel updating algorithm to settle to a local minimum of the mean-field free energy (Hopfield, 1984). The mean field equivalent of simulated annealing is to increase the gain of each unit as the network settles. With low gains, the network will typically settle to a state in which the units have intermediate activity levels and, using an independence assumption, this state can be viewed as representing a high entropy blend of many possible binary states. In the model we describe, each such binary state in the blend represents a possible way of aligning the model of a word, stored in the tied weights, with the input data.

The input/output learning rule for MFN's (Peterson and Anderson, 1987) is based on an approximation of the Boltzmann machine learning procedure (Ackley, Hinton and Sejnowski, 1985). The replacement of stochastic binary units by deterministic real values units permits much faster learning. Below we present a different learning rule for use when the task is the classification of temporally distorted strings (which might correspond to vector-quantised time-slices of speech data) into one of N possible "word" models (word is in quotes as the entity may actually be a phoneme or some other unit - word is used for convenience only).

Each word has its own mean-field module which computes a score indicating how likely it was that that model could have generated the particular string presented. Each module has weight constraints as shown in Fig. 1 that permit dynamic time warping in a similar manner to HMMs. Then the word is classified as belonging to the model that produces the highest score. The score $q_i(y)$ for module i when presented with string y is

$$q_i(y) = e^{-F_i^*(y)} \tag{1}$$

where F^* represents the free energy of the module at a minimum of free energy. This minimum is attained by performing the mean field equivalent of simulated annealing from a high temperature to $T = 1$.

With activities of the units in the range $[0, 1]$, a mean field module has free energy

$$F = -\frac{1}{2}\sum_{i,j\neq i} p_i p_j w_{ij} + T\sum_i [p_i \ln p_i + (1-p_i)\ln(1-p_i)] \tag{2}$$

where p_i is the activation of the unit i. At a minimum

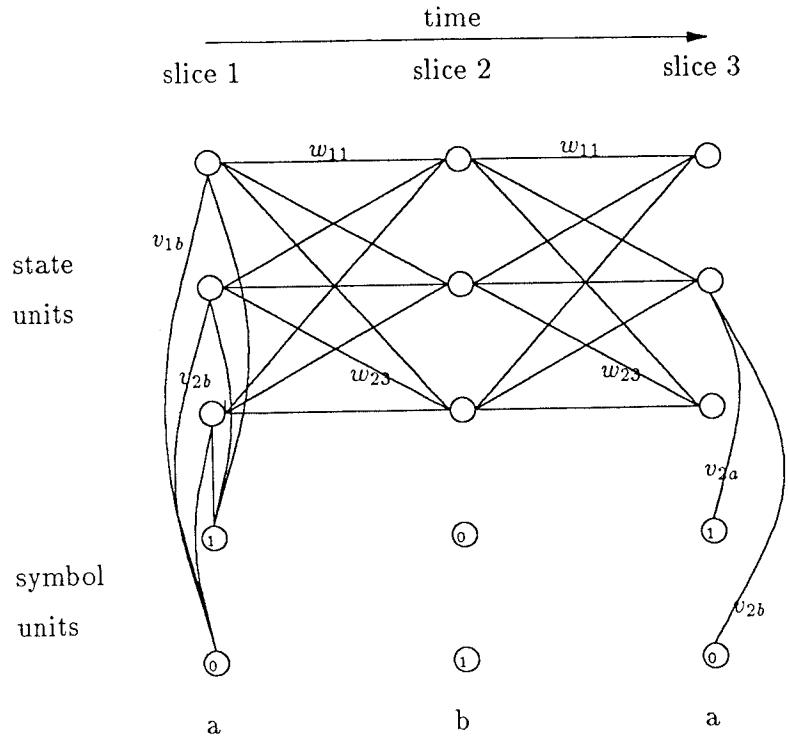

Figure 1: Part of a MFN showing some of the weight constraints (not all weights to the symbol units are shown)

of F the activities obey

$$p_i = \sigma(\frac{1}{T}\sum_j p_j w_{ij}) \tag{3}$$

where $\sigma(x)$ is the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$.

At the minimum, the derivative of F_m^* with respect to a particular weight at $T = 1$ is given by (Hinton, 1989)

$$\frac{\partial F_m^*}{\partial w_{jk}^m} = - < p_j p_k >^m \tag{4}$$

where $< >^m$ indicates that we are considering module m. The learning rule for each module is based on increasing the normalized score for those occasions when the module is the correct generator of the string, and decreasing it otherwise. Define the normalized score to be

$$r_i(y) = \frac{q_i(y)}{\sum_{j=1}^N q_j(y)} \tag{5}$$

Then the objective function to be maximized by the training is

$$B = \sum_{y \in examples} \ln r_i(y) \tag{6}$$

where i indexes the correct word class. Differentiating with respect to weight w_{jk} in module m, we get

$$\frac{\partial B}{\partial w_{jk}^m} = \sum_{examples} < p_j p_k >^m [\delta_{im} - r_m] \tag{7}$$

where δ_{im} is the Kronecker delta.

This gradient can then be used by steepest ascent techniques or more sophisticated line-search/conjugate-gradient methods to maximize B. This learning procedure has the flavour of a rule that maximizes the mutual information between the input vectors and the classification, as it maximizes the $r_i(y)$'s which take into account the scores of the other modules, rather than just maximizing the score of the correct class. In fact, if the $q_i(y)$'s represent the probability of module i producing string y, then the algorithm exactly maximizes the mutual information.

THE TASK USED IN THE SIMULATIONS

To test the learning algorithm shown above, two word models were set up using HMMs with componential structure. HMMs were used because it is easy to generate and test data with HMMs, and because the Baum-Welch training algorithm (Baum *et al.*, 1970) is optimal as a *discriminant* training procedure for data generated by HMMs (Brown, 1987). This means that we can fairly compare learning by HMMs and the neural networks.

Good datasets for discriminant tasks must have similar zero order statistics (symbol frequencies), otherwise good discrimination can be achieved by HMMs

with just one unit which simply detects the symbol frequencies, even though the generating HMMs had many more states. The data we used was generated by the "cross-product" of two three state HMMs. The state transition diagram for the three state HMMs is shown in Fig. 2

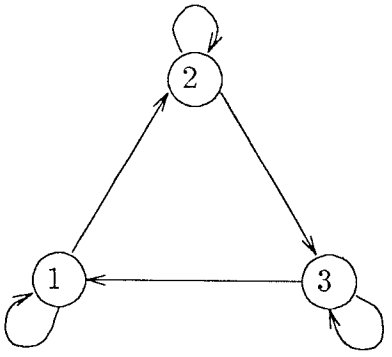

Figure 2: State transition diagram for the three state HMMs

One three state HMM called the ABC model produced symbols "a", "b" and "c" with high probability in states 1, 2 and 3 respectively. The ACB model produced "a", "c" and "b" in states 1, 2 and 3 respectively. The probability of producing the correct symbol was 0.94, and the probability of producing the other symbols was 0.03 . The transition from the start state to each of the generating states was equiprobable, so that strings generated were equally likely to begin with "a", "b" or "c".

To make componential data, one dataset was produced from the cross-product of two ABC models, and the other dataset was produced from two ACB generators. The symbols output by the cross-product HMMs are related to the two component HMMs by the following code

gen1 output : a a a b b bc c c
gen2 output : a b c a b c a b c

combined output : 1 2 3 4 5 6 7 8 9

The symbol frequencies for each of the nine symbols were similar between the data generated by the two cross-product generators. Datasets of 1000 examples of strings six symbols long generated by each cross-product HMM were used as a training set, with test and cross-validation sets also of 1000 examples each.

First order Markov models gave 28 errors on the test data. Nine state HMMs trained by the Baum-Welch algorithm gave an average of 18.25 errors and six-state HMMs gave an average of 64.68 errors with a standard deviation of 8.70 on 16 runs, with a best performance of 52 errors.

RESULTS AND DISCUSSION

Networks with six units in each hidden slice were trained on the task. Two slightly different architectures were tried. In one, the hidden units were partitioned into two fully connected groups of three units each, and the symbol units were fully connected to all units. In the other, there was no such split, all six hidden units being fully connected. The training was carried out with a conjugate-gradient with restarts algorithm, stopping when the number of errors on the cross-validation set began to increase.

The results of the simulations were 29 and 41 errors on the test data for two runs with the split weights, and 31 errors for a run with fully connected weights. These are significantly better than the results of the six state HMMs, indicating that componential structure is being discovered by the networks. Further proof of this was found by analysing the unit activities. In some of the split networks each group of three units was found to develop a distributed coding of one of the component generators' state. For the network without a split, the activities of the hidden units show that the state of one of the components is represented by the activities of all six units, the other generator's state being indicated by small modulations in these activities.

It is hard for the MFN learning rule to compete with the Baum-Welch algorithm when learning to discriminate data generated by non-componential HMMs. This is partly because the MFN learning rule is a step-size dependent method of gradient ascent, rather than a re-estimation algorithm.

With sequences generated by two three-state HMMs, the potential benefits of MFNs are small because $3+3$ is not much smaller than 3×3. We are working on further simulations with data generated by pairs of four-state HMMs, which should show the advantages of the MFNs more clearly. It is also possible to design stochastic networks with architectures similar to that of Fig. 1 that will produce data which cannot be generated by HMMs of polynomially related size.

Acknowledgements

Thanks go to Tony Plate for help with the Xerion connectionist simulator, and to Radford Neal for his HMM programs. Steve Nowlan, Evan Steeg, Tony Plate and Prof. Evangelos Milios provided helpful comments on the M.Sc thesis version of this work.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[3] J. S. Bridle. Alpha-Nets: A recurrent neural network architecture with a Hidden Markov Model interpretation. SP4 Research Note 104, Royal Signals and Radar Establishment, Great Malvern, UK, 1989. To appear in *Speech Communication* special *Neurospeech* issue, 1990.

[4] Peter Brown. *The acoustic modelling problem in Automatic Speech Recognition*. PhD thesis, Carnegie-Mellon University, 1987. Also published as IBM Research Division Technical Report RC 12750.

[5] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[6] G. E. Hinton. Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation*, *1*:143–150, 1989.

[7] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences U.S.A.*, 81:3088–3092, May 1984.

[8] G. M. Kuhn, R. L. Watrous, and B. Ladendorf. Connected recognition with a recurrent network. In *Proceedings Neurospeech*, 1989. To appear in *Speech Communication* special *Neurospeech* issue, 1990.

[9] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[10] C. Peterson and J.R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[11] R. L. Watrous, B. Ladendorf, and G. Kuhn. Complete gradient optimization of a recurrent network applied to /b/, /d/, /g/ discrimination. *J. Acoust. Soc. Am.*, 87(3):1301–1309, 1990.

[12] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. Technical Report ICS Report 8805, University of California at San Diego, 1988.