Proceedings of the International Joint Conference on Neural Networks, paper 2188, 1999

A Neural Model of Centered Tri-gram Speech Recognition

Dan Ventura, D. Randall Wilson and Brian Moncur
fonix corporation
{dventura, wilsonr, bmoncur}@fonix.com

Tony Martinez
Computer Science Department
Brigham Young University
martinez@cs.byu.edu
http://axon.cs.byu.edu

Abstract

A relaxation network model that includes higher order weight connections is introduced. To demonstrate its utility, the model is applied to the speech recognition domain. Traditional speech recognition systems typically consider only that context preceding the word to be recognized. However, intuition suggests that considering both preceding context as well as following context should improve recognition accuracy. The work described here tests this hypothesis by applying the higher order relaxation network to consider both precedes and follows context in speech recognition. The results demonstrate both the general utility of the higher order relaxation network as well as its improvement over traditional methods on a speech recognition task.

Introduction

The use of relaxation networks for optimization was introduced in the mid-eighties by Hopfield and Tank [3]. The original model considered only first-order correlations (represented in the network by pairwise interconnection weights); however, later work has been done in extending the idea to include higher-order connections [1]. This paper continues in that vein by introducing a relaxation network model that includes higher-order weight connections. To demonstrate its utility, the model is applied to the speech recognition domain, considering the problem of word recognition as an optimization task. The neural relaxation model developed here is then used to compare traditional statistics-based speech recognition techniques with more intuitively appealing models.

Traditional speech recognition systems (see for example [5]) are usually based upon a Hidden Markov Model (HMM) that is in turn based upon a left-to-right factoring of the conditional probabilities of the observations:

$$P(w_0 w_1 \cdots w_n) = P(w_0)P(w_1|w_0)P(w_2|w_0 w_1)\cdots P(w_n|w_0 w_1 \cdots w_{n-1})$$

where $P(\mathrm{x})$ is the probability of x and $P(\mathrm{y}|\mathrm{x})$ is the conditional probability of y given x. In practice, of course, computing the entire series of conditional probabilities is not feasible. This leads to n-gram language models that truncate the series after the first few terms. Most commonly, the truncation is effected after the second or third term of the series, giving rise to bi-gram and tri-gram models, respectively. The interesting point, however, is not the fact that an approximation is used but rather that the approximation considers only context *preceding* the word in question. In contrast, most practical systems process speech a phrase at a time or at least buffer several words at a time, allowing the possibility of considering both preceding as well as following context when attempting to recognize a word. Further, intuition leads us to believe that considering both preceding context as well as following context should improve recognition accuracy. The work described in this paper tests this intuitive hypothesis and demonstrates both the general utility of the higher order relaxation network as well as its improvement over traditional methods on a speech recognition task.

The rest of the paper introduces in some detail the neural model and discusses how it may be a applied to a concrete application, that of speech recognition. Empirical results of applying the network to a specific speech recognition task of medium size are reported, and these results demonstrate both the general utility of the neural model as well as an improvement in word recognition over traditional n-gram based methods.

Network Equations

The network presented here differs in three important ways from traditional Hopfield-style networks. First, the

network may contain weights between three or more nodes (higher order weights), and here we will limit ourselves to ternary weights (as well as traditional binary ones). Second, weights between nodes are anti-symmetric so that in general

$$W_{ij} \neq W_{ji}$$
$$W_{ijk} \neq W_{jki} \neq \ldots$$

Third, the node update equations incorporate the sigmoid at a different point and include a tunable relaxation parameter. The net input (incorporating only binary and ternary weights) into a node *i* at time *t* is computed as follows:

$$U_i^{(t)} = \sum_{j \neq i} V_j^{(t-1)} W_{ji} + \sum_{j \neq i} \sum_{k \neq j \neq i} \left(\frac{\left| V_j^{(t-1)} \right| + V_j^{(t-1)}}{2} \right) \left(\frac{\left| V_k^{(t-1)} \right| + V_k^{(t-1)}}{2} \right) W_{jki}$$

The first summation term represents the standard binary weights -- for a node *i*, the activations of every other node are multiplied by the weight between that node and node *i*, and the sum over all nodes represents the net input to node *i*. The double summation term represents the additional ternary weight connections and the basic idea is the same as for the binary weights. The only modification is the way in which we treat the activations of the two impinging nodes. The product of ratios guarantees that the input through a ternary weight into node *i* is only significant if both the impinging nodes' activations are high.

The activation of node *i* at time *t* is computed as

$$V_i^{(t)} = V_i^{(t-1)} + \rho\left(\sigma\left(U_i^t\right) - V_i^{(t-1)}\right)$$

where ρ is the relaxation rate parameter, and σ is any appropriate sigmoid-type function. We prefer a sigmoidal function with range [-1,1] such as

$$\sigma\left(U_i^t\right) = \tanh\left(\frac{U_i^t}{\upsilon}\right)$$

so that a value of 0, which is often problematic in neural networks, can represent a "don't know" state. For further discussion of this kind of relaxation network see [6][7][8].

Implementing Speech Recognition

The relaxation network can be used to implement a speech recognition engine as follows. A word sequence may be modeled as a sequence of network nodes. Excitatory weight connections between neighboring words (in the temporal sense) can be based upon n-gram statistics calculated from a training corpus. Additionally, words competing for the same temporal position do so through inhibitory connections. Initially, word node activations are set by an acoustic model. Here we use a Viterbi-based [2] dynamic programming method coupled with an HMM scoring mechanism.

Different network configurations representing various methods of speech modeling can easily be constructed, for example (a) networks implementing a traditional bi-gram language model, (b) networks implementing an improved bi-gram model that employs both preceding and following contextual information, (c) networks implementing a traditional tri-gram language model (using higher order weights) and (d) networks implementing a centered tri-gram language model (using higher order weights) that again considers both preceding and following contextual information. Figure 1 illustrates a small part of each of these four networks. In the figure, the darkened node is the node of interest and only those weights impinging on that node are shown.

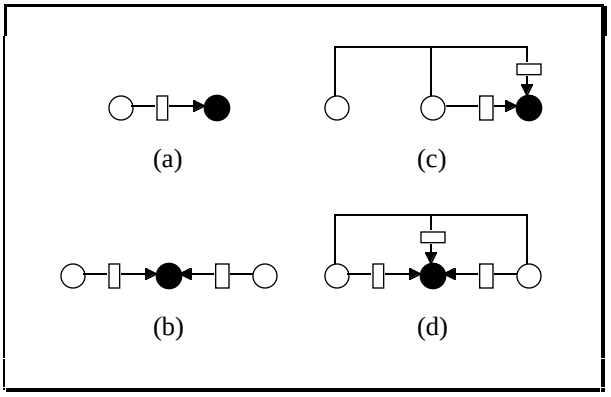

Figure 1. Networks based on (a) traditional bi-grams, (b) improved bi-grams, (c) traditional tri-grams and (d) centered tri-grams

As a more concrete example, and to illustrate the differences between traditional and centered language models, consider the phrase, "I love tasty ribs". Figure 2 illustrates a network representing this phrase using weights based on traditional bi- and tri-gram statistics.

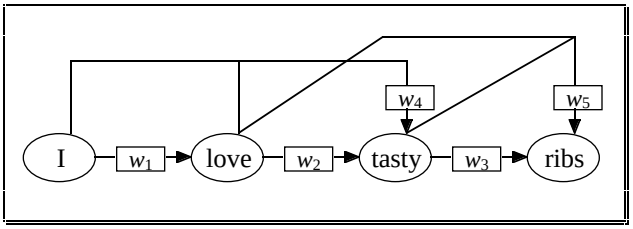

Figure 2. Network implementing traditional bi-gram and tri-gram model for the phrase "I love tasty ribs"

In Figure 2,

$$w_1 = f\left(P(\text{love}|\text{I}), P(\text{love})\right)$$
$$w_2 = f\left(P(\text{tasty}|\text{love}), P(\text{tasty})\right)$$
$$w_3 = f\left(P(\text{ribs}|\text{tasty}), P(\text{ribs})\right)$$
$$w_4 = f\left(P(\text{tasty}|\text{I love}), P(\text{tasty})\right)$$
$$w_5 = f\left(P(\text{ribs}|\text{love tasty}), P(\text{ribs})\right)$$

where f is a mapping function designed to closely relate the network weights to the training corpus statistics, and several different functions have been used in the experiments. One example of a typical weight function is

$$f\left(P(y|x), P(y)\right) = \min\left(1.0, \frac{N(xy)}{\tau}\right) \cdot \max\left(\mu, \log\left(\frac{P(y|x)}{P(y)}\right)\right)$$

where $N(xy)$ signifies the number of times the word sequence xy was seen in the corpus, τ is an integer threshold for the minimum number of times a sequence should be seen in the corpus to be statistically significant, and μ is a minimum base value for the weights. The key term is the log of the probability ratio. Taking the *max*() of this value and μ guarantees that the weight value will not get too small. Multiplying by the *min*() term scales the weight according to how often the corresponding word sequence was seen in the corpus. If it has been seen fewer than τ times, the weight is scaled back accordingly.

In contrast, Figure 3 shows a network representing the same phrase using improved bi-gram and tri-gram models. Notice that from a purely probabilistic standpoint, this model does not represent a proper decomposition as there are cycles in the network graph.

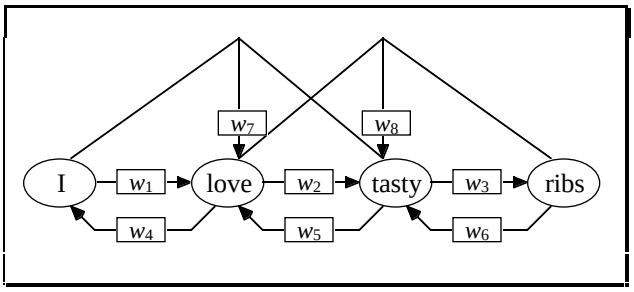

Figure 3. Network implementing centered bi-gram and tri-gram model for the phrase "I love tasty ribs"

The weights in Figure 3 are computed in a similar manner as those in Figure 2, with the notable exception that we must introduce the subscript notation on the conditional statements. Here the subscript p indicates a *preceding* word while a subscript f indicates a *following* word.

$$w_1 = f\left(P\left(\text{love}\middle|\text{I}_\text{p}\right), P(\text{love})\right)$$
$$w_2 = f\left(P\left(\text{tasty}\middle|\text{love}_\text{p}\right), P(\text{tasty})\right)$$
$$w_3 = f\left(P\left(\text{ribs}\middle|\text{tasty}_\text{p}\right), P(\text{ribs})\right)$$
$$w_4 = f\left(P\left(\text{I}\middle|\text{love}_\text{f}\right), P(\text{I})\right)$$
$$w_5 = f\left(P\left(\text{love}\middle|\text{tasty}_\text{f}\right), P(\text{love})\right)$$
$$w_6 = f\left(P\left(\text{tasty}\middle|\text{ribs}_\text{f}\right), P(\text{tasty})\right)$$
$$w_7 = f\left(P\left(\text{love}\middle|\text{I}_\text{p},\ \text{tasty}_\text{f}\right), P(\text{love})\right)$$
$$w_8 = f\left(P\left(\text{tasty}\middle|\text{love}_\text{p},\ \text{ribs}_\text{f}\right), P(\text{tasty})\right)$$

In addition to the inputs shown to the nodes in Figures 2 and 3, inhibitory connections between competing candidates may be incorporated. This may be done at the word level, the phrase level, or both. When the network has relaxed, the candidate with the highest activation is the result of the recognition. Again, this recognition may be done at the word or at the phrase level.

Results

Table 1 summarizes the results of applying the various neural models to the DARPA Resource Management task [4]. This application consists of a set of 7200 queries about naval resources over a 1000 word vocabulary. Using weights based on traditional precedes-only bi-gram probabilities, a baseline error rate of 11.68% in word recognition accuracy is achieved. Using weights based on the improved bi-gram model that includes following contextual information, the error rate is reduced by 11.82%. As expected, adding higher order weights further improves the accuracy. Weights based upon traditional tri-grams result in an improvement of 28.76% over the baseline error. Finally, using weights based upon centered tri-grams reduces the baseline error rate by 37.76%.

Table 1. Word Recognition Error Rates

weights based on traditional bi-grams:	11.68%
weights based on improved bi-grams:	10.30%
weights based on traditional tri-grams:	8.32%
weights based on centered tri-grams:	7.27%

Conclusions

A method of incorporating higher order connections into relaxation networks is introduced. Using this method a neural model has been developed for processing temporal

patterns, in particular spoken phrases. In contrast to traditional speech processing based upon HMM technologies, which considers only context *prior* to a given word, the neural model presented here considers context both *preceding and following* a word. The model represents word correlations as weights based upon preceding bi-gram, following bi-gram and centered tri-gram statistics. Empirical results demonstrate that this neural alternative to n-gram language models improves over the traditional method in terms of reducing word recognition error on a real-world speech recognition task.

Ongoing work includes improving the initial acoustic information provided to the word nodes, incorporation of higher-level language information (such as grammar) into the network, modification of the word nodes squashing function, improving the weight function, and developing learning algorithms for dynamically optimizing network performance.

References

[1] Cooper, B.S., "Higher Order Neural Networks -- Can They Help Us Optimize?", *Proceedings of the 6th Australian Conference on Neural Networks*, pp. 29-32, 1995.

[2] Forney, G., "The Viterbi Algorithm", *Proceedings of the IEEE*, v61, pp. 268-278, 1973.

[3] Hopfield, J.J. and D.W. Tank, "Neural Computations of Decisions in Optimization Problems", *Biological Cybernetics*, v52, pp. 141-152, 1985.

[4] Price, P., W.M. Fisher, J. Bernstein and D.S. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 651-654, April 1988.

[5] Rabiner, L. and B-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1989.

[6] Wilson, D.R., Dan Ventura, Brian Moncur and Tony Martinez, "The Robustness of Relaxation Rates in Constraint Satisfaction Networks", *Proceedings of the International Joint Conference on Neural Networks* (this proceedings), 1999.

[7] Zeng, Xinchuan and Tony Martinez, "A New Relaxation Procedure in the Hopfield Network for Solving Optimization Problems, *Neural Processing Letters*, to appear, 1999.

[8] Zeng, Xinchuan and Tony Martinez, "Improving the Performance of the Hopfield Network by Using a Relaxation Network", *Proceedings of the International Conference on Neural Networks and Genetic Algorithms*, to appear, 1999.