Proceedings of the International Joint Conference on Neural Networks, pp.934-938, 2002

Probabilistic Connections in Relaxation Networks

Dan Ventura
Computer Science Department
Brigham Young University
ventura@cs.byu.edu
http://axon.cs.byu.edu/Dan

Abstract – This paper reports results from studying the behavior of Hopfield-type networks with probabilistic connections. As the probabilities decrease, network performance degrades. In order to compensate, two network modifications – *input persistence* and a new activation function – are suggested, and empirical results indicate that the modifications significantly improve network performance.

I. INTRODUCTION

This work considers the problem of probabilistic connections in artificial neural networks similar to those first introduced by Hopfield and Tank [1]. Probabilistic connections can model the dynamics of a spatially distributed network or, alternatively, a network exhibiting certain types of pathologies.

Another motivation for considering neural networks with probabilistic connections is that of modeling a group of cooperating agents as a distributed constraint satisfaction network or associative memory. The stored vectors represent patterns of activity among the agents. Agent activities will result in changes to the network inputs, and when the input pattern changes are great enough, the network will converge to a different pattern of activity for the individual agents. A simple application of this concept is area coverage by a group of mobile agents, for example as a distributed sensing network. Symmetric, equal sized weights could represent equal spacing with network activations representing physical position of the agent. In the event of agent loss, the network will no longer be in equilibrium. This change is propagated through the network until a new equilibrium is reached. Since the weights are fixed, this is accomplished through the modification of the node activations, which means that the agents' physical locations will change.

In general, the weights in the network would be a function of agent attributes (for example, power, power consumption, location, sensor data, id, etc.) and problem constraints (such as time limits, power limits, number of agents, spatial limitations, etc).

Of course, we can consider network activations to represent agent behaviors other than position and, in fact, the activation could be a vector with each component representing a candidate action. The vector component with the highest activation at a given time could be considered the behavior appropriate for that agent at that time. Alternatively, *any* vector component with positive activation could be considered an appropriate behavior at that time and all such behaviors could be performed in parallel or with some probability.

In such a scenario, some connections in the network (those between agents) may be less reliable or operate on a different time scale than others (those within an agent). Such heterogeneity can be modeled by introducing a probability term k associated with each network connection. We define a probabilistic connection as a connection that is only active with probability $1/k$ at any given time step during network relaxation.

This paper studies the effects of heterogeneous connections in simple constraint satisfaction networks. In particular, it considers the case for which there are two distinct types of connections within the network – those for which $k = 1$ and those for which $k > 1$.

Two simple networks are employed to perform experiments testing network stability and network reliability as the value of k is increased for some nodes in the network. As expected, network performance degrades as k increases. Introducing input persistence or memory mitigates to some extent this performance degradation. Combining input persistence with a modified activation function significantly improves network performance, almost completely compensating for the deficit due to unreliable connections.

II. NETWORK EQUATIONS

The networks presented here differ somewhat from traditional Hopfield-style networks. The node update equations incorporate the sigmoid at a different point and include a tunable relaxation parameter. The net input into a node i at time t is computed as follows

$$U_i^{(t)} = \sum_{j \neq i} V_j^{(t-1)} W_{ji} \qquad (1)$$

For a node i, the activations of every other node j are multiplied by the weight between node j and node i, and the

sum over all nodes j represents the net input to node i. In order to incorporate probabilistic connections, a value k_{ij} is associated with each weight and Equation 1 is modified as

$$U_i^{(t)} = \sum_{j \neq i} V_j^{(t-1)} W_{ji} \delta_{ji} \qquad (2)$$

where $\delta_{ij} = 1$ with probability $1/k_{ij}$ and $\delta_{ij} = 0$ with probability $1\text{-}1/k_{ij}$. The activation of node i at time t is computed as

$$V_i^{(t)} = V_i^{(t-1)} + \rho\left(\sigma\left(U_i^{(t)}\right) - V_i^{(t-1)}\right) \qquad (3)$$

where ρ is the relaxation rate parameter, and σ is any appropriate squashing function. Here we employ a sigmoidal function with range [-1,1] so that a value of 0 can represent a "don't know" state:

$$\sigma(U) = \tanh\left(\frac{U}{\upsilon}\right) \qquad (4)$$

where υ controls the gain of the sigmoid.

Similar modifications have been suggested elsewhere. The utility of this type of network relaxation has been shown to improve performance on optimization problems [2], and networks of this type have been shown to be robust with respect to the relaxation rate parameter below a threshold [3]. A modified activation function has also been introduced and shown to provide improved network stability and performance [4]. The utility of these kinds of modifications to traditional Hopfield networks has also been demonstrated for practical applications such as speech recognition [5]. For related discussion of this kind of relaxation network see [6][7].

III. EXPERIMENTAL RESULTS

Figures 1 and 2 show the two networks used for investigating the effect of probabilistic connections on network stability and fidelity. The first is a 4-node network with two reliable connections (indicated as solid lines in the figure) and four probabilistic connections (indicated as dashed lines). From the standpoint of interacting agents, this network can be considered to represent the cooperative dynamics of two simple agents. Alternatively, we may consider the network to represent a distributed constraint satisfaction system or a Hopfield-type network exhibiting some pathology.

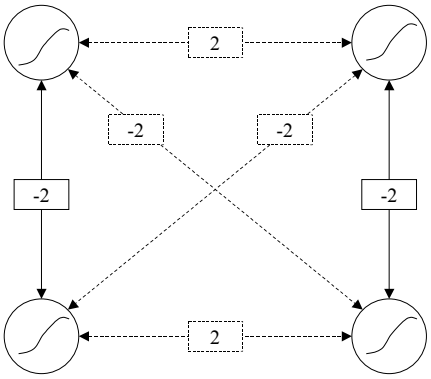

Figure 1. 4-node network with two connections with $k = 1$ (solid) and four connections with $k > 1$ (dashed).

Assuming all the weights were reliable ($k = 1$), the network provides two basins of attraction: one with characteristic pattern

[-1, 1, -1, 1]

and one with characteristic pattern

[1, -1, 1, -1]

If we limit our consideration to the 16 unique bipolar patterns, besides these two trivial cases each basin attracts four other patterns (those with Hamming distance 1 from the characteristic patterns). The remaining six patterns converge to the spurious stable state

[0, 0, 0, 0]

because of the symmetry of the network weights and the fact that they are Hamming distance 2 from both characteristic patterns. These patterns will be referred to as *symmetric* while those that converge to a characteristic pattern will be called *asymmetric*.

In order to quantify network behavior with respect to k, this network behavior at $k = 1$ will be considered the target behavior. The experiments investigate the network's deviation from this "ideal" behavior as we increase the value of k for the probabilistic network connections.

Figure 2 shows a 6-node generalization of the network with three reliable connections and 12 probabilistic ones, possibly representing three simple cooperative agents. The network weights are again set to encode two basins of attraction with characteristic patterns

[-1, 1, -1, 1, -1, 1]

and

[1, -1, 1, -1, 1, -1]

Of the 64 possible bipolar patterns, 22 (including the characteristic pattern) converge to one of the basins, 22 converge to the other and 20 again converge to the pattern

[0, 0, 0, 0, 0, 0]

which may reasonably considered as a "don't know" state.

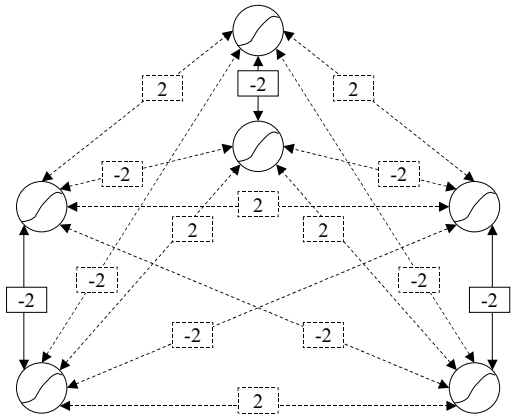

Figure 2. 6-node network with three connections with $k = 1$ (solid) and twelve connections with $k > 1$ (dashed).

The network equations depend on two parameters: the relaxation rate ρ and the gain υ. Figure 3 shows the results of experimenting with a 4-node network with various relaxations rates, tracking average pattern error vs. k. To facilitate readability, results have been translated along the y axis. For larger learning rates, the network is unstable; but for $\rho \leq 0.0001$ the network dynamics are robust (in good agreement with the results in [3]).

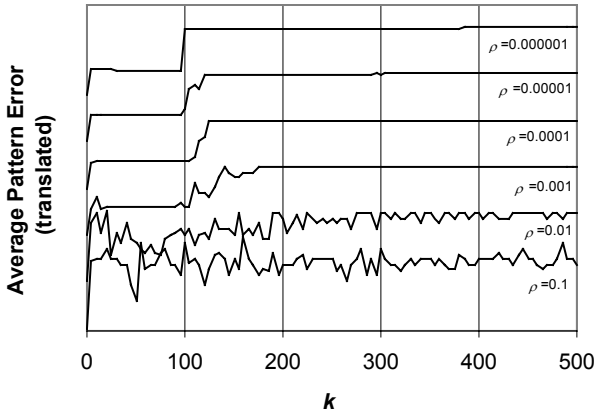

Figure 3. Average pattern error vs. k for various relaxation rates.

The network dynamics were also found to be robust over a range of values for the gain υ. For all the experiments reported, we used the values

$$\rho = 0.0001$$
$$\upsilon = 0.1$$

Each experiment measured the effect of k on average pattern error (as measured against the "ideal" network behavior when $k = 1$). For each value of k, each possible bipolar pattern is introduced into the network and the network is allowed to relax to equilibrium. The first experiment employs Equations 2-4 to simulate a standard network. As k is increased, for any given time step input to a node is often incomplete in the sense that some connections that should contribute do not. As expected, this has a significant effect on network performance (see Figure 4).

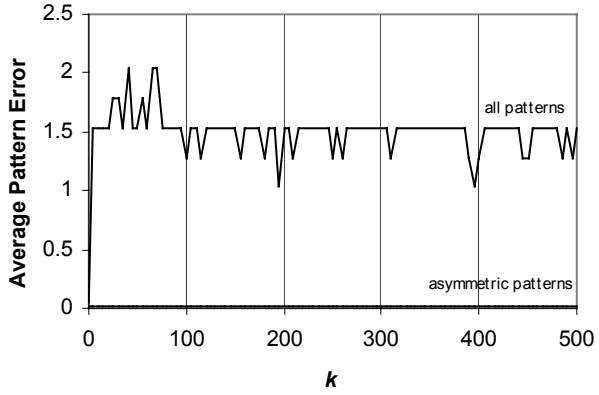

Figure 4. Average pattern error vs. k for standard network. The upper curve includes all patterns. The lower curve includes only those patterns that are *not* "don't know" patterns.

Because the error is due to the fact that some connections are not regularly contributing to the input summation, the natural way to attempt to remedy the situation is to somehow compensate for the missing input. Implementing *input persistence* can be accomplished by replacing Equation 2 with

$$U_i^{(t)} = \sum_{j \neq i} I_{ji}^{(t)} \tag{5}$$

where

$$I_{ji}^{(t)} = \begin{cases} V_j^{(t-1)} W_{ji} \delta_{ji} & \text{if } \delta_{ji} = 1 \\ I_{ji}^{(t-1)} & \text{otherwise} \end{cases} \tag{6}$$

and

$$I_{ji}^{(0)} = 0 \tag{7}$$

Equations 6-7 ensure that the summation term almost always is reasonably close to the value it would be if $k = 1$. Equation 7 says that for node i the initial input from any node j is 0 and

Equation 6 says that unless the connection from node j to node i is active at time t, then the input from node j to node i is the same as it was at time t-1. Therefore, after an initial lag all inputs to a node will be nonzero and on average no more than k time steps "out of date".

Figure 5 compares the average pattern error using the input persistence of Equations 5-7 with that of the original network. The results shown with a broken line reproduce the upper curve from Figure 4. The solid line indicates network performance incorporating input persistence. Note that although performance is not quantitatively improve, it is improved qualitatively as the network exhibits better stability.

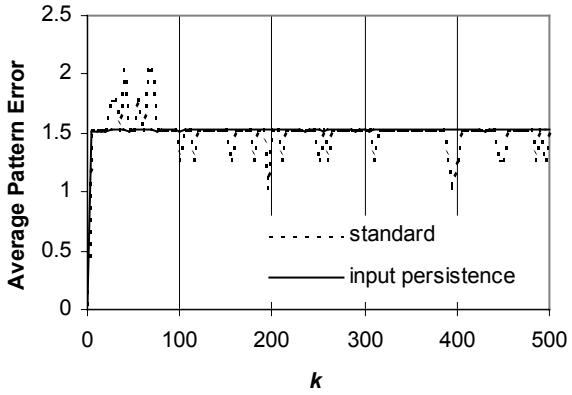

Figure 5. Average pattern error vs. k for standard network and for network with input persistence.

Interestingly, the symmetric patterns contribute almost all of the error (refer back to Figure 4). If we consider only asymmetric patterns (lower curve in Figure 4), there is almost no error even for large values of k. In other words, the network error is due to the fact that the symmetric patterns are not converging to the "don't know" pattern [0, 0, 0, 0]. The ideal network (for which all connections have k = 1) converges to "don't know" for the symmetric patterns because its weights are symmetric *and* because all connections contribute to the input summation every time step. For networks with probabilistic connections this second condition is violated and the symmetry is broken. To understand why the symmetry is so fragile, consider the activation function given in Equation 4. It is most unstable around 0. Therefore, even slight perturbations to the network dynamics will have drastic (and recoverable effects). As has been noted in [4], this is not intuitively reasonable if 0 is to act as "don't know" – lack of confidence should instead entail little effect. In other words the activation function should not be unstable around 0.

Using a approach similar to [4], Equation 4 can be replaced with a sigmoidal function that introduces a tunable stable region around 0:

$$\sigma(U)=\begin{cases}\dfrac{\tanh\left(\dfrac{U-\gamma}{\upsilon}\right)+\tanh\left(\dfrac{\gamma}{\upsilon}\right)}{1+\tanh\left(\dfrac{\gamma}{\upsilon}\right)} & \text{if } U\geq 0\\[2ex] \dfrac{\tanh\left(\dfrac{U-\gamma}{\upsilon}\right)-\tanh\left(\dfrac{\gamma}{\upsilon}\right)}{1+\tanh\left(\dfrac{\gamma}{\upsilon}\right)} & \text{if } U<0\end{cases} \tag{8}$$

with two unstable regions centered around γ and $-\gamma$. Figure 6 shows an example of this improved activation function with γ = 3 and υ = 0.8.

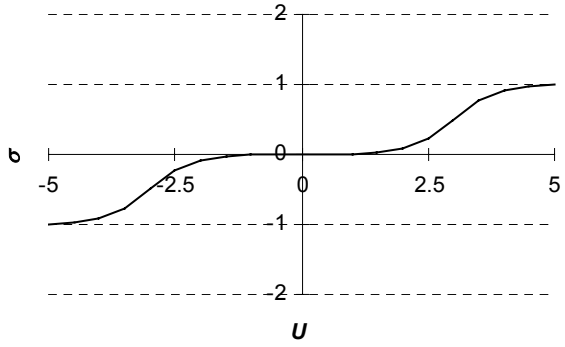

Figure 6. Improved activation function with stable region around 0.

What value should γ take? It must be large enough to ensure stability around 0 in order to restore symmetry to the "don't know" patterns. However, if γ is too large, some asymmetric patterns may experience an artificial equilibrium around the pattern [0, 0, 0, 0]. Figure 7 demonstrates this, showing error results using the improved activation function (together with input persistence) for three different values of γ. Note the excellent network performance and stability when γ = 2. In contrast, with γ = 1 the network is very unstable and exhibits little error reduction; and with γ = 3, although the network maintains its stability error is actually increased significantly (even over the networks employing the original sigmoid function – refer back to Figure 5).

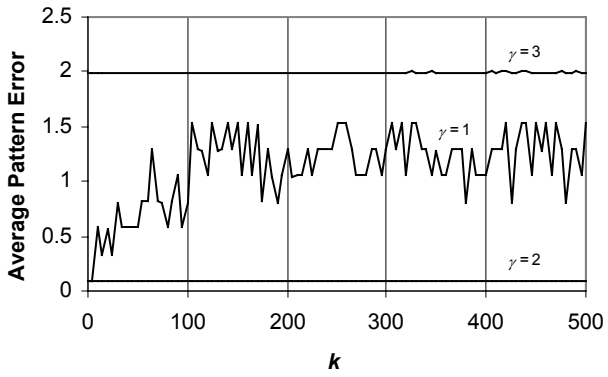

Figure 7. Average pattern error for input persistent network employing improved activation function for three different sizes of the stable region around 0.

Figure 8 summarizes the results of repeating the experiments on the 6-node network, reporting average pattern error for a standard network, for a network with input persistence and for a network with both input persistence and one of four different versions of the improved activation function. As before, adding input persistence improved performance over the standard network and employing the new activation function further improved network performance, producing excellent results for the right value of γ. Note that for the 6-node network, $\gamma = 3$ is the best value. Network performance and stability are very similar for the 6- and 4- node networks suggesting that the results scale.

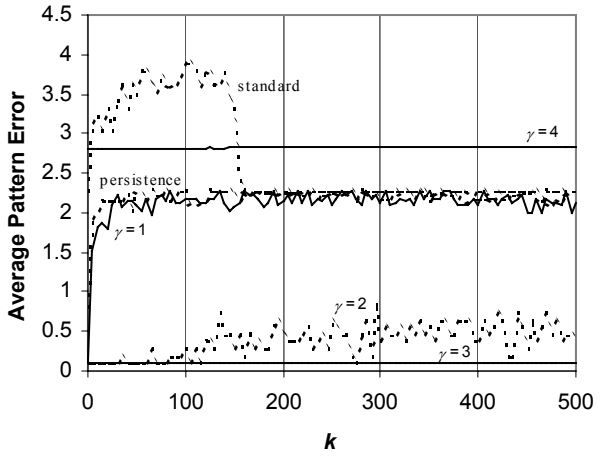

Figure 8. Average pattern error for 6-node networks. Results are shown for the standard network, a network employing input persistence and a network employing both input persistence and the improved activation function for four different sizes of the stable region around 0.

IV. CONCLUSION

Probabilistic delays in constraint satisfaction networks can be used to model network pathologies, distributed network processing or a group of cooperative agents. The probabilistic connections can severely degrade network performance and we have suggested two modifications – input persistence and the use of a modified activation function – that together appear to adequately compensate. Empirical results from experiments with small networks are favorable and suggest that the techniques may scale to larger networks. The improved network is sensitive to the value of the parameter γ and further work must be done to discover the appropriate value for γ for a particular network. Will γ be tied closely to the number of "partitions" in the network? Also, the maximum k for which stability can be maintained in the network must be related to the relaxation rate. If statistics on connection activity could be maintained, the network relaxation rate could be optimized to balance speed with network performance. Future work will include evaluating these results in the context of a network partitioned by delayed connections rather than probabilistic ones. Can these results or modifications thereof be applied to overcome connection delays? Could the network learn those delays and compensate for them or better yet, make use of them (perhaps for some type of temporal processing)?

V. REFERENCES

[1] Hopfield, J.J. and D.W. Tank, "Neural Computations of Decisions in Optimization Problems", *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.

[2] Zeng, Xinchuan and Tony Martinez, "A New Relaxation Procedure in the Hopfield Network for Solving Optimization Problems", *Neural Processing Letters*, vol. 10, pp. 1-12, 1999.

[3] 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*, paper 162 (CD-ROM), July 1999.

[4] Zeng, Xinchuan and Tony Martinez, "A New Activation Function in the Hopfield Network for Solving Optimization Problems", *Proceedings of the International Conference on Neural Networks and Genetic Algorithms*, pp. 67-72, April 1999.

[5] Ventura, Dan, D.R. Wilson, Brian Moncur and Tony Martinez, "A Neural Model of Centered Trigram Speech Recognition", *Proceedings of the International Joint Conference on Neural Networks*, paper 2188 (CD-ROM), July 1999.

[6] 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*, pp. 73-77, April 1999.

[7] Zeng, Xinchuan and Tony Martinez, "Extending the Power and Capacity of Constraint Satisfaction Networks", *Proceedings of the International Joint Conference on Neural Networks*, paper 190 (CD-ROM), July 1999.