Distributed Neural Network Routing

Joshua Menke

Researchers have shown machine learning techniques are able to perform just as well as more traditional algorithms on several problems. The following research demonstrates the ability of a simple single-layer neural network to find the shortest paths from each graph vertex to another when applied in a distributed fashion. The purpose being to show that a distributed neural network routing scheme can perform at least as well as the traditionally used Dijkstra's algorithm.

Introduction

Common machine learning approaches successfully applied to the problem of communication network route finding include the network topology suggested by Hopfield and Tank in [1] as used by Lee and Chang in [4], the DIANNER approach described by Tseng in [2], and the reinforcement learning approach used by Boyan and Littman in [3]. The following research seeks to reproduce Tseng's research in [2] in order to better understand the details involved in the implementation.

Methods

The following explanation is based on assumptions made about Tseng's description with some modifications that improved performance in the initial stages.

Running

Each router in the communication network consists of a single-layer neural network. A detailed explanation of neural networks is beyond the scope of this paper (but can be found in [5]); suffice it to say that each neural network has an input for every possible destination in the communication network and an output for every port leading out of the router. When a packet arrives the destination is placed on the inputs of the neural network and an estimate of the cost of the path from the current router to the destination is output on each corresponding port. In other words, the output for port x will be how long the neural network estimates it will take a packet to travel from this router to its destination through port x. The packet would then be sent on the port corresponding to the lowest output.

Training

Before a router's neural network can give meaningful outputs, it must first "learn" the costs of each path it could take to every possible destination. The initial outputs of the neural network will be completely random. To improve the path costs estimates, the router will send out "training packets" on every port to every destination. These packets can be routed using the initial random configuration of the network, or using a broadcast protocol. If the former is used, the packets will need a time to live parameter and a time out will be necessary on the router because the random initial configuration will almost assuredly contain routing loops. In the implementation used for this paper, a time out was used on the sending router causing the estimate to change by one more than its current estimate if it timed out. If a broadcast protocol is used to send the packets, more network congestion will result, but the packets are more likely to arrive at their destinations.

To obtain the time estimate for this implementation, the training packets summed the amount of time taken across each link as they traveled through the network. Other possible cost estimation schemes would include time stamping or queuing theory. Once a training packet arrives, the destination router sends back an acknowledgement packet containing the actual cost estimate (this may also be done using broadcast to make sure it arrives). When the sending router gets acknowledgement packet back, it updates the current estimate through the corresponding port using a basic neural network learning algorithm (see [5]).

The process for the entire neural network routed communication network basically functions like this:

1. Each router is initialized with random estimates
2. Each router sends out training packets to every other router through every port and updates its current estimate when the acknowledgement packets return.
3. Each router repeats step 2 until all current estimates are the same as the estimates in the acknowledgement packets.
4. Each router can repeat steps 2 and 3 at regular intervals to keep the paths current.

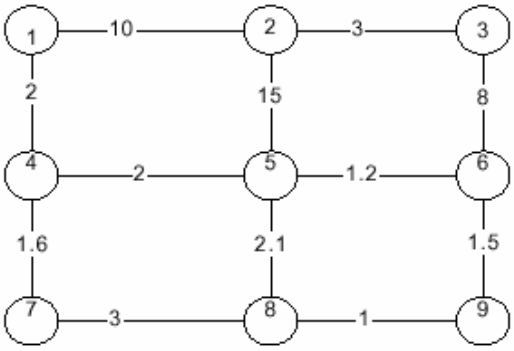
There is one tunable parameter involved in the training cycle called the learning rate. For the main experiments in this paper, the learning rate was set to 1. The learning rate affects how fast the network corrects its estimates and should never be more than 1 (for example, 1 means change the estimate to be exactly what the ack packet reports and .5 means change the current estimate by only half the difference between the current and the ack estimate). A network will find the paths fastest with a learning rate of 1, assuming the ack estimates are accurate. If the ack estimates are less accurate, or if it is desirable to have the network "adapt slowly" to changing conditions, a lower learning rate should be used.

The time complexity for training is basically the number of iterations to train times the number of routers in the network times the average number of ports on each router. The number of training iterations is generally considered to be constant (as seen below, a fairly low one) and the average number of ports on a router is also—on average—a small

constant. Therefore, assuming the number of iterations is not related to the number of routers (which has yet to be tested) the time complexity is basically linear in the number of nodes in the network, which is simpler than Dijkstra's algorithm's quadratic time for solving the same problem.

Experiment

The techniques explained above were applied to the 9-node mesh network shown below:



Each vertex in the above graph represents a neural network router and the labels on the edges represent the costs between each connected router.

After running the routers on the above network, congestion was simulating by resetting the following links to the following values, and the network was run again (Tseng did the same thing in [2]):

- node 2 to node 5 changes from 15 to 20,
- node 4 to node 5 changes from 2 to 21
- node 6 to node 5 changes from 1.2 to 10
- node 8 to node 5 changes from 2.1 to 11

Results

Using a learning rate of 1, every router learned the optimal paths to every other router in the above network after 6 iterations of training. After "congestion" was added, the network learned the new optimal paths after 4 more iterations. Tseng used a learning rate of .5 and reported optimal paths after 9 iterations and adapted to congestion after another 11.

Conclusion and Future Work

The above results support the hypothesis that the above distributed neural network routing system can learn paths through a network as well as current traditional methods (i.e. OSPF). If the number of iterations needed to learn the paths remains relatively constant for larger communication networks, this new routing method will yield better time complexity than strictly quadratic for networks greater than 18 nodes. This estimate ignores the constants on common quadratic methods (like Dijkstra's algorithm), which could mean this algorithm is better on even smaller networks.

Future work can be done in two areas, the first being validation. This would include running the same experiment on several different topologies each with several different configurations. The next step would be to test it with a discrete time simulator, and finally on real networks. One of the most important observations would be to determine if the number of iterations needed to learn the networks increases with larger networks (or with networks with a higher average number of ports per router). If these numbers stay relatively constant, the complexity will remain linear.

Another area for future work would be to extend the network's current capabilities to more interesting routing problems like multi-criterion path selection, especially if the time complexity reduction holds. Success in this area would add to the constantly growing number of instances where machine learning simplifies many of the more difficult problems applying more traditional algorithms to important problems.

References

[1] Hopfield J.J., Tank D.W., (1985). Neural Computation of Decisions in Optimization Problems. *Biological Cybernetics 52* (1985) 141.

[2] Chiu-Che Tseng and Max Garzon, Hybrid Distributed Adaptive Neural Router, Proceeding of the ANNIE '98

[3] Boyan, J. A., and Littman, M. L. 1994. *Packet routing in dynamically changing networks: A reinforcement learning approach*. In Cowan, J. D.; Tesauro, G.; and Alspector, J., eds., Advances In Neural Information Processing Systems 6. Morgan Kaufmann Publishers.

[4] Lee S.L., Chang S., (1993) *Neural Networks for routing of Communication Networks with Unreliable Components.* IEEE Transactions on Neural Networks 4:5, pp.854-863.

[5] Mohamad H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, MA, 1995.