Memory Neural Networks

David Biggs and Andrew Nuttall
Department of Aeronautics and Astronautics
Stanford University

Motivation

Neural Networks have been used for image processing, natural language processing, and vehicle control. Recently they have become more popular with the advent of 'deep learning', along with such algorithms as recurrent neural networks and long short term memory (LSTM) networks. The trend with research groups are pushing towards competing with human level pattern recognition. The current hurdle is in creating a framework for neural networks to base future decisions on past knowledge. Indeed, recent work by groups led by Weston et al. (2014) and Graves et al. (2014) have looked at introducing more concrete memory structures. In this poster we present a similar such work.

A simple implementation of memory would be to use an external memory and use this to augment the input into a neural networks.

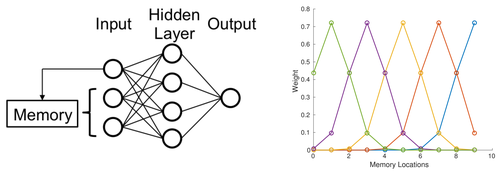


Figure 1: Augmented feed-forward neural network and memory weights

This architecture was implemented with a noisy signal prediction data set. A feed-forward neural network has its input augmented with a memory bank that outputs a weighted distribution of stored values. The network is trained with a single sinusoid then tested with a sum of three sinusoids of varying frequencies with moderate results.

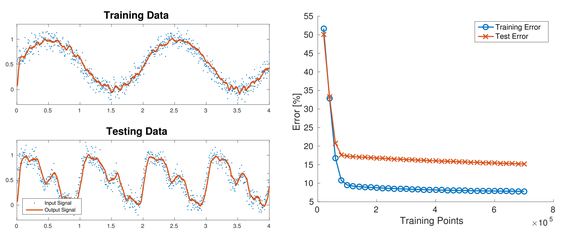


Figure 2: Noisy signal prediction with an augmented feed-forward neural network

The limitation with external memory implementations is that the neural network is that the memory architecture is predetermined, which limits the functionality. A better implementation would store memories in an orthogonal basis within the network to allow the network to fully learn how to utilize memory.

Algorithm

The memory neural network algorithm has a feed-forward neural network with neurons in hidden layers that store and output memory. Training and testing of the algorithm is comprised of three key steps:

- *Forward propagation* of the input across each network layer, resulting in the output
- *Back propagation* of the errors across each layer starting at the output to perform gradient descent updates on network parameters
- *Memory storage* in each layer of nearby neurons outputs as an orthogonal set in a higher dimensional space

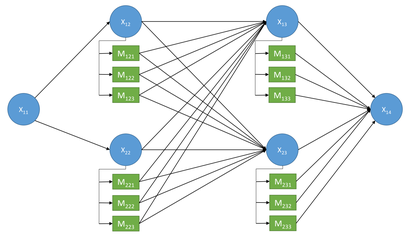


Figure 3: Memory Neural Network Neural Diagram

Forward Propagation: Each layer of the network first calculates its node's primary output

$$x^{(i)} := f\left(W^{(i-1)}x^{(i-1)} + b^{(i-1)}\right)$$

then concatenates the output with weighted memory

$$x^{(i)} := \left[x_1^{(i)},\ x_1^{(i)}M_1^{(i)\intercal},\ ...,\ x_k^{(i)},\ x_k^{(i)}M_k^{(i)\intercal}\right]^\intercal$$

Memory Storage: The output of each hidden layer is stored in a higher dimesion in sets, where $g(x) : \mathcal{R}^k \rightarrow \mathcal{R}^m$,

$$M_{j:j+m}^{(i)} = g\left(\left[x_j^{(i)},\ x_{j+1}^{(i)},\ ...,\ x_{j+k}^{(i)}\right]\right)$$

Back Propagation: The parameters of the network are updated using gradient descent – back propagation starts at the output with error

$$\delta^{(\text{output})} = (x^{(\text{output})} - y)x^{(\text{output})}(1 - x^{(\text{output})})$$

The errors of the subsequent layers are weighted by the parameters

$$\delta^{(i)} = W^{(i+1)}\delta^{(i+1)}x^{(i+1)}(1 - x^{(i+1)})$$

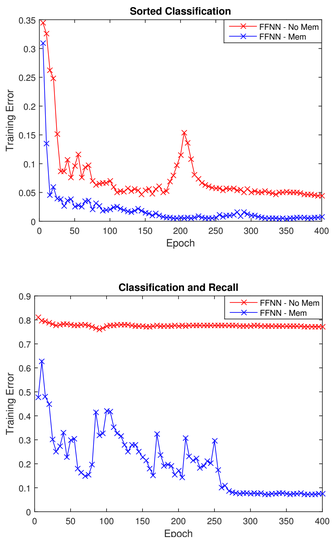
There are two paths back to the primary neurons, so the errors of each layer must be summed across memory nodes

$$\delta_j^{(i)} = \delta_j^{(i)} + \sum_k \delta_{j+k}^{(i)} M_{j+k}^{(i)}$$

Results

Sorted Classification: A data set composed of points sampled from 10 randomly generated multivariate Gaussian distributions, with the goal being to classify which distribution a point was sampled from. Samples were drawn from the distributions in a stationary order. Due to overlap in the distributions a standard feed forward neural net was only able to achieve 4.5% classification error. By implementing memory the neural network was able to identify that there was a pattern to the input points and reduces classification error to 0.7%, an improvement of 600%.

Classification and Recall: The task of this data set was to classify an input and also state the previous two classifications it made, in the order it made them. A memory-less neural network can perform no better than converging to the most prolific output on this type of task. A memory based neural network with the same parameters was able to achieve 7% classification error on the same dataset.



Memory Architecture

Every node in the hidden layers contains a static memory bank. Each unique memory in a node is orthogonal to all other memories in that node, such that no information is lost when the memories are added together.

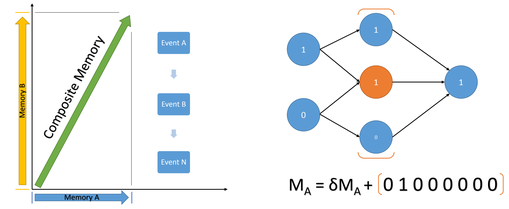


Figure 4: Composite Memories are Sums of Orthogonal Individual Memories

A collection of memories is then represented as a single vector in a higher dimensional space. The direction of the vector dictates what memories are present, and the magnitude of the vector when projected onto each memory's axis determines the order in which the memories occurred. When a node fires it stores into memory its output and the output of its nearest neighbors within some radius. This information is then projected into a higher dimensional space to ensure orthogonality.

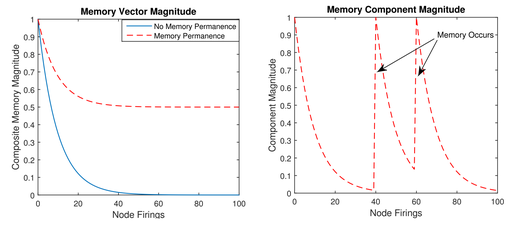


Figure 5: The Magnitude of Memories Decay Each Time A Node is Fired to Provide Transient Information

Every time a node fires, the composite memory vector of that node decays according to some pre-described decay schedule, and a new memory is added. During the training phase memories are allowed to decay to zero to provide the neural network the plasticity to converge to its desired orientation. During use memories can be retained indefinitely.

References

[1] Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

[2] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines." arXiv preprint arXiv:1410.5401 (2014).