Supervised HTM and Catastrophic Forgetting

Eric Laukien

October 31, 2014

Abstract

An investigation of how to combine hierarchical temporal memory (HTM) with supervised learning on catastrophic forgetting tasks.

1 Introduction

In this paper I attempt to investigate the applicability of hierarchical temporal memory (HTM) to supervised learning tasks. In doing so, I come up with a new version of HTM which differs from the original proposed by Numenta[1] in their cortical learning algorithm whitepaper in that it operates on continuous states as opposed to discrete states. I call this version SDRRBFNetwork (sparse distributed representation radial basis function network). Furthermore, I show that on tasks where typical machine learning techniques suffer from catastrophic forgetting, SDRRBFNetwork vastly outperforms the current state-of-the-art.

2 Catastrophic Forgetting

2.1 The Problem

Catastrophic forgetting, also known as catastrophic interference, is a problem in which a function approximator or classifier forgets previous information in favor of new information that is similar[2]. This means that the algorithm cannot effectively run in online situations, where information is presented incrementally in small steps.

2.2 Proposed Solutions - Offline

In a offline situation the entirety of the information is available beforehand. One can therefore select samples to learn from this information in such a way that samples differ significantly (temporally distinct) and do not override previously learned associations. When done randomly, this is called stochastic sampling.

2.3 Proposed Solutions - Online

In a online situation the training data is received by the function approximator in incremental pieces that are strongly correlated temporally. This means that for small changes in input, generally there will also be small changes in output.

Some suggest that catastrophic forgetting results from overlapping sets of hidden nodes being active for different inputs. Therefore, it would make sense to "sharpen"[3] the output of the most active hidden nodes and decrease strongly the output of less active nodes. This way one can enforce a more or less constant number of active hidden nodes.

Others suggest using unsupervised pre-training to deal with catastrophic forgetting[4]. With this approach, the initial weights will have been pre-destined to form a low-overlap hidden representation, thus leading to less catastrophic forgetting.

3 My Algorithm: SDRRBFNetwork

3.1 Overview

I take an approach similar to the "sharpening" approach. However, I argue that one can improve upon sharpening by using something similar to radial basis function networks (RBF). I employ an online unsupervised learning algorithm to learn time-varying efficient non-overlapping representations of the input data. I then apply a simple linear function approximation technique to translate these hidden layer representations to have the system act as a nonlinear function approximator.

3.2 Continuous HTM

The unsupervised portion of this algorithm is very similar to the spatial pooler in HTM, but instead of using discrete states, it uses continuous states for the entirety of the spatial pooler. I leave out the temporal pooler and prediction in this paper. This can be the subject of a future work.

The unsupervised portion is refered to as continuous HTM in this paper. It consists of a N dimensional grid of nodes, each with a (potentially convolutional) set of connections to the input grid.

Each node also contains a prototype vector, called P, the size of that node's receptive field. These can also be viewed as weights. Each node has a activation value A and a separate output value O. Furthermore, there is a width value W for each node that describes how sensitive it is to different input values.
The first step is to calculate the activation values of the nodes,

$$A = e^{-W*||P-x||^2}$$

where x is the input vector.

After all activations have been computed, the output of a node is computed by comparing a node's activation to all of the surrounding nodes. This is the inhibitory step; nodes that are very active inhibit neighbors. Where I is the set of all nodes in a particular node's inhibition radius, the output of a node is

$$O = e^{C_1 * \frac{A - max(I)}{max(I) - \frac{1}{|I|} * \sum I_i}}$$

where C_1 is some positive constant. The unsupervised training works as follows: When an input is received and activation/output computation has concluded, the nodes with the strongest outputs should move their prototype vectors towards the input and modify their widths W to better approximate the variance of the input.

$$P_{t+1} = P_t + \alpha * O * (x - P_t)$$

where α is a learning rate, and t denotes the timestep.

$$W_{t+1} = W_t + \beta * O * (\frac{C_2}{||x - P||^2} - W_t)$$

where C_2 is another positive constant, and β is another learning rate.

3.3 Supervised Portion

The supervised portion of the algorithm is simply a single-layer linear perceptron with one input for each output of the continuous HTM unsupervised learner. In addition, it has a single bias unit, whose output is always 1. The supervised portion is updated with standard gradient descent. Due to the well-known nature of this algorithm, I do not feel the need to go over it again here.

4 Experiments

4.1 Setup

Here, I will compare my algorithm, SDRRBFNetwork, to a multilayer perceptron with and without hidden activation sharpening and stochastic sampling, and a network with unsupervised pre-training (trained using constrastive divergence followed by backpropagation). I will compare the results on a simple task: Learning the $sin(3x)$ function. Despite the simplicity of the function, I hope to show the advantages of my algorithm over the others for online learning tasks.

4.2 Procedure

For trials in which no stochastic sampling was used (online scenario), I iterate over the $sin(3x)$ curve by incrementing x by 0.01 for every time step, all the way up tp 2π. In the stochastic sampling case, x is chosen randomly from the $[0, 2\pi)$ range.

For the multilayer perceptron, I used 16 hidden units, with a learning rate of 0.001. In the sharpening multilayer perceptron, I set the activation sharpness factor to 0.75 and the number of sharpened nodes to 3.

The unsupervised pre-training version still uses 16 hidden units (one hidden layer). It is trained with contrastive divergence for 10000 iterations with a learning rate of 0.01 before backpropagation is used for fine-tuning with a learning rate of 0.001.

The SDRRBFNetwork uses $16 * 16$ nodes, a C_1 value of 32, a C_2 value of 2, and a learning rate α of 0.01 and a learning rate β of 0.01.

The number of passes over the $sin(3x)$ function will be indicated per experiment. Stochastic sampling versions will receive the same number of samples as the incremental (online) ones.

All of the algorithms used in this paper have been tested on other problems, to assure that the implementation is indeed correct. Also, note that the learning rate has been lowered on the multilayer perceptrons, since I found that with higher learning rates such as the one used by SDRRBFNetwork the multilayer perceptrons tends to diverge.

4.3 Results

I plotted the performance of the experiments. See the end of the paper for figures labeled with the number of training iterations. Yellow is the $sin(3x)$ function. Light blue indicates the SDRRBFNetwork. Red indicates the multilayer perceptron without sharpening, green indicates the multilayer perceptron with sharpening, magenta is the unsupervised pre-trained network, and dark yellow is the multilayer perceptron with stochastic sampling (no sharpening).

5 Discussion

From these results, one can see that even at the lowest number of iterations (16), the SDRRBFNetwork gets a decent albeit noisy approximation of the curve. The multilayer perceptron without sharpening never gets a decent approximation. The multilayer perceptron with sharpening is able to approximate the first wave after 256 iterations. The pre-trained network doesn't seem to do anything of value, despite having its implementation verified on other tasks. Finally, the stochastic sampling network gets the first wave after 64 iterations and the second as well in 128 iterations.

Only the SDRRBFNetwork achieved satisfactory results. The results are not as smooth as they others due to the way the spatial pooler sharpens activations, but overall the error is far lower.

The SDRRBFNetwork uses a lot more nodes than the other techniques in these experiments. I increased the count from the original $8 * 8$ to $16 * 16$ since I found that more nodes helped accuracy (a property not necessarily shared by the other algorithms in my tests).

Even when stochastic sampling is used, the results are not as good as with the SDRRBFNetwork with so few iterations. Standard feedforward neural networks seem to require vastly more iterations in general in order to be effective, but the resulting approximations are smoother.

SDRRBFNetwork would seem to lend itself well to reinforcement learning due to its online nature. It does not require a large memory bank to draw stochastic samples from (experience replay), and it does not need pre-training.

6 Conclusion

I have demonstrated that in online learning scenarios and when little data is available, SDRRBFNetwork vastly outpeforms multilayer perceptron-based techniques. I have tested sharpening, unsupervised pre-training, and stochastic sampling, all of which cannot perform as well as the SDRRBFNetwork on few iterations. I have yet to test the generalization capabilities of the SDRRBFNetwork, but this is the subject of another work.

7 References

[1] Hawkins, J. & Ahmad, S. & Dubinsky, D. et.al (2011) Hierarchical Temporal Memory.

[2] McCloskey, M. & Cohen, N. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (ed.) The Psychology of Learning and Motivation,24, 109-164

[3] French, R. M. (1991). Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectioniost Networks. In: Proceedings of the 13th Annual Cognitive Science Society Conference (pp. 173-178) New Jersey: Lawrence Erlbaum.

[4] McRae, K., & Hetherington, P. (1993). Catastrophic Interference is Eliminated in Pre-Trained Networks. In: Proceedings of the 15th Annual Conference of the Cognitive Science Society (pp. 723-728). Hillsdale, NJ: Lawrence Erlbaum

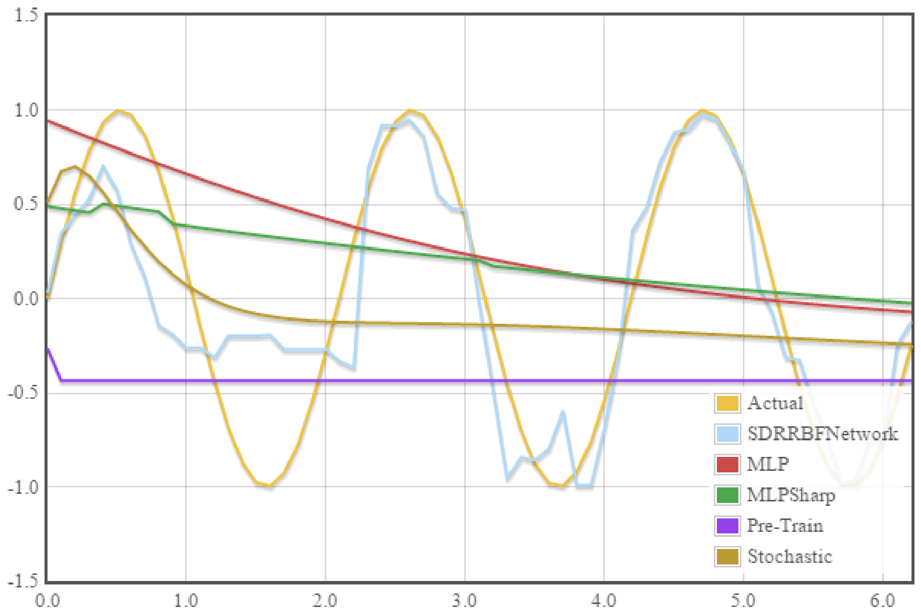

Figure 1: Comparison of techniques, 16 iterations.

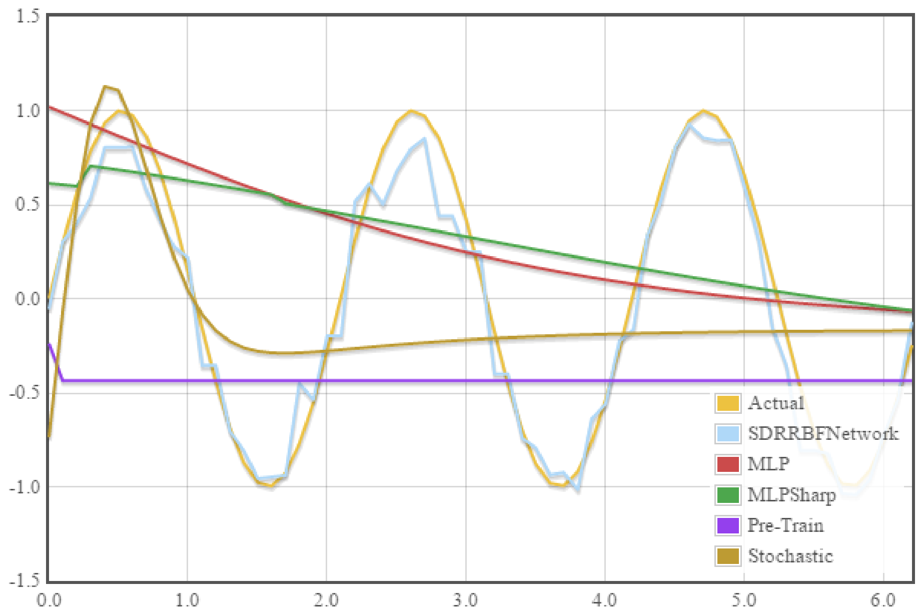

Figure 2: Comparison of techniques, 32 iterations.

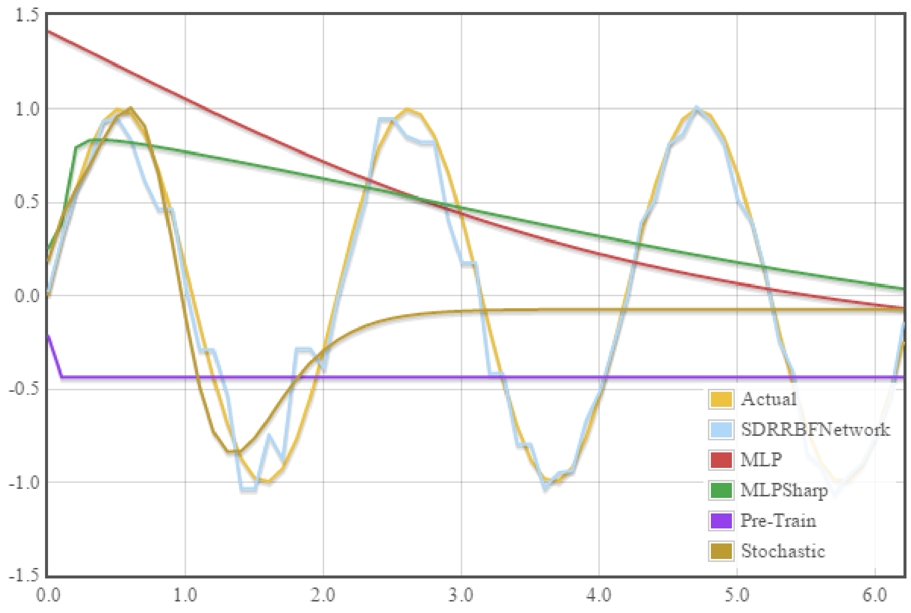

Figure 3: Comparison of techniques, 64 iterations.

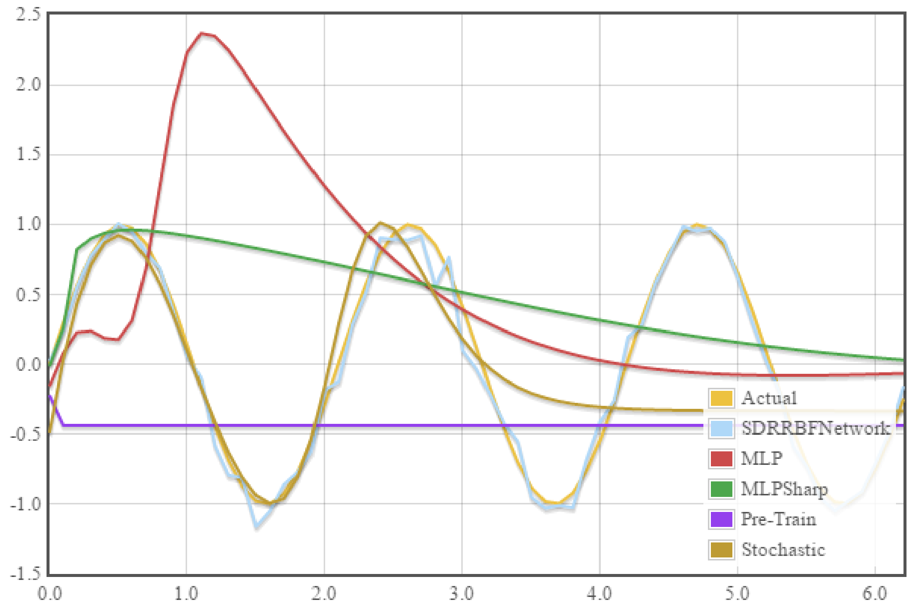

Figure 4: Comparison of techniques, 128 iterations.

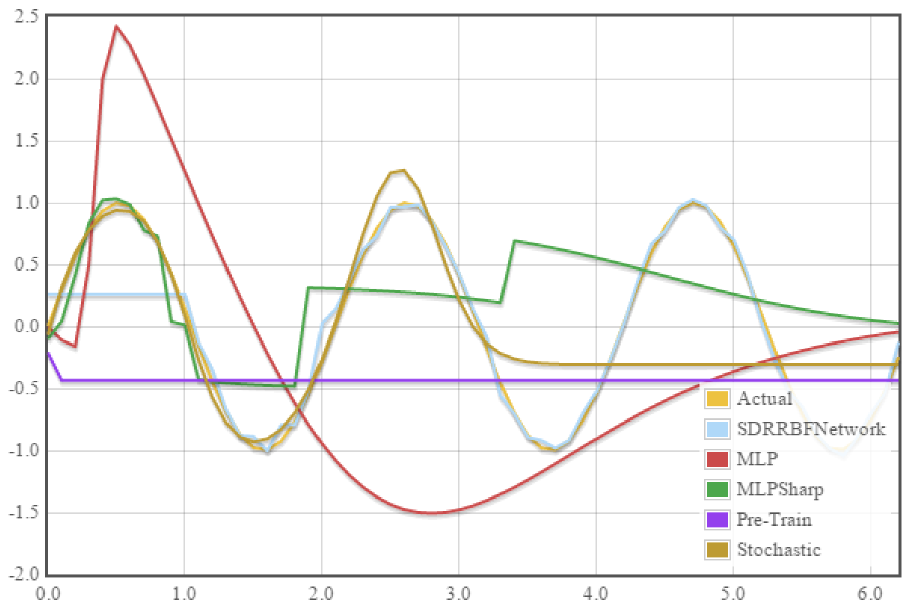

Figure 5: Comparison of techniques, 256 iterations.