Tingyu Thomas Lin

Juan M. Navarro

CS 263B – Natural Language Processing

Final Project Report

Due: Monday, June 8th, 2009

**A Perceptron-Based Approach for Image-Word Association**

Our project has 2 initial purposes:

1. ***Associate an image to a concrete noun:*** Our project will attempt to associate an image to a concrete noun. For example, when we feed in the word “tree” into our system, we want the system to generate the image of a tree. A predefined set of nouns, along with corresponding images will be used to train our system.
2. ***Understand relative positioning of objects:*** The ultimate goal is to have our system recognize relative positions of images. For example, the system should generate the appropriate image if we feed in as input, “tree right house.” The system should take in as input a phrase of the form <NOUN> <RELATIVE POSITION> <NOUN> and output the correct image.

Two modules were designed to handle each purpose:

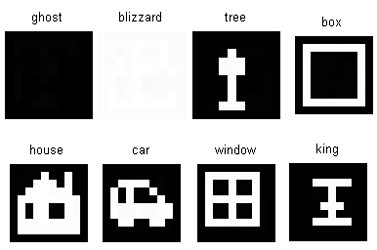
1. ***Image learning module:*** The goal of the image learning module is to learn word-image associations. We expect that training a neural network with word/image pairs until enough epochs are performed, or a minimal goal error is reached will achieve the desired association between nouns and images.
2. ***Relation learning module:***  The goal of the relation learning module is to learn the relative positioning of objects. The architecture of the relation learning module builds on the functionality provided by the image learning module, and trains a multi-layer perceptron with a subset of all the available phrase combinations. After training is completed, we expect the module to have learned the meaning of relative positions: left, right, below, and above.

Image learning module

1. The image learning module will take in as input a noun encoded in 7-bit ASCII. The output of the image learning module will be a 10x10 grayscale image representing the input noun. Our training set of nouns is in the set :

{ ghost, blizzard, tree, house, car, windows, box, king }

1. The input/output that we used to train the image module consists of 8 word/image pairs as shown in Figure 1:



***Figure 1 –*** Original training set

1. Two different architectures are used in our system:
   1. First approach consisted of a multi-layer perceptron:
      1. Architecture
         1. 3 layers: input, hidden and output

***Figure 2 –*** Multi-layer Perceptron

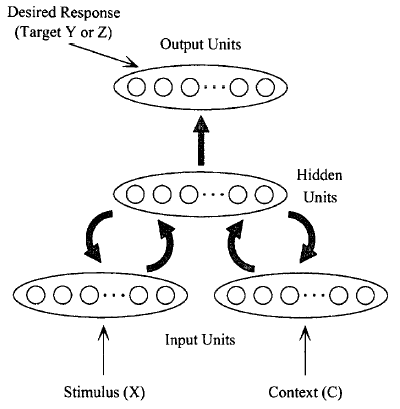
* + - 1. Logistic sigmoid function applied at the outputs of hidden and output layers
      2. Learning is based on the standard feed-forward backwards propagating neural network
    1. Three approaches were taken to implement the learning algorithm:
       1. Online training, where the weights are updated after each training image, was our first approach.
       2. Rprop
       3. iRprop+
  1. Second approach consisted of a recurrent Elman network
     1. Purpose: We selected the Elman network to help us resolve the issue of catastrophic forgetting.
     2. Architecture
* 2 layers: Uses feedback from the first layer output to the first layer input. The Elman network has tansig neurons in its hidden (recurrent) layer, and logsig neurons in its output layer.
  + 1. Issues: We attempted to use this network since we believed that [1] used it as part of the reverberating network. However, we later found that [1] instead used a modified version of the Elman net to prevent catastrophic forgetting (See Part 4.c.i).

1. Current status
   1. Online learning: converges in 1737 epochs for the 8 image set with a learning rate of 0.8
   2. Rprop learning: converges in 131 epochs for the 8 image set
   3. Two approaches were implemented to prevent catastrophic forgetting:
      1. ***Coupling two neural networks:*** Two neural nets, NET1 and NET2, are used to help prevent catastrophic forgetting. After NET1 has learned the original set of word/image associations, it is bombarded by a noise generator. The neural activity (inputs/outputs) of NET1 are captured and learned by NET2. After NET2 has learned the pseudo-patterns from NET1, NET1 is then given an external input to learn. In our implementation, this external input is the diagonal shown in Figure A.4; the noise consists of a randomly generated 7-bit ASCII string 8 characters in length. While NET1 learns the new input, it is also trained with pseudo-patterns generated by NET2 after NET2 has been bombarded with noise. Figures 3 and 4 below illustrate this process:

|  |  |
| --- | --- |
|  |  |
| ***Figure 3 –*** Noise is fed into NET1 to generate pseudo-patterns for NET2 | ***Figure 4 –*** Noise is fed into NET2 to generate pseudo-patterns for NET1 |

The results of this approach with our implementation is shown in Appendix A. Figure A.1 shows the original images learned by NET1. After NET1 has learned the associations in Figure A.1, it is then bombarded with noise, and the results shown in Figure A.2. These associations (inputs and outputs) are then transferred to NET2 for learning. Noise is then fed to NET2, and the generated pseudo-patterns, along with external input, are given to NET1 for learning (Figure A.4). Finally, NET1 is tested with the original inputs to see if catastrophic forgetting has occurred. Our results in Figure A.5 shows that with our current architecture, catastrophic forgetting is not prevented.

Our research [1] indicated that catastrophic forgetting can be prevented by using a reverberating network structure, as shown in Figure 5:



***Figure 5 –*** Architecture of a reverberating network

In such a network, the deep structure of the distributed information represented in the connection weights is captured by feedback provided to the input layer from the hidden layer.

Our (Juan’s) initial impression was that the Elman network could be used to prevent catastrophic forgetting since the reverberating network in [1] uses an architecture similar to an Elman network. Unfortunately, we had not realized that the architecture described in [1] to help prevent catastrophic forgetting was a modified version of the Elman [3] network. Hence, our tests to help prevent catastrophic forgetting with two coupled neural nets were not successful. Furthermore, the architecture described in [2] implies that noise is generated after every iteration (i.e. epoch); our (Juan’s) implementation generates a static noise vector for all epochs.

* + 1. ***Using simple pseudo-patterns:*** Research [2] presented the idea of simple pseudo-patterns: a neural net is bombarded with noise to generate associated output vectors. Each input-output pair collected is a reflection of the function previously learned in the network. The results of our implementation using this approach are shown in Appendix B. Figure B.1 shows the original set of word/image pairs learned by the net. Figure B.2 shows the learned pseudo-pattern sequence along with the new word/image association to learn (i.e. diagonal). The results, shown in Figure B.3, seem promising: the network is able to vaguely remember the associations for blizzard, tree, house, window, and box, albeit with some noise.

Relations learning module

1. The goal of the relation learning module is to understand the relative position of objects. Input will consist of a 7-bit-ASCII-encoded phrase in the form:

<NOUN> <RELATIVE POSITION> <NOUN>

1. The relations to learn are: left, right, above, below.
2. Examples of inputs are: “tree right car”, “ghost below tree”, etc.

The output of the relations learning module will consist of a 20x20 image that is described by the input phrase.

1. The input/output that we used to train the relations module consists of 196 <noun><relative position><noun> phrases:
   1. Four relations were implemented: left, right, above, below
   2. Example: house right car, tree above box
   3. 16 phrases removed from training set and into testing set
2. Architecture: similar to image learning module
3. Current status

Three major scalability problems were identified:

1. Increasing training test size results in noisy output
2. Online learning method is obscenely slow for large training sets
3. Batch learning method get stuck at local minima

Current status:

1. Learning algorithms
2. ***Catastrophic forgetting:*** Two approaches were implemented to help prevent catastrophic forgetting: simple pseudo-patterns and two coupled neural nets. Our results from the simple pseudo-patterns (see Appendix B) seem more promising than the results of the two coupled neural nets (see Appendix A). However, the simple pseudo-pattern approach failed to prevent catastrophic forgetting on a consistent basis: the results in Appendix A were the best we were able to obtain after doing several runs. We attempted the two-coupled-neural-net approach with multi-layer perceptrons: research [1-3] shows that it is best if this approach is taken with a reverberating network as the underlying architecture.

Description of software packages used:

* We decided to use Matlab for our implementation since it provides great handling of matrix computations and offers a rich variety of functions.

Appendix

1. Results of catastrophic forgetting by coupling two multi-layer perceptrons:

|  |  |
| --- | --- |
| report-pp-1.jpg  ***Figure A.1 –*** Original set of images learned by NET1 | report-pp-2.jpg  ***Figure A.2 –*** Pseudo-patterns generated by NET1 after it has been bombarded with noise |
| report-pp-3.jpg  ***Figure A.3 –*** Pseudo-patterns learned by NET2 and originally generated by NET1 | report-pp-4.jpg  ***Figure A.4 –*** Pseudo-patterns originally generated by NET2, learned by NET1 along with external input |
| report-pp-5.jpg  ***Figure A.5 –*** Output of NET1 after learning pseudo-patterns from NET2 and external input |  |

1. Results of catastrophic forgetting by using simple pseudo-patterns:

|  |  |
| --- | --- |
| pseudo-original-2.jpg  ***Figure B.1 –*** Original set of learned images | pseudo-vectorsinput-2.jpg  ***Figure B.2 –*** Learned pseudo-patterns along with learned image (i.e. diagonal) |
| pseudo-remember-2.jpg  ***Figure B.3 –*** Attempt to remember after new image (i.e. diagonal) has been learned |  |

1. Other Appendix Items

References

1. Ans, B., Rousset, S. Avoiding Catastrophic Forgetting by Coupling Two Reverberating Neural Networks. Laboratoire de Psychologie Experimentale (CNRS EP 617), Universite Pierre-Mendes-France, BP 47, 38040 Grenoble cedex 9, France. November 3, 1997. DOI= <http://adsabs.harvard.edu/abs/1997CRASG.320..989>
2. Ans, B., Rousset, S., French, R. M., and Musca, S. Self-Refreshing Memory in Artificial Neural Networks: Learning Temporal Sequences without Catastrophic Forgetting. Connectionist Science, Vol. 16, No. 2, June 2004, pp. 71-79. DOI= <http://www.citeulike.org/user/chchatham/article/461062>
3. Elman, J. L. Finding Structure in Time. University of California, San Diego. Cognitive Science, 14, pp. 179-211, 1990. DOI= <http://homepages.inf.ed.ac.uk/keller/teaching/connectionism/CogSci90-Elman.pdf>