CS3105 AI

Neural Network practical

1. The aim over the break should be to explore Encog and simple problems with a view to judging what you will need to tackle the practical. My advice is to make limited attempts at parts 1 and 2 and then once these are successful to make fuller attempts. If you find that these notes and the relevant parts of Russell and Norvig are insufficient, please let me know what further information you require.

2. I/O mapping, sigmoid, activation, and hidden units:
   Each neural input unit receives an input value from the environment at time t. The set of all neural input values at time t is an input pattern that corresponds to an input coordinate in neural Input Space where an axis corresponds to a neural input unit.
   Each neural output unit sends an output value into the environment from the network at time t. The set of all neural output values at time t is an output pattern that corresponds to the input pattern at time t. Each output value at time t is associated with an input coordinate at time t in neural Input Space. The set of these I/O associations form an *I/O mapping*. Note that the output value sent, i.e. the actual output value, will typically differ from the target output value and that training aims to close this gap acceptably.
   The total excitation of a neuron at any time is made up from the sum of individual excitations along each incoming connection to the neuron. An individual excitation is the product of the connection's weight value and the output from the source neuron at the other end of the connection. The weights can range from –infinity to +infinity in value as can the individual and total excitations. The total excitation is squashed by the *sigmoid* activation function to generate an output *activation* between 0 and 1.
   The *hidden units* have no direct connection to the environment and provide a non-linear transformation from network input to output via their sigmoids.

3. Feedforward nets:
   This type of net can provide any I/O mapping with sufficient hidden units to partition the I/O. That is, each hidden unit splits the Input Space linearly into two halves, where one half has Input coordinates that lead to the hidden unit producing output activations greater than 0.5 and the other half produces ones with less than 0.5. The I/O mapping for the network as a whole depends on how the I/O associations are set by the hidden unit partitioning. We will see that the orientation of each hidden unit split is set by its incoming weights. Training changes the weight values until the splits are all in the right place.

4. Recurrent nets:
   Recurrent, i.e. cyclic, connections can be added to a purely feedforward network to create a more powerful network. In the practical, an Elman network has a recurrent connection from each hidden unit back to itself.

Such recurrent connections enable the previous activation to flow along them and influence the activation for the next Input pattern. This effectively creates an internal state. Consequently, recurrent networks can be trained to be Finite State Machines.

5. Feedforward network practicalities:
As described above, training consists of weight adjustment. Encog can be used to set up a network with random or certain initial weights and then a training regime such as back-propagation or resilient propagation can be used to change the weights to improve the splits and consequently the error due to the difference between the actual and the target outputs for the various Input training patterns. Encog provides information and code for doing this for various examples, e.g. http://www.heatonresearch.com/wiki/Hello_World. I suggest you try the simple examples such as AND, and XOR, until you feel you have a grasp of what numbers are important and what they mean.

6. Error:
For instance, there are various forms of error. The sum of the squared differences between the actual and the target outputs over all the Input training patterns divided by the number of training patterns is Least Mean Square (LMS) Error. We will see that this directly drives weight adjustment. Total (LMS) error should typically go steadily down during training while errors for individual training patterns may go up as well as down.
There is also classification error. If the network's analogue output between 0 and 1 is given a binary interpretation, with 0.5 and above treated as 1 and below 0.5 interpreted as 0, then each actual output on the wrong side of 0.5 relative to its target has a classification error of 1.

7. Targets:
Analogue targets may be set to be 0.0 and 1.0 to draw the actual outputs towards their extreme values, or to values such as 0.2 and 0.8 so the outputs remain within the approximately linear part of the sigmoid and promote lower and more flexible weights. Training can stop when the actual outputs gets close enough to their targets, e.g. the right side of 0.5 or within 0.1 or 0.2 of their target value – i.e. a goal tolerance is set. Training may also require a timeout so it terminates feasibly and when the problem cannot be solved within goal tolerance.

8. Weights:
Initial weights are usually set to be random values within -1 and +1. This enables training to start with low excitation and so within the flexible, approximately linear, part of the sigmoid.

9. Training methods:
Back propagation requires a pair of training parameters called the learning rate and momentum coefficient to be set. They are usually set between 0 and 1 with the learning rate nearer 0 and the momentum coefficient nearer

1. Resilient propagation does not require training parameters to be set but offers less control over the learning.

10. What can definitely be done now:
More theory will be provided in the lectures after the break, but you can get a lot from doing your own exploration during the break. Try training a solution for AND, and then XOR in Encog. How many hidden units do you need for each and why? What learning parameters work best for back propagation for these problems? Does the output and error behave as you expect during training? What is required to convert 20Q data into neural Input and targets? What would be a minimal Finite State Machine problem for an Elman net?

11. Training will fail when there are insufficient hidden units or the learning parameters are set too large or small. Think about how you could add further hidden units or reset the learning parameters automatically when this happens.