DTAD-20: Feed-Forward Neural Net with Back-Propagation Learning

Purpose: Gain hands-on experience with neural networks and the classic back-propagation algorithm.

1 Assignment

Implement a feed-forward neural network classifier with gradient-descent learning (via the back-propagation algorithm) as explained in class. The details of back-propagation are in the document backprop.pdf on the course web site.

Your system will act as a classifier in that attribute values will be encoded into values in the input layer, and the (encoded) correct class should come out on the output layer.

The inputs to your system should include:

1. The topology of the network in terms of how many layers to include and how many nodes are in each layer. It can be assumed that every node in layer k sends an arc to every node in layer k+1, but to no other nodes in any other layers.
2. The transfer function - each neuron should use the same function.
3. The data set.
4. The percentage of the data set that should be used for training, with the rest being used for testing.
5. The attribute in the data set that will constitute the class, and hence the output, of the neural network.
6. The learning rate.
7. The number of training epochs.

It is recommended that, in addition to its normal inputs from layer k-1, any layer-k neuron, X, should have one auxiliary input from a dummy node whose output is always 1. By learning the weight, W, on the arc between this dummy node and X, the network essentially learns the threshold for the transfer function. For example, if W = -5, then a sigmoidal transfer function would need a total of +5 weighted inputs from the k-1 layer in order to fire (since the sigmoid begins to go high when its summed input exceeds 0). These dummy nodes and weights are evident in most of the examples in the lecture-note file suplearn.ppt.

Run your ANN on at least two different data sets from the Irvine repository. Each data set should contain at least 100 entries. On each data set, run experiments where you test out the effects of different:

1. network topologies - in particular, different numbers of nodes in the hidden layer. Also, check the difference between using one versus two hidden layers. Test AT LEAST 3 different topologies for each data set.
2. learning rates - select at least 3 different values between 0.1 and 1.
3. transfer functions - try, at least, a sigmoidal and a linear output function.

You do not need to test all permutations of these. You might begin with a simple topology and find the learning rate and transfer function that work best for it. Then use that same rate and function when you test other topologies.

For each test, produce a graph that shows the training error on the y axis and the epoch number on the x axis. This will illustrate the progression of gradient-descent learning.

2 Deliverables

1. The working code, to be demoed as usual.
2. A short report documenting the many test runs that you did, including the graphs of training error as a function of the epoch. Also, for each of the data sets, include a description of how case information is encoded into values in your input layer and how class information is decoded from the values on your output layer. Pictures are usually quite helpful in illustrating encodings and decodings!

3 Additional Deliverable

Also, for week 9, remember to find ONE interesting AI application on the internet and bring the URL to class. Be prepared to give a brief 5-minute description of the application. We will use one lecture-hour of my next visit for this purpose. Applications that involve one or more of the tools that you have learned in your 2 AI courses are preferred, and your presentation should focus on those tools and how they are employed in the project.