Homework # 2

15-486/782: Artificial Neural Networks

Dave Touretzky

- Due September 13, 2006. Answers must be typed.
- Software you need is in `/afs/cs/academic/class/15782-f06/matlab/bp`

Problems

1. Consider a multilayer perceptron whose hidden units use x^3 and whose output units use $\sin(3x)$ as the transfer function, rather than the usual sigmoid or tanh. Using the chain rule, starting from $\partial E/\partial y_k$, derive the formulas for the weight updates Δw_{jk} and Δw_{ij}. Your final formulas should be purely algebraic, i.e., they should not contain partial derivatives.

2. The XOR function is the 2-bit parity problem. Consider the 4-bit parity problem: the input is a four element binary vector, and the output is a $+1$ or a -1 depending on whether the input contains an even or an odd number of ones. There are 16 possible input patterns. Parity is a hard problem because changing any one bit of the input causes the desired output to flip.

   (a) Write code to train a backprop network to solve this problem. Your code should use the `bp_innerloop` routine. Look at demos like `bpxor` and `encoder` to see how to set up the input and output patterns, and the training loop. To make learning a little easier, use targets of $+0.8$ and -0.8 instead of $+1$ and -1. Use a learning rate of 0.04, and no momentum.

   (b) Plot the set of 16 output values, `Result2`, as a line with 16 points numbered 0–15. Update this plot every 10 epochs. Also plot the desired outputs as a set of points (not a line); use a different symbol and a different color. This will help you monitor the network's progress as it tries to get all the patterns to come out right. Note: in order to superimpose two plots, you will need to use the command `hold on`.

   (c) Also plot the output of one of four hidden units as a set of points superimposed on your other plots. Try to analyze what that hidden unit is doing, e.g., "this unit is for looking input patterns where bit 2 is on and bit 4 is off; it ignores the other two bits."

   (d) Be sure to answer every part of this question. (i) How many epochs does it take to learn this problem using 5 hidden units? Run your network ten times and record the number of epochs each time, and the mean and standard deviation for the ten runs. (Use Matlab's `mean` and `std` functions.) (ii) Now run the same experiment using only 4 hidden units. (iii) Now try training with 20 hidden units. (iv) How about 100 hidden units? (v) Two hidden units are not enough to solve four-bit parity. But how well can the network do, i.e., how many of the 16 input cases can it get right, where "right" just means the output has the correct sign?

Turn in a source listing, a sample plot, a table listing your observations about training time as a function of number of hidden units, and your analysis of one hidden unit.