CS 420/594 Biologically Inspired Computation Project 4 Due Thursday, Nov. 5, 2009

Bruce J. MacLennan

Implement a Back-Propagation Software System.[1] Your system should permit specification of:

1. number of layers
2. number of neurons in each layer
3. learning rate
4. training, testing, and validation files (see below)
5. number of training epochs

Inputs and outputs to the net are floating-point numbers.

Your program takes three files as input (it is permissible to combine them into one file divided into three parts):

Training Set — used for training the net by modifying the weights according to the back-propagation algorithm.

Testing Set — used at the end of each epoch to test generalization (the weights are not modified).

Validation Set — used at the end of training to evaluate the performance of the trained net.

I suggest that you test your BP program on some simple problems, such as *and*, *inclusive or*, and *exclusive or*, to make sure it's operating correctly, before trying the problems on the next page.

[1]Additional information, including data files, can be found on Kristy's website, <`http://www.cs.utk.edu/~kvanhorn/cs594_bio/project4/backprop.html`>.

You will generate two sets of Training/Testing/Validating data from the following functions:[2]

Problem 1:[3]

$$f(x,y) = \frac{1 + \sin(\pi x/2)\cos(\pi y/2)}{2}, \quad x \in (-2,2), y \in (-2,2).$$

Problem 2:

$$f(x,y,z) = \frac{3}{13}\left[\frac{x^2}{2} + \frac{y^2}{3} + \frac{z^2}{4}\right], \quad x \in (-2,2), y \in (-2,2), z \in (-2,2).$$

In each case generate (input, output) pairs from random inputs in the ranges specified, and outputs determined by the above formulas. For each problem, generate:

- 200 training patterns
- 100 testing patterns
- 50 validation problems

For each Problem, do the following experiments:

- Experiment with the number of hidden layers.
- Experiment with the number of neurons in each hidden layer.
- Try to determine the optimum network architecture for each problem.
- Try to determine how sensitive the performance is to the architecture.

In evaluating the architectures, pay attention to performance on the training, testing, and validation datasets.[4] Note that, depending on the architecture, these problems might take several thousands of epochs to converge.

Be sure to include some graphs in your report. For example, you could plot how the error changes as you increase the number of nodes, or plot how the error changes as you increase the learning rate, or plot the error as a function of time (number of epochs), or anything else you can think of. Of course including more graphs/experiments/discussion will lead to higher grades.

[2]You can use the datasets from Kristy's website or generate your own.

[3]Note that in this formula the arguments to sin and cos are expressed in radian measure; you will have to make appropriate changes if your sin and cos expect arguments in degrees.

[4]For small architectures, the net should achieve average errors less than 4% (which corresponds to 0.001 average square error) within a few hundred, or at most a few thousand, epochs. The error may not seem to change for many epochs and then drop in steps.

For graduate credit: Problem 3

In addition to the preceding, explore your network's ability to classify points generated from the following two overlapping two-dimensional Gaussians:

$$\begin{aligned} A(x,y) &= \frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right), \\ B(x,y) &= \frac{1}{8\pi}\exp\left(-\frac{(x-2)^2+y^2}{8}\right). \end{aligned}$$

Let $x \in (-4, 10)$ and $y \in (-6, 6)$. Generate input-output pairs $((x,y),1)$ with probability $A(x,y)$ and pairs $((x,y),0)$ with probability $B(x,y)$, with equal probability for A and B. Explore your network's ability to discriminate points generated by distribution A (1 output) from those generated by distribution B (0 output). Experiment with network architecture, generalization, etc.

The simplest way to generate the N samples is as follows:

1. Generate $N/2$ samples from A as follows:
   (a) Generate a pair of coordinates (x,y) with probability $A(x,y)$.
   (b) Add $((x,y),1)$ to the dataset.
   One way to do (a) and (b) is (1) generate random (x,y), (2) generate a random number r uniformly distributed in $(0,1)$, and (3) if $r < A(x,y)$, then put $((x,y),1)$ into the dataset and count it.
2. Generate $N/2$ samples from B as follows:
   (a) Generate a pair of coordinates (x,y) with probability $B(x,y)$.
   (b) Add $((x,y),0)$ to the dataset.

Then you will have an equal number of points from the A and B distributions. (Note, you will probably want to shuffle their order if you do online learning.)

Here is a scenario that may help you to understand this problem. Suppose x and y represent two physiological measurements, such as heart rate and body temperature. (Maybe they are measured relative to some baseline so that negative values make sense.) Suppose that people with a certain serious medical condition are determined to have (x,y) measurements distributed as $A(x,y)$. On the other hand, people with a different, non-serious medical condition have measurements distributed as $B(x,y)$. The problem is to train the network so that given a particular set of (x,y) measurements the network can make the best guess as to whether the patient has the serious condition or the non-serious condition. (Because the distributions overlap, it cannot be correct all the time. However, since the network produces outputs in [0, 1], it in effect estimates the probability of the two conditions given the measurements. For example, if the net output is 0.5 it is saying that the two conditions are equally likely.)