CPSC 636-600 Homework 2 (Total 100 points)

See the course web page for the **due date** and **submission info**.

Instructor: Yoonsuck Choe

February 9, 2008

Problem 1 (Written: 10 pts): Solve exercise 3.1

Problem 2 (Written: 20 pts): Solve exercise 3.2

Problem 3 (Program: 25 pts): Using the skeleton code in `src/bp.m`, implement a working backpropagation learning algorithm. The only part that you need to complete is the input-to-hidden weight learning part. See the documentation in the skeleton code for input arguments and returned values.

1. We will use the $\tanh(\cdot)$ activation function, so you should normalize your inputs and output between $[-1, 1]$.
2. In the program, each *row* in the input matrix and the output matrix represent one sample and its target.
3. In the program, the input, output, and hidden layer activations are represented as *row vectors*.
4. In the program, the weight vectors for each layer's units are stored as *column vectors*.
5. As a result, given the input vector $\mathbf{x}$ as a row vector, multiplying it with the weight matrix W_{hid} will give you the hidden layer activity vector as a row vector $\mathbf{h}$.

   $$\tanh(\mathbf{x}\mathbf{W}_{\text{hid}}) = \mathbf{h}.$$

   The same goes for hidden to output activation:

   $$\tanh(\mathbf{h}\mathbf{W}_{\text{out}}) = \mathbf{o}.$$

Problem 4 (Written: 5 pts): Test the backprop algorithm on the XOR problem:

```
[w_hid, w_out, output, hidden, err_curve] =  \
    bp([-1 -1 ; -1 1; 1 -1; 1 1], [-1; 1; 1; -1], 2, 0.1, 2000);
```

and check the output and the error curve, and report the results. Note "\" at the end of the line means the line should be continued on the next line.

```
output
...
...
plot(err_curve)
```

Problem 5 (Written: 10 pts): Solve exercise 4.16 item 4 only, with $-\pi \leq x \leq \pi$, and address all tasks (a), (b), and (c). Note: first calculate $\sin(x)$ for the given range to collect the target output. Then, scale the input between $[-1, 1]$. In other words, your input-output mapping should be $\{(x/\pi, \sin(x))\}_{x=-\pi,..\pi}$.

```
x = (-pi:0.1:pi)';
Input = x/pi;
Target = sin(x);
[w_hid, w_out, output, hidden, err_curve] =  \
    bp(Input, Target, 4, 0.01, 4000);
plot(x,sin(x),x,output);
plot(err_curve);
```

Problem 6 (Written: 15 pts): Using the same backpropagation program, test the binary code example discussed in the class, with a varying number of hidden units. Note again, you need to scale the input between $[-1, 1]$. `eye(n,n)` gives you an $n \times n$ identity matrix. Multiply this by 2 and subtract 1 to scale it between $[-1, 1]$.

```
Input = eye(8,8)*2-1;
Target = Input;
[w_hid, w_out, output, hidden, err_curve] =  \
    bp(Input, Target, 3, 0.05, 10000);
imagesc([Target,output]);
```

Try hidden unit count of 1, 2, and 3, and report your findings: (1) How accurately are the targets learned? (2) How does the hidden layer representation look like? (`hidden` returns the hidden layer activation for all input patterns from the last epoch).

```
output
hidden
imagesc(hidden);
plot(err_curve);
plot(mean(err_curve'));
```

Problem 7 (Written: 15 pts): Train the backprop program with the following input set. Note that each row corresponds to one input vector. The first set is the positive examples (output should be +1), and the second set negative (output = -1).

```
pos =

  -1  -1
   0  -1
   1  -1
  -1   0
   1   0
  -1   1
   0   1
   1   1
```

```
neg =

  -0.50000   0.00000
   0.50000   0.00000
   0.00000   0.50000
```

(1) Plot the data set, either manually or with Matlab/octave. For example,

```
plot(pos(:,1),pos(:,2)),'b+',neg(:,1),neg(:,2),'rx');
```

will plot the inputs in two distinct color/shape. (2) Before you begin, mark how many hidden units you think is necessary to correctly learn the input set perfectly: choose from 2, 3, and 4. (3) Try training with all three hidden layer size, and report the results. Did they come out as you expected? Explain why.

```
Input = [pos; neg];
Target = [ones(8,1);-ones(3,1)];
[w_hid, w_out, output, hidden, err_curve] =  \
    bp(Input, Target, 3, 0.05, 10000);
```

(4) You can actually visualize the decision boundary using the following approach. Do this for all three hidden layer size and report your findings, especially comparing hidden layer size 2 and 3.

1. Generate an input set to tile between $[-1, 1]$ at a step size of 0.1.
2. Calculate the output of the trained network with the new input set. You can achieve this by providing `w_hid` and `w_out` as the last two arguments when calling the `bp` function. The returned `output` is the result. Make sure you do not overwrite your previous calculations by renaming the returned values.

   ```
   dat  = ....
   [w_hid2,w_out2, output2, hidden2, err_curve2] = \
   bp(dat, Target, 3, 0.1, 1, w_hid,w_out);
   ```

3. Divide the output into two groups, those with output > 0 and those with output ≤ 0, and plot them. Suppose `dat` contains your new input set. You can also plot it in the context of the training set.

   ```
   opos = find(output>0);
   oneg = find(output<=0);
   plot(dat(opos,1),dat(opos,2),'rx',dat(oneg,1),dat(oneg,2),'b+');
   plot(dat(opos,1),dat(opos,2),'rx',dat(oneg,1),dat(oneg,2),'b+',\
   neg(:,1),neg(:,2),'g*',pos(:,1),pos(:,2),'b*');
   ```