Problem Set 4
Due Sunday, May 14, 2017 at 11:55pm

How to Submit

Create one .zip file (**not** .rar or something else) of your code and written answers and submit it via `ilearn.ucr.edu`. Your zip file should contain the following 3 files

- runq1.m
- trainneuralnet.m
- q1out.pdf

plus any other matlab functions your code depends on that you wrote.

Each file should include at the top (in comments if necessary)

- Your name & UCR student ID number
- The date
- The course (CS 171) & assignment number (PS 4)

Note: Do not use any Matlab function that is in a toolbox for this problem set.

Problem 1. [25 pts]

In this problem you are implement a two-layer (2 layers of weights, 1 hidden layer of units) neural network for classification with all non-linearities as sigmoids. The file `toy.data` contains the X (first 2 columns) and Y (last column) values for a 2D problem. In this case the Y values are either 0 or 1 (instead of -1 or +1), because that matches a sigmoid output more naturally.

There are no testing data for this. Instead, you are to produce an output plot of the decision surface. To help, two functions are provided:

- `getgridpts`: This takes the training X matrix (just to know the range of the data) and returns a set of X points, laid out on a grid.
- `plotdecision`: This takes the training X, the training Y, the grid X points (from getgridpts) and the grid Y points (the values your classifier predicts for the grid X points), and plots the decision surface.

The pair of functions can be used as follows.

```
% load in and set up trainX as the training X pts and trainY as the Y pts
gridX = getgridpts(trainX);
% train classifier
% use classifer to predict labels for gridX, call this the vector gridY
plotdecision(trainX,trainY,gridX,gridY)
```

The supplied function `runq1` learns a neural network with different numbers of hidden units and different λ (regularization). It calls a function you should write: `trainneuralnet(X,Y,nhid,lambda)`. Getting a neural network to converge can be a little tricky. Please follow the guidelines below.

1. Start the weights randomly chosen using `randn` (that is each weight is selected from a normal distribution with mean 0 and unit standard deviation.
2. For a problem this small, use batch updates. That is, the step is based on the sum of the gradients for each element in the training set.
3. For the step size use $\eta = 0.1$. For real neural network training, people use rules like those discussed on `http://sebastianruder.com/optimizing-gradient-descent/index.html`, but they are more complex than needed for this simple example.

4. Check the maximum gradient element (before multiplying it by η). If it is less than $1e-4$, stop the algorithm.

5. The method will probably take 100,000 – 500,000 iterations to converge.

At the end, your function needs to plot the decision surface (see above) and return the two matrices of weights (the first for the input-to-hidden layer and the second for the hidden-to-output layer).

A few notes to help

1. Every 1000 iterations (or so), plot the surface so that you can see what is going on (see `drawnow`).

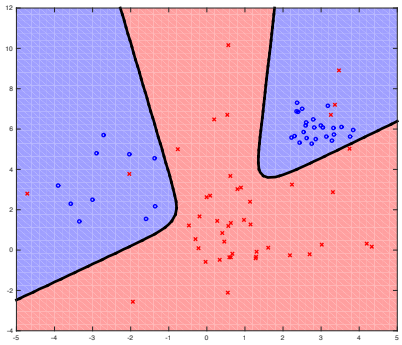
2. Display the loss function's value every 1000 iterations (or so). It should generally be getting smaller.

3. The smaller λ is and the more units, the more difficult the optimization will be for gradient descent.

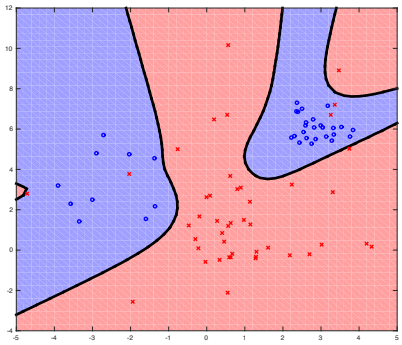
4. Write a function to do forward propagation and one to do backward propagation. Write a function to evaluate a given neural network. In general, break things up to keep things straight.

5. Processing the entire batch at once is much more efficient in Matlab. Work first with loops, but then look to propagate the entire batch forward (and backward) without loops.

6. My answers for two of the nine plots look like those below. Note that because of the random starting weights, your solutions might look a little different, but probably will look very much like these.



nhid = 5, $\lambda = 0.00025$



nhid = 15, $\lambda = 0.00025$

The `runq1` function will save a PDF (`q1out.pdf`) of the results. Include this file in your submission.