Problem Set 3
Due on Friday, October 31, 2014 at 11:59 pm

How to Submit

Create one archieve file (.tgz or .zip, but **not** .rar) of your code and written answer and submit it via `ilearn.ucr.edu`. Supply one script for each method. Name them `runlogres.m` and `runpercep.m`. Supply your answer to the last question in a text file called `ans.txt`. You may have additional functions (and indeed are encouraged to do so!), but there should be one script per question that executes them and shows the results. Do not supply any directories in your zip file. Each file should (in comments) list

- Your name
- Your UCR student ID number
- The date
- The course (CS 229)
- The assignment number (PS 3)

Linear Classification

The supplied file `class2d.ascii` contains a toy problem. The first two dimensions are the x_1 and x_2 values for each of the 80 examples. The third dimension is -1 for "negative" examples and $+1$ for "positive" examples: the labels y of the examples. Real data is seldom two-dimensional, but we often would like to visualize how our algorithms work. In this problem, you will use Matlab to visualize two different linear classification algorithms on this dataset.

Your code should execute two linear classification algorithms: logistic regression and the perceptron learning algorithm. For logistic regression, you should use the iteratively reweighted least squares method. For the perceptron learning algorithm, you should start with a learning rate of 0.1 and decrease it by multiplying it by 0.8 after each pass through the whole data.[1] Do not forget (for either method) to have a "bias" term so the line does not have to pass through the origin.

While running the algorithms, your code should display the current linear decision surface in the following fashion. Plot the positive training points as blue Xs. Plot the negative training points as red Os. Draw the current decision boundary as a dark black line (use the supplied `drawline` function). For logistic regression, you should plot this after each update. For the perceptron learning algorithm, you should plot this after processing each data point. Pause slightly (very slightly for the perceptron) after each update so that you can see the algorithms progress.

How many iterations does it take for each method to converge? Which method do you prefer for this problem? What if there were thousands (or millions) of data points?

[1] In general, this is not a good schedule for the learning rate. Multiplication by a constant less than 1 does not have great convergence properties for general gradient descent. However, for this homework it works fine and is simple.