Problem Set 2
Due on Friday, October 24, 2014 at 11:55 pm

How to Submit

Create one archieve file (.tgz or .zip, but **not** .rar) of your code and submit it via `ilearn.ucr.edu`. Supply one script for each plot. Name them `plot1.m` and `plot2.m`. You may also include other `.m` files containing functions.

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 2)

Linear Regression

The supplied file `machine.ascii` contains the data for the "Computer Hardware" problem from the UCI Machine Learning repository.[1] It can be loaded in Matlab with the command `load -ascii machine.ascii`. This problem has 209 examples, each with 7 attributes. This is small by modern machine learning standards, but will provide a simple example for this problem set. The attributes are (in order) the machine cycle time in nanoseconds, the minimum main memory size in kilobytes, the maximum main memory size in kilobytes, the cache memory size in kilobytes, the minimum number of channels, the maximum number of channels, and the relative performance. Each example represents a different computer and the goal is to build a function that predicts the performance of a computer (the seventh attribute) from the other attributes.

Your task is to use ridge regression to do this. You should build a linear function of the inputs (and include a "constant" or "bias" term). Use the form of ridge regression presented in class (and the textbook). Matlab does provide a ridge regression function (called `ridge`). **Do not use the one Matlab supplies**; write your own.

There are two important pragmatic points when applying ridge regression you should observe:

- First, do not regularize the bias coefficient. Only penalize the other coefficients. You will have to work out how to do this.
- Second, you will need to scale each feature's value so that the data values have roughly the same range. This is true for other regression and classification methods are well.
  The simplest (and very common) way to do this is to divide each feature value by its emperical standard deviation in the dataset. This is often called z-score normalization (or sometimes just z-normalization). That is, for dimension i, find the standard deviation (`std` in Matlab) of dimension i across all examples and then divide every example's value for dimension i by this standard deviation. This helps to compensate for different features being measured in different units. Otherwise, the regularization will have different strengths for different features. Usually (and you should do this too), the mean of each dimension is subtracted first (to "zero" the data). The resulting function is the same either way, but the weights differ.
  Remember, you have to apply the same transformation to the testing data before using it!

You are to explore the effect of changing the regularization strength, λ. To do this, generate two plots. The first should plot the performance of ridge regression as a function of the value of λ. The second should plot the resulting coefficients (or weights or parameters) as a function of λ.

[1] The repository can be found at `http://www.ics.uci.edu/~mlearn/MLRepository.html`. The raw data can be found at `http://archive.ics.uci.edu/ml/datasets/Computer+Hardware`

For the first plot, you will need to split the data into two portions: a training and a testing portion. As we are most interested in the situation with little training data, select 20% of the data for training, and 80% of the data for testing. (This dataset has only a few attributes, so to see the effect, we need to limit the amount of training data drastically.) The data should be split randomly; look at the Matlab function `randperm`. For a random split, vary λ and calculate the testing accuracy in terms of average squared error over the testing data. Repeat this procedure for 20 different training-testing splits and plot the average performance (across splits) for each λ value.

For the second plot, use the whole dataset (it will be easier). Plot the value of each learned coefficient (and the bias coefficient) as a function of λ, each as a separate curve on the same plot. Be sure to label them.

I recommend plotting both plots with the x-axis in log-space (a semi-log plot); use the function `semilogx` instead of `plot` to do this. Plot a range for λ from approximately 10^{-3} to 10^{3}.