**CS 294 Homework Assignment 1** (due September 24)

**Prerequisites for R.** Parts of this homework assignment will be done using a statistical programming language called R. The software is freely available at http://www.r-project.org/ and is available in Windows, OS X and Linux versions. Knowledge of R will be very useful for future work with statistical models, and furthermore, R has a very good SVM package which can be used on large-scale real-life problems. This section will have you install R; no deliverables are required.

**Installation.** You will need to install R and a package called e1071. Once R is installed, run it.

So, in order to install e1071, type in

install-packages("e1071")

*Note.* On Unix machines, the installer will attempt to install the libraries to a global location. If you do not have permissions to do so (or do not want an unknown program to execute as root), you can set the environment variable R_LIBS to point to a writable directory and all package installation will occur there.

**Basic Usage.** R is a very powerful language which has many useful features. It is suggested that you take a look at a tutorial to get familiar with basic functionality.

In order to use a library, execute

library("libname")

(1) Toy Problem

See file `toy.R` for code and assignment. Perform the experiments required in the assignment and report on your findings. Submitting code is not required.

(2) YALE and Spam Filtering

YALE (Yet Another Learning Environment) is a Java library and GUI implementing a large number of ML functions. Download and install it from http://rapid-i.com. It is suggested that you go through the tutorial to familiarize yourself with its features. If you wish to experiment with the data for problem 1), load the experiment `toy-yale.xml`. Remember to turn on the expert mode in YALE.

This time, our dataset will be the real-life spam/ham (not spam) dataset used to train SpamAssassin. Originally, the training set is in the form of email messages. In order to perform classification, it is necessary to convert the email messages into feature vectors; in this case, we use a representation called the "bag of words." That means we model a document as a collection of word frequency counts – each dimension in the feature vector is the number of times that the corresponding word appeared in the document. As part of preprocessing, the headers were removed, the HTML tags were stripped, and the 500 most relevant (you'll learn about this later) words were left.

(a) Basic Model

Load `spam-basic.xml` to have a basic experiment loaded. This loads the data, then cross-validates a classifier. Run the model and observe the performance. The model currently used is a decision tree. Look at the accuracy statistics. There is something profoundly wrong. What's the problem? Try using other classifiers (except for SVM). Report what works. Consider using preprocessing filters such as Preprocessing->Attributes->Filter->Numeric2Binary to transform the data before classifying it. Numeric2Binary replaces the word counts by indicators of whether the word was present or not. Why can it help accuracy?

(b) Choosing Parameters

Load `spam-grid.xml`. This experiment uses the SVM as a classifier, and the SVM has several parameters. All kernel types have the C parameter, polynomial kernels have the degree, gamma and coef0 parameters, rbf kernels have gamma. These can be chosen by grid search, with cross-validation used to estimate performance with a certain set of parameters.

Currently, the experiment searches for the best C and degree for the polynomial kernel. Modify it to find the best coef0 as well; also try changing the kernel to rbf and selecting the gamma (but don't sweep across coef0 and degree when using the rbf; that'll just waste cycles).

Look at the log file to see what the accuracy is with each set of parameters. Try to understand and explain why changing the C has an effect on the accuracy and why having a too high C is not optimal (hint, higher C penalizes the SVM more for selecting a function which makes mistakes or comes close to making mistakes on some data).

(c) Training, validation, testing

In 2a) and b) we have been abusing our data. Since we ran crossvalidation on the whole dataset, the resulting choice of model parameters may overfit the data. Explain what the problem is, since we never train on the data we test on. Why is using the same data set for selecting parameters and estimating final performance a bad idea?

If you are interested, `spam-fancy.xml` sets up the proper experiment. However, due to some peculiarities of YALE, it is fairly complex and it is not trivial to change classifiers.

(3) Regression

In this question we will fit linear regressions to predict housing prices in Boston. The data is given in `housing.data` and a description of the column values is given in `housing.names`. For our purpose, the input variables ($X$) are the first 13 columns of the dataset, and the response variables are in the last column. Augment the data $X$ with a constant *intercept term* before fitting the regressions. Recall that you almost never need to explicitly compute a matrix inverse in practice. Instead, solve a linear system (for example via the backslash operator in Matlab) whenever possible.

(a) Linear regression

Recall that in standard linear regression we model the $X$-$\mathbf{y}$ dependence as

$$y = f(x; \hat{\mathbf{w}}) = \hat{w}_0 + \sum_{i=1}^{d} \hat{w}_i x_i,$$

where

$$\hat{\mathbf{w}} = \text{argmin}_{\mathbf{w}} \sum_{t=1}^{n} \left(y_t - f(x_t; \mathbf{w})\right)^2.$$

Begin by writing three functions:

- A function that takes as input a set of training data $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n), \mathbf{y} = (y_1, \ldots, y_n)$ and produces as output an estimate of the optimal weight vector $\hat{\mathbf{w}}$.
- A function that takes a weight vector $\mathbf{w}$ and a case $\mathbf{x}$ and produces a prediction of the associated value $y$.
- A function that takes as input a training set of input and output values and a test set of input and output values and returns the mean squared training error and the mean squared test error.

To test our regression, divide the data into a training and test set. For a given training set size use the first few rows of the housing data as the training set and the rest as the test set. Report the mean squared training and test error for training sets of sizes 200, 300, and 400. Also submit your code for computing $\hat{\mathbf{w}}$.

(b) Ridge regression

It turns out that for smaller training sets the calculations of the optimal weight-vector above become unstable. A common solution is to use a regularization strategy for the magnitude of the vector $\mathbf{w}$. Make a small change to your code so that it now optimizes the objective

$$\hat{\mathbf{w}} = \text{argmin}_{\mathbf{w}} \sum_{t=1}^{n} \left(y_t - f(x_t; \mathbf{w})\right)^2 + \lambda \mathbf{w}^\top \mathbf{w},$$

with $\lambda = 10^{-6}$. Use this new regression model to compute mean squared training and test error for training set sizes 10, 50, 100, 200, 300, 400. Submit your updated code for computing $\hat{\mathbf{w}}$.

(c) Polynomial regression

By computing new features from our existing data we can considerably increase the flexibility of the linear regression framework. Particularly, we will now model the dependence as

$$y = \hat{w}_0 + \sum_{i=1}^{d} \sum_{p=1}^{m} \hat{w}_{i,p} x_i^p,$$

for some choice of maximum degree $m$. Make small adjustments to the code of the previous question so that it now computes the ridge regression solution $\mathbf{w}$ for the expanded set of features. Some of the new features we added to $X$ introduced columns that have significantly larger magnitude than other columns, which can create problems at compute time. To sidestep this issue, it is good practice to rescale the columns prior to fitting the regression. Rescale your training features and use the same rescaling on the test data. For fitting this model, let $\lambda = 10^{-6}$. Report the mean squared training and test error for values of $m = 0, \ldots, 10$. For the case of $m = 0$ only leave the intercept term of $X$. Argue in two sentences why we expect the training error to be monotonically decreasing with maximal degree $m$. Which value $m$ would you choose to evaluate your model on an unknown test set?

To check your results, for $m = 2$ you should see a training error of about 14.5 and a test error of about 32.8.

(4) Properties of Ordinary Least Squares

Recall the normal equations that were used to solve the ordinary least squares problem:

$$(X^\top X)w = X^\top y.$$

(a) Invariance to rescaling

In the previous question you rescaled the data prior to fitting a regression. By appealing to the normal equations, prove that rescaling the data will not change the predicted values $\hat{y}_*$ for test data $x_*$.

(b) Perfect fitting

What property must hold for the training data $X$ so that the training error will be zero for any set of training responses $y$?