

# CS 294 - Homework 3

March 9, 2008

This homework is due on March 18 at 5pm on bSpace. If you have questions, please contact Alexandre Bouchard (bouchard@cs.berkeley.edu) for question 1 and Gad Kimmel (kimmel@cs.berkeley.edu) for question 2.

1. (From Ben Blum) This question is based around the R code in featsel.R; you shouldn't execute this script directly, but you can copy and paste the code from it into an R session (or use it as a starting point to write your own R scripts).

The idea behind this exercise is to explore how filtering performs versus Lasso regression and forward selection.

- (a) Generate a dataset of size 100 with 10 features, following the instructions in featsel.R. Create a response variable that depends linearly on just two of the covariates, with some added noise (follow the guidelines in featsel.R closely). Perform filtering, using as a score the squared Pearson correlation coefficient. The Pearson correlation coefficient is a popular measure of how related two random variables are. It can be computed as

$$\text{Corr}(X, Y) := \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}}$$

From a finite sample of points  $(X_i, Y_i)$ , this is estimated as

$$\frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2} \sqrt{\sum_i (Y_i - \bar{Y})^2}}$$

It lies between  $-1$  and  $1$ . It is equal to  $1$  if and only if the variables are perfectly correlated, and is equal to  $-1$  if and only if they are perfectly anticorrelated. Generate random datasets at least five different times, running filtering each time. Suppose we want a model with two variables; we will discard all but the top two scoring features. Does this filtering score consistently rank the top two relevant features as the best? If not, why not? (see featsel.R for a hint)

- (b) On each of the datasets you used for filtering in part (a), fit a sparse linear model using LARS. Print out the plot.lars output (just one graph will do, for one of the five runs). This output traces the weights for each feature as the regularization coefficient decreases. As it decreases, do the top two relevant features consistently receive nonzero weight before any others?
- (c) Make  $y$  a different linear function of several of the covariates—your choice. Pick a new dataset size—perhaps increase the ratio of features to examples. You can alter the variance of the noise term if you like by multiplying it by a constant. Redo parts (a) and (b).
- (d) Generate a dataset of size 1000 with 1000 features; this dataset is a little large for LARS. Perform filtering using the squared Pearson correlation. Suppose we want a model with just two variables. Does filtering on its own consistently give us the best model? Now try filtering out all but 200 features and then using LARS. Does this combination consistently pick out the correct model? Does it run quickly?

- (e) Now we'll take our dataset into YALE to try forward selection. First, output one of your generated datasets from part (a) to a file, following the instructions in featsel.R. Pick a dataset for which filtering failed to rank the relevant features highest. Name the file featsel.dat and put it in the same directory as featsel.aml and featsel.xml. Now open the experiment "featsel.xml" in YALE. Look through the operator chain; it performs forward selection using cross validation with squared error as an objective function, with linear regression as the internal learning module. Edit the "attributes" key in the "ExampleSource" operator to point to featsel.aml, edit the "filename" key in the "ExperimentLog" operator to specify an output file, and run the experiment. See which features were selected by clicking the ExampleSource tab in the results section. What were they?
  - (f) Optional: Try feature selection algorithms on a problem from your own research (or from the UCI Machine Learning Repository). For instance, compare LARS regression against  $L_2$  regularized and unregularized regression. What are the number of non-zero features selected? Is there a gain of accuracy? Is regression (or classification) faster at test time?
2. Download the file "case-control\_dataset.zip" and unzip it. The zipped file contains two files: "cases" and "controls". cases contains mean blood pressure measurements of 10,000 samples who have a disease, and controls contains mean blood pressure measurements of healthy 10,000 samples. The goal is to find a learning algorithm which classifies samples based on their mean blood pressure measurement into case or control. The measurement is also called a *trait*.

We consider two different classifiers (or algorithms):

- (a) If the trait  $> t$  - classify as case. Otherwise, classify as control.
- (b) If the trait  $< t$  - classify as case. Otherwise, classify as control.

Where  $t$  is the parameter of the algorithm.

- (a) Draw ROC plots for both algorithms. Try different values for  $t$  starting from 70 to 150.
- (b) For both classifiers, calculate the value of the true positive rate, given that the false positive rate equals 0.5.
- (c) Which of the classifiers seems to be a better one for the problem? Explain your answer.