Statistics 151B Homework 3 Due: 2 PM Mon 19 March 2012

Theory

Submit written solutions for these problems at the beginning of section on the due date.

1. ESL 5.1.

   Hint. First argue that any cubic spline with the two indicated knots can be represented as a linear combination of the functions in (5.3). Then argue that any linear combination of those functions must be a cubic spline with the indicated knots.

2. ESL 10.1.

3. Let $\mathcal{G} = \{g(x; \gamma) \mid \gamma \in \Gamma\}$ be a base family for a boosting procedure. Each $g \in \mathcal{G}$ maps x to $\{-1, +1\}$. Suppose $\mathcal{G}$ is *closed under negation*: for each $g(\cdot; \gamma) \in \mathcal{G}$, there is some $\gamma' \in \Gamma$ such that $g(\cdot; \gamma') = -g(\cdot; \gamma)$.

   Show that the minimal weighted training error attainable using $\mathcal{G}$, on any training set and with any weights, cannot exceed $1/2$. (Assume the weights are non-negative; otherwise the question does not make sense.) Show that the same holds for the test error, i.e., misclassification probability, defined for a classifier $f(x)$ as

   $$P(Y \neq f(X)) = E_{X,Y} 1[Y \neq f(X)] .$$

4. Consider an (unintelligent) classifier $f^*(x)$ which ignores x and produces as its predicted label Z, a $\{0, 1\}$-valued random variable such that $P(Z = 1) = P(Z = 0) = 1/2$. Z is independent of all other random variables. Show that, averaged over the choice of Z, the misclassification probability of f^* is 1/2. (This is a different way to argue we can never do worse than a test error rate of 50% in binary classification.)

5. ESL 10.2.

Choosing a complexity parameter by cross-validation

We revisit the abalone data set of homework 2. The goal is to model the relationship between rings and the predictor variables using the lasso, with the lasso complexity parameter λ set by 5-fold cross-validation. To recall how one fits and predicts with the lasso in R, review the relevant portion of homework 2.

- Load `abalone.data` into R as a data frame `abalone`.
- Make a list `all.folds` with 5 entries: the ith entry of the list is a vector containing the indexes of the abalone data frame rows that you randomly assigned to fold i. For example, if `all.folds[[1]]` contained the vector $(3, 11, 4)$, then the third, eleventh, and fourth rows of the abalone data frame would belong in fold 1. (There will be many more rows

than that in each fold.) To create `all.folds`, use the functions `split()`, `sample()`, and `rep()`. (Notice that `sample()` can be used to generate a random permutation of the integers $1, \ldots, n$.)

- Make a vector `beta.fracs` of 100 equally spaced values between 0 and 1. These correspond to the λ values you will consider in the cross-validation. Specifically, each entry in `beta.frac` gives the desired 1-norm of the lasso solution vector, expressed as a fraction of the OLS solution's 1-norm. For example, if `beta.frac[50]` = 0.50, then it corresponds to a value of λ whose corresponding β_λ has $\|\beta_\lambda\| = 0.5\|\beta_{\text{OLS}}\|$.

- Make a vector `lambda.cv.err` of 100 zeroes – after the next step, this vector will contain the cross-validated estimate of prediction error for each value in `beta.fracs`.

- Use a `for` loop to compute the cross-validated estimate of prediction error, for each value in `beta.fracs`. The `for` loop should be over the integers $1, \ldots, 5$. On the ith pass through the loop, you should

  - fit the lasso (one fit per entry in `beta.fracs`) to all the data except the rows in `all.folds[[i]]`
  - use the fit to predict the y values in `all.folds[[i]]`, with one vector of predictions for each entry in `beta.fracs` (see the help page for `predict.lars`)
  - add the mean-squared prediction error (across observations in fold i) at each value in `beta.fracs` into `lambda.cv.err`

  After the `for` loop, divide each entry in `lambda.cv.err` by 5.

- Make a plot of `lambda.cv.err` against `beta.fracs`. Comment on the plot.

What to turn in for this problem: email the following three files to the GSI and myself by the deadline.

1. A file containing the R commands you used. Call it `commands.txt`. This file should have only your input to R.

2. A file containing a transcript of your R session. Call it `transcript.txt`.

3. A file with the figure you generated, and a discussion of your analysis.