Statistical Machine Learning GU4241/GR5241
Spring 2019
https://courseworks.columbia.edu/

Homework 4

Due: Thursday, April, 18th, 2019

Homework submission: Please submit your homework electronically through Gradescope (pdf file) and Canvas (code) by 11:59pm on the due date. You need to submit both the pdf file and your code (either in R or Python).

Problem 1 (Ridge Regression and Lasso for Correlated Variables, ISL 6.5, 4 points)

It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.

Suppose that $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\hat{\beta}_0 = 0$.

Homework Problems.

1. Write out the ridge regression optimization problem in this setting.
2. Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$.
3. Write out the lasso optimization problem in this setting.
4. Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique—in other words, there are many possible solutions to the optimization problem in 3. Describe these solutions.

Problem 2 (Regression Trees, ISL 8.4, 3 points)

(a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 1. The numbers inside the boxes indicate the mean of Y within each region.

(b) Create a diagram similar to the left-hand panel of Figure 1, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.

Problem 3 (Bagging, ISL 8.5, 3 points)

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of p (Class is Red$|X$):

$$0.1,\ 0.15,\ 0.2,\ 0.2,\ 0.55,\ 0.6,\ 0.6,\ 0.65,\ 0.7,\ \text{and}\ 0.75.$$

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in class. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

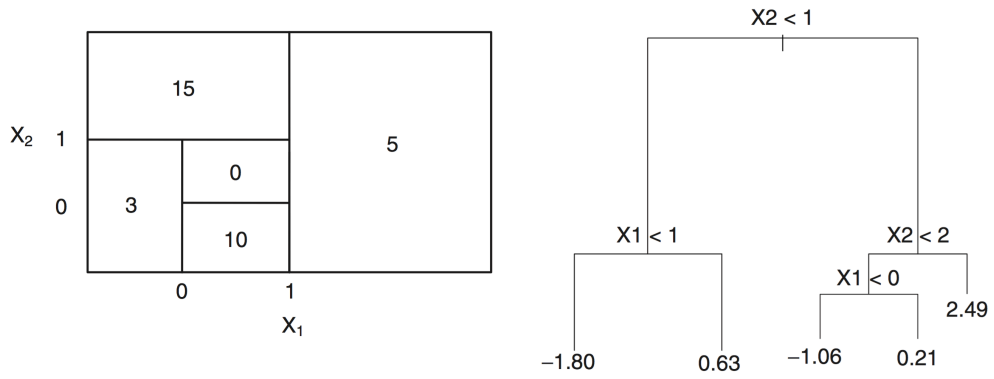

Figure 1: Left: *A partition of the predictor space corresponding to part a*. Right: *A tree corresponding to part b*.

Problem 4 (Boosting, 20 points)

The objective of this problem is to implement the AdaBoost algorithm. We will test the algorithm on handwritten digits from the USPS data set.

AdaBoost: Assume we are given a training sample $(\mathbf{x}^{(i)}, y_i), i = 1, ..., n$, where $\mathbf{x}^{(i)}$ are data values in $\mathbb{R}^d$ and $y_i \in \{-1, +1\}$ are class labels. Along with the training data, we provide the algorithm with a training routine for some classifier c (the "weak learner"). Here is the AdaBoost algorithm for the two-class problem:

1. Initialize weights: $w_i = \frac{1}{n}$
2. for $b = 1, ..., B$
   (a) Train a weak learner c_b on the weighted training data.
   (b) Compute error: $\epsilon_b := \frac{\sum_{i=1}^n w_i \mathbb{I}\{y_i \neq c_b(\mathbf{x}^{(i)})\}}{\sum_{i=1}^n w_i}$
   (c) Compute voting weights: $\alpha_b = \log\left(\frac{1-\epsilon_b}{\epsilon_b}\right)$
   (d) Recompute weights: $w_i = w_i \exp\left(\alpha_b \mathbb{I}\{y_i \neq c_b(\mathbf{x}^{(i)})\}\right)$
3. Return classifier $\hat{c}_B(\mathbf{x}^{(i)}) = \operatorname{sgn}\left(\sum_{b=1}^B \alpha_b c_b(\mathbf{x}^{(i)})\right)$

Decision stumps: Recall that a stump classifier c is defined by

$$c(\mathbf{x}|j, \theta, m) := \begin{cases} +m & x_j > \theta \\ -m & \text{otherwise.} \end{cases} \tag{1}$$

Since the stump ignores all entries of $\mathbf{x}$ except x_j, it is equivalent to a linear classifier defined by an affine hyperplane. The plane is orthogonal to the jth axis, with which it intersects at $x_j = \theta$. The orientation of the hyperplane is determined by $m \in \{-1, +1\}$. We will employ stumps as weak learners in our boosting algorithm. To train stumps on weighted data, use the learning rule

$$(j^*, \theta^*) := \arg\min_{j,\theta} \frac{\sum_{i=1}^n w_i \mathbb{I}\{y_i \neq c(\mathbf{x}^{(i)}|j, \theta, m)\}}{\sum_{i=1}^n w_i} \,. \tag{2}$$

In the implementation of your training routine, first determine an optimal parameter θ_j^* for each dimension $j = 1, ..., d$, and then select the j^* for which the cost term in (2) is minimal.

Homework problems:

1. Implement the AdaBoost algorithm in R. The algorithm requires two auxiliary functions, to train and evaluate the weak learner. We also need a third function which implements the resulting boosting classifier. We will use decision stumps as weak learners, but a good implementation of the boosting algorithm should permit you to easily plug in arbitrary weak learners. To make sure that is possible, please use function calls of the following form:

   - `pars <- train(X, w, y)` for the weak learner training routine, where `X` is a matrix the columns of which are the training vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, and `w` and `y` are vectors containing the weights and class labels. The output `pars` is a list which contains the parameters specifying the resulting classifier. (In our case, `pars` will be the triplet (j, θ, m) which specifies the decision stump).
   - `label <- classify(X, pars)` for the classification routine, which evaluates the weak learner on `X` using the parametrization `pars`.
   - A function `c_hat <- agg_class(X, alpha, allPars)` which evaluates the boosting classifier ("aggregated classifier") on `X`. The argument `alpha` denotes the vector of voting weights and `allPars` contains the parameters of all weak learners.

2. Implement the functions `train` and `classify` for decision stumps.

3. Run your algorithm on the USPS data (the digit data we used in Homework 3, use the training and test data for the 3s and 8s) and evaluate your results using cross validation.

   More precisely: Your AdaBoost algorithm returns a classifier that is a combination of B weak learners. Since it is an incremental algorithm, we can evaluate the AdaBoost at every iteration b by considering the sum up to the b-th weak learner. At each iteration, perform 5-fold cross validation to estimate the training and test error of the current classifier (that is, the errors measured on the cross validation training and test sets, respectively).

4. Plot the training error and the test error as a function of b.

Submission. Please make sure your solution contains the following:

- Your implementation for `train`, `classify` and `agg_class`.
- Your implementation of `AdaBoost`.
- Plots of your results (training error and cross-validated test error).