ISYE 6740 Homework 6
Fall 2024
Prof. Yao Xie
Total 100 points

1. **Conceptual questions.** (25 points)

   1.1. (5 points) What's the main difference between boosting and bagging? The random forest belongs to which type?

   Answer:

   Boosting combines weak learners to generate a stronger learner. Bagging runs weaker learners on bootstrap replicates of the training data. Random forest belongs to bagging, because bagging reduces variance especially for unstable learner like decision tree. And random forest builds multiple random forest.

   1.2. (5 points) List several ways to prevent overfitting in CART.

   Answer:

   Tree Pre-pruning: stops the growth of the decision tree and prevents it from reaching its full depth.

   Tree Post-Pruning: grow decision tree to its full depth, removes branches by minimizing cost function.

   1.3. (5 points) Explain how we control the data-fit complexity in the regression tree. Name at least one hyperparameter that we can turn to achieve this goal.

   Answer:

   Maximum depth of the decision tree, if it is too high, the decision trees learn too fine details of the training data and learn from the noise. By setting a limit on how deep the tree grows, we resricted the complexity of the model.

   1.4. (5 points) Explain how OOB errors are constructed and how to use them to understand a good choice for the number of trees in a random forest. Is OOB an error test or training error, and why?

   Answer:

   Because the bootstrap algorithm randomly get sample of the training data with replacement, a part of the data are not included in the bootstrap sample, which is called out-of-bag data(OOB).

   The overall OOB error is the average of errors across all instances that were not used in bootstrap samples.

   The OOB error typically decreases with the number of tress increases, so when the OOB error stabilizes, the number of trees is a good choice.

   OOB error is the mean prediction error on each training sample so it can be considered as training error.

1.5. (5 points) Explain what the bias-variance tradeoff means in the linear regression setting.

Answer:

The bias-variance tradeoff in the linear regression setting refers to balance between minimizing the mean square error and minimizing the error for unseen test data points with respect to the entire distribution of data. A high bias leads to underfitting and the variance would be low. A high variance results in overfitting and the bias would be low.

2. **AdaBoost.** (25 points)

Consider the following dataset, plotted in the following figure. The first two coordinates represent the value of two features, and the last coordinate is the binary label of the data.

$$X_1 = (-1, 0, +1), X_2 = (-0.5, 0.5, +1), X_3 = (0, 1, -1), X_4 = (0.5, 1, -1),$$
$$X_5 = (1, 0, +1), X_6 = (1, -1, +1), X_7 = (0, -1, -1), X_8 = (0, 0, -1).$$

In this problem, you will run through $T = 3$ iterations of AdaBoost with decision stumps (as explained in the lecture) as weak learners.

2.1. (15 points) For each iteration $t = 1, 2, 3$, compute ϵ_t, α_t, Z_t, D_t by hand (i.e., show the calculation steps) and draw the decision stumps on the figure (you can draw this by hand).

Answer:

t = 1

$D_1(i) = \frac{1}{8}$, decision stump h_1 showed in the following plot, we can see that the point 5 and 6 are misclassfied. So we can get

$$\epsilon_1 = \frac{1}{4}$$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) \approx 0.54931$$

There are 6 correctly classified points and 2 misclassified points:

$$Z_1 = 6 \times \frac{1}{8}\exp(-\alpha_1) + 2 \times \frac{1}{8}\exp(\alpha_1) = 0.86603$$

t = 2

For points 1, 2, 3, 4, 7, 8, the $D_2(i) = \frac{D_1(i)}{Z_1}e^{-\alpha_1 y^{(i)} h_1(x^{(i)})} = 0.08333$.

For points 5, 6, which have bigger weight, the $D_2(i) = \frac{1}{8\times 0.866}e^{\frac{1}{2}\ln 3} = 0.2500$.

The incorrectly classified pints are 1 and 2, so

$$\epsilon_2 = 2 * D_2(i) = 0.16667$$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 0.8047$$

There are 6 correctly classified points and 2 misclassified points:

$$Z_2 = \sum_{i=1}^{m} D_2(i)e^{-\alpha_2 y^{(i)} h_2(x^{(i)})} = \sum_{i\in\{1,2\}} D_2(i)e^{\alpha_2} + \sum_{i\in\{3,4,7,8\}} D_2(i)e^{-\alpha_2} + \sum_{i\in\{5,6\}} D_2(i)e^{-\alpha_2} = 0.74536$$

t = 3

For points 3, 4, 7, 8, the $D_3(i) = \frac{D_2(i)}{Z_2}e^{-\alpha_2 y^{(i)} h_1(x^{(i}} = \frac{0.08333}{0.74536}e^{-0.8047} = 0.05$.

For points 1 and 2, the $D_3(i) = \frac{0.08333}{0.74536}e^{0.8047} = 0.25$.
For points 5 and 6, the $D_3(i) = \frac{0.25}{0.74536}e^{-0.8047} = 0.15$.
The incorrectly classified pints are 7 and 8, so

$$\epsilon_2 = 2 * D_3(i) = 0.1$$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 1.09861$$

There are 6 correctly classified points and 2 misclassified points:

$$Z_3 = 0.6$$

2.2. (10 points) What is the training error of this AdaBoost? Give a short explanation for why AdaBoost outperforms a single decision stump.

Answer:

$$\begin{aligned} f_3(x) &= \text{sign}(\alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x)) \\ &= \text{sign}(0.5493 - 0.8047 + 1.0986) + \text{sign}(0.5493 + 0.8047 + 1.0986) \\ &+ \text{sign}(0.5493 + 0.8047 - 1.0986) + \text{sign}(-0.5493 + 0.8047 + 1.0986) \\ &= 0 \end{aligned}$$

Based on the calculation above, we can get the training error of this AdaBoost is 0. Adaboost contains several single decision stump that each single decision stump will improve the performance based on the previous one. Then, they combined each linear boundary together to be a complicated decision boundary.

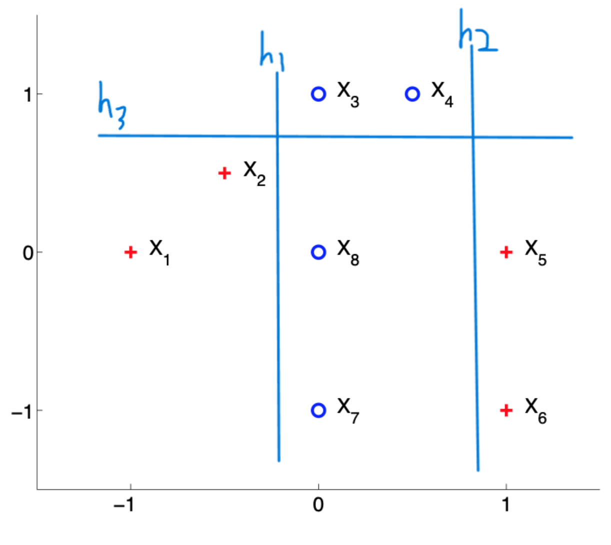


Figure 1: A small dataset for binary classification with AdaBoost.

Table 1: Values of AdaBoost parameters at each timestep.

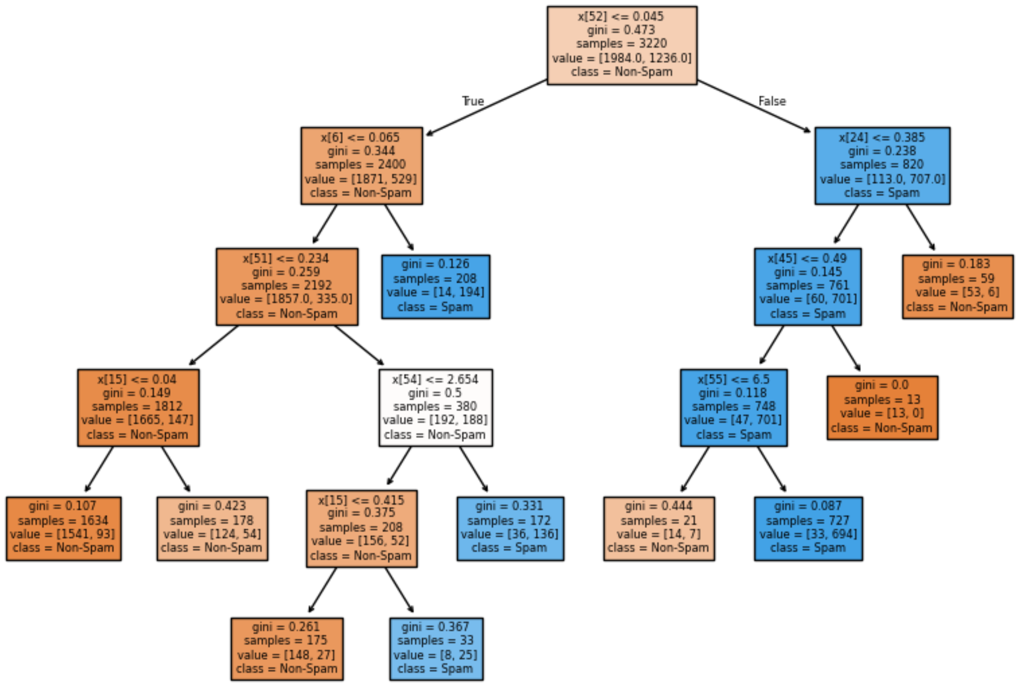
t	ϵ_t	α_t	Z_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$
1	0.25	0.54931	0.86603	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
2	0.16667	0.8047	0.74536	0.08333	0.08333	0.08333	0.08333	0.25	0.25	0.08333	0.08333
3	0.1	1.09861	0.6	0.25	0.25	0.05	0.05	0.15	0.15	0.05	0.05

3. **Random forest and one-class SVM for email spam classifier** (30 points)

Your task for this question is to build a spam classifier using the UCR email spam dataset `https://archive.ics.uci.edu/ml/datasets/Spambase` came from the postmaster and individuals who had filed spam. Please download the data from this link. Headers are not necessary for this problem, but can be obtained by the 'spambase.names' file. The collection of non-spam emails came from filed work and personal emails, and hence the word '`george`' and the area code '`650`' (Palo Alto, CA) are indicators of non-spam. These are useful when constructing a personalized spam filter. You are free to choose any package for this homework. Note: there may be some missing values. You can just fill in zero. You should use the same train-test split for every part.

3.1. (5 points) Build a CART model and visualize the fitted classification tree. Please adjust the plot size/font as appropriate to ensure it is legible. Pruning this tree is not required, and if done reasoning should be stated as to why.

Answer:



x[52] <= 0.045 gini = 0.473 samples = 3220 value = [1984.0, 1236.0] class = Non-Spam
True
False
x[6] <= 0.065 gini = 0.344 samples = 2400 value = [1871, 529] class = Non-Spam
x[24] <= 0.385 gini = 0.238 samples = 820 value = [113.0, 707.0] class = Spam
x[51] <= 0.234 gini = 0.259 samples = 2192 value = [1857.0, 335.0] class = Non-Spam
gini = 0.126 samples = 208 value = [14, 194] class = Spam
x[45] <= 0.49 gini = 0.145 samples = 761 value = [60, 701] class = Spam
gini = 0.183 samples = 59 value = [53, 6] class = Non-Spam
x[15] <= 0.04 gini = 0.149 samples = 1812 value = [1665, 147] class = Non-Spam
x[54] <= 2.654 gini = 0.5 samples = 380 value = [192, 188] class = Non-Spam
x[55] <= 6.5 gini = 0.118 samples = 748 value = [47, 701] class = Spam
gini = 0.0 samples = 13 value = [13, 0] class = Non-Spam
gini = 0.107 samples = 1634 value = [1541, 93] class = Non-Spam
gini = 0.423 samples = 178 value = [124, 54] class = Non-Spam
x[15] <= 0.415 gini = 0.375 samples = 208 value = [156, 52] class = Non-Spam
gini = 0.331 samples = 172 value = [36, 136] class = Spam
gini = 0.444 samples = 21 value = [14, 7] class = Non-Spam
gini = 0.087 samples = 727 value = [33, 694] class = Spam
gini = 0.261 samples = 175 value = [148, 27] class = Non-Spam
gini = 0.367 samples = 33 value = [8, 25] class = Spam

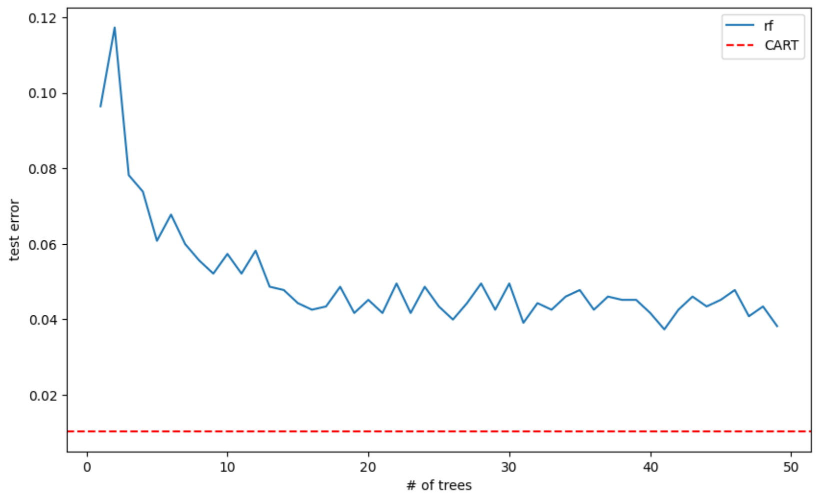
I use max_leaf_nodes=10 in 'DecisionTreeClassifier' function to limit the number of nodes. we can see that the gini decreases and there is a gini = 0, which indicates that the node correctly classify the spam/non-spam emails.

3.2. (5 points) Now, also build a random forest model. Randomly shuffle the data and partition to use 75% for training and the remaining 25% for testing. Compare and report the test error for your classification tree and random forest models on testing data. Plot the curve of test error (total misclassification error rate) versus the number of trees for the random forest, and plot the test error for the CART model (which should be a constant with respect to the number of trees).

Answer:

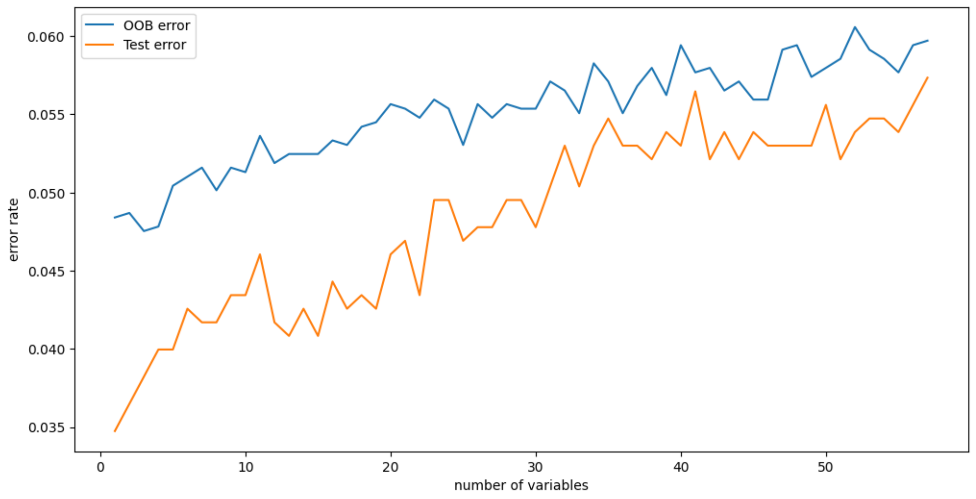
the test error for CART: 0.01043

the test error for random forest: 0.04257



3.3. (10 points) Fit a series of random-forest classifiers to the data to explore the sensitivity to the parameter ν (the number of variables selected at random to split). Plot both the OOB error as well as the test error against a suitably chosen range of values for ν.

Answer:



3.4. (10 points) Now, we will use a one-class SVM approach for spam filtering. Randomly shuffle the data and partition to use 75% for training and the remaining 25% for testing. Extract all *non-spam* emails from the training block (75% of data you have selected) to build the one-class kernel SVM using RBF kernel. Then apply it to the 25% of data reserved for testing (thus, this is a novelty detection situation), and report the total misclassification error rate on these testing data. Tune your models appropriately to achieve good performance, i.e. by tuning the kernal bandwidth or other parameters. Give a short explanation on how you reached your final error rate and whether you feel this is a good model.

Answer:

Only non-spam emails are used to train the one-class SVM, and the test error rate is 0.384 which is pretty high, so I do not think it is a good model.

4. **Locally weighted linear regression and bias-variance tradeoff.** (20 points)

The idea of locally weighted linear regression is that we will give more weight to data points in the training data that are close to the point at which we want to make a prediction. This can improve the bias but will also face a bias-variance tradeoff. This homework question is designed to look into this problem.

Denote data point as (x_i, y_i), $x_i \in \mathbb{R}^p$, $i = 1, \dots, n$. Given a Gaussian kernel function

$$K_h(z) = \frac{1}{(\sqrt{2\pi}h)^p} e^{-\frac{\|z\|^2}{2h^2}}, \quad z \in \mathbb{R}^p.$$

Local linear regression solves $\beta_0 \in \mathbb{R}$, $\beta_1 \in \mathbb{R}^p$, for a given predictor $x \in \mathbb{R}^p$:

$$\widehat{\beta} := (\widehat{\beta}_0, \widehat{\beta}_1) = \arg\min \sum_{i=1}^{n} (y_i - \beta_0 - (x - x^i)^T \beta_1)^2 K_h(x - x_i)$$

4.1. (10 points) Show that the solution is given in the form

$$\widehat{\beta} = (X^T W X)^{-1} X^T W Y$$

for properly defined X, W, and Y (specify clearly what these need to be).

Answer:

The X matrix for locally weighted linear regression:

$$X = \begin{bmatrix} 1 & (x - x_1)^T \\ 1 & (x - x_2)^T \\ \vdots & \vdots \\ 1 & (x - x_n)^T \end{bmatrix}$$

The W matrix is a weight matrix:

$$W = \begin{bmatrix} K_h(x - x_1) & & \cdots & & 0 \\ & K_h(x - x_2) & & & \\ \vdots & & \ddots & & \vdots \\ & & & K_h(x - x_{n-1}) & \\ 0 & & \cdots & & K_h(x - x_n) \end{bmatrix}$$

The Y matrix:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

In order to minimize the weighted squared error, which can be written as:

$$L(\beta) = (Y - X\beta)^T W (Y - X\beta)$$

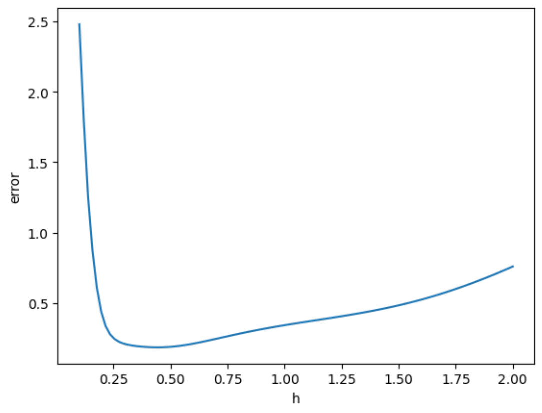
$$\frac{\partial L}{\partial \beta} = -2X^T W (Y - X\beta) = 0$$

$$X^T W Y = X^T W X \beta$$

$$\hat{\beta} = (X^T W X)^{-1} X^T W Y$$

4.2. (5 points) Use the data.mat file to perform local linear weighted linear regression. Using 5-fold cross validation to tune the bandwidth parameter h, report a plot showing your cross validation curve and provide your optimal bandwidth, h.

Answer: h = 0.44545



4.3. (5 points) Using the tuned hyper-parameter h to make a prediction for $x = -1.5$. Provide the predicted y value, and report a plot showing your training data, prediction curve, and a marker indicating your prediction.

Answer:

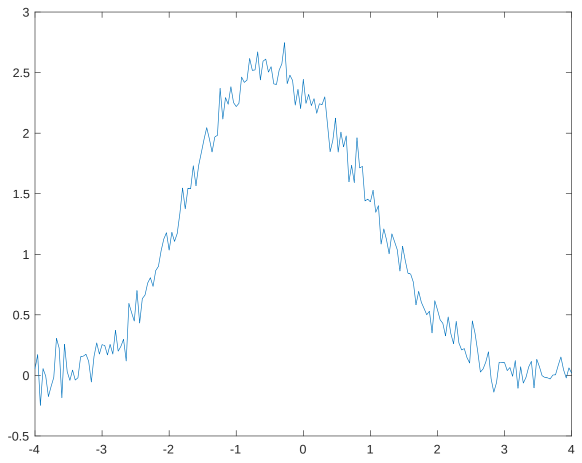


Figure 2: Illustration of noisy curve.