ISYE 6740 Homework 6
Fall 2020

Total 100 points.

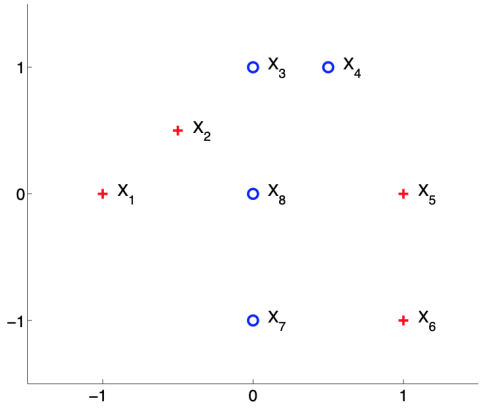
Farshad Rafiei

1. **AdaBoost.** (30 points)

Consider the following dataset, plotting 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.



(a) (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:

step 1:

Let's assume that h_1 (Figure 1.A) returns +1 when $x_1 < -0.25$ and -1, otherwise (All points are in the form of (x_1, x_2, y)).

$D_1 = \frac{1}{8} = 0.125$ for all points.

$\epsilon_1 = \sum_{i=1}^{m} D_1(i)\mathbb{1}(y^i \neq h_1(x^i)) = 2 \times 0.125 = 0.25$

$\alpha_1 = \frac{1}{2}\ln(\frac{1-\epsilon_1}{\epsilon_1}) = \frac{1}{2}\ln(\frac{1-0.25}{0.25}) = 0.5493$

$Z_1 = \sum_{i=1}^{m} D_1(i)e^{-\alpha_1 y^i h_1(x^i)} = 6 \times 0.125 \times e^{-0.5493} + 2 \times 0.125 \times e^{0.5493} = 0.8660$

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	0.2500	0.5493	0.8660	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1666	0.8050	0.7451	0.0833	0.0833	0.0833	0.0833	0.2500	0.2500	0.0833	0.0833
3	0.1000	1.0986	0.6001	0.2501	0.2501	0.0500	0.0500	0.1500	0.1500	0.0500	0.0500

$D_2(1) = D_2(2) = D_2(3) = D_2(4) = D_2(7) = D_2(8) = \frac{D_1(1)}{Z_1} e^{-\alpha_1} = \frac{0.125}{0.8660} e^{-0.5493} = 0.0833$

$D_2(5) = D_2(6) = \frac{D_1(5)}{Z_1} e^{\alpha_1} = \frac{0.125}{0.8660} e^{0.5493} = 0.2500$

step 2:

Let's assume that h_2 (Figure 1.B) returns +1 when $x_1 > 0.75$ and -1, otherwise (All points are in the form of (x_1, x_2, y)).

$\epsilon_2 = \sum_{i=1}^{m} D_2(i) \mathbb{1}(y^i \neq h_2(x^i)) = 2 \times 0.0833 = 0.1666$

$\alpha_2 = \frac{1}{2} \ln(\frac{1-\epsilon_2}{\epsilon_2}) = \frac{1}{2} \ln(\frac{1-0.1666}{0.1666}) = 0.8050$

$Z_2 = \sum_{i=1}^{m} D_2(i) e^{-\alpha_2 y^i h_2(x^i)} = 2 \times 0.0833 \times e^{0.8050} + (4 \times 0.0833 + 2 \times 0.25) \times e^{-0.8050} = 0.7451$

$D_3(1) = D_3(2) = \frac{D_2(1)}{Z_2} e^{\alpha_2} = \frac{0.0833}{0.7451} e^{0.8050} = 0.2501$

$D_3(5) = D_3(6) = \frac{D_2(5)}{Z_2} e^{-\alpha_2} = \frac{0.25}{0.7451} e^{-0.8050} = 0.1500$

$D_3(3) = D_3(4) = D_3(7) = D_3(8) = \frac{D_2(3)}{Z_2} e^{-\alpha_2} = \frac{0.0833}{0.7451} e^{-0.8050} = 0.0500$

step 3:

Let's assume that h_3 (Figure 1.C) returns -1 when $x_2 > 0.75$ and +1, otherwise (All points are in the form of (x_1, x_2, y)).

$\epsilon_3 = \sum_{i=1}^{m} D_3(i) \mathbb{1}(y^i \neq h_3(x^i)) = 2 \times 0.0500 = 0.1000$

$\alpha_3 = \frac{1}{2} \ln(\frac{1-\epsilon_3}{\epsilon_3}) = \frac{1}{2} \ln(\frac{1-0.1000}{0.1000}) = 1.0986$

$Z_3 = \sum_{i=1}^{m} D_3(i) e^{-\alpha_3 y^i h_3(x^i)} = 2 \times 0.2501 \times e^{-1.0986} + 2 \times 0.0500 \times e^{-1.0986} + 2 \times 0.1500 \times e^{-1.0986} + 2 \times 0.0500 \times e^{1.0986} = 0.6001$

(b) (15 points) What is the training error of this AdaBoost? Give a short explanation for why AdaBoost outperforms a single decision stump.
Answer: The final Adaboost classifier will after 3 steps, will be as following:
$H = sign(0.5493h_1 + 0.8050h_2 + 1.0986h_3)$
The training error for this classifier is 0, since it classifies all of the instances correctly (accuracy = 1). It clearly outperforms a simple decision stump which at least makes two mistakes in training data. The reason that Adaboost performs better on this training data, is that it creates a nonlinear decision boundary (Figure 2) which is able to classify the data efficiently. On the

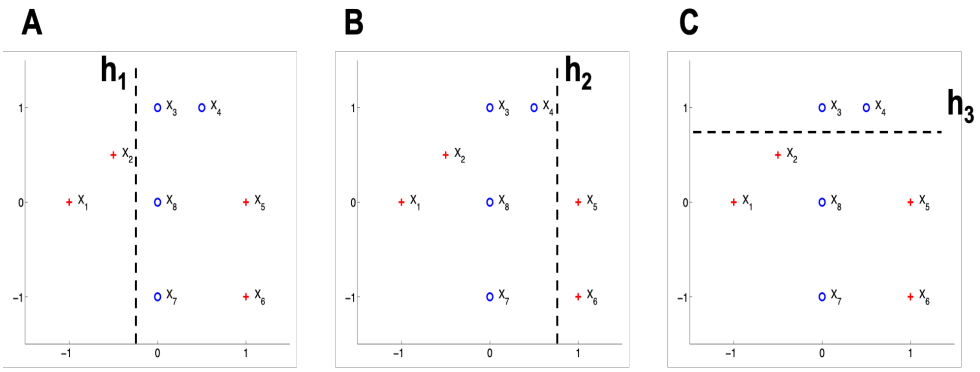


Figure 1: **Decision stumps generated in different steps of Adaboost**.

contrary, a single decision stump partitions the space into two parts which can not perform very well on separating the data with nonlinear nature.

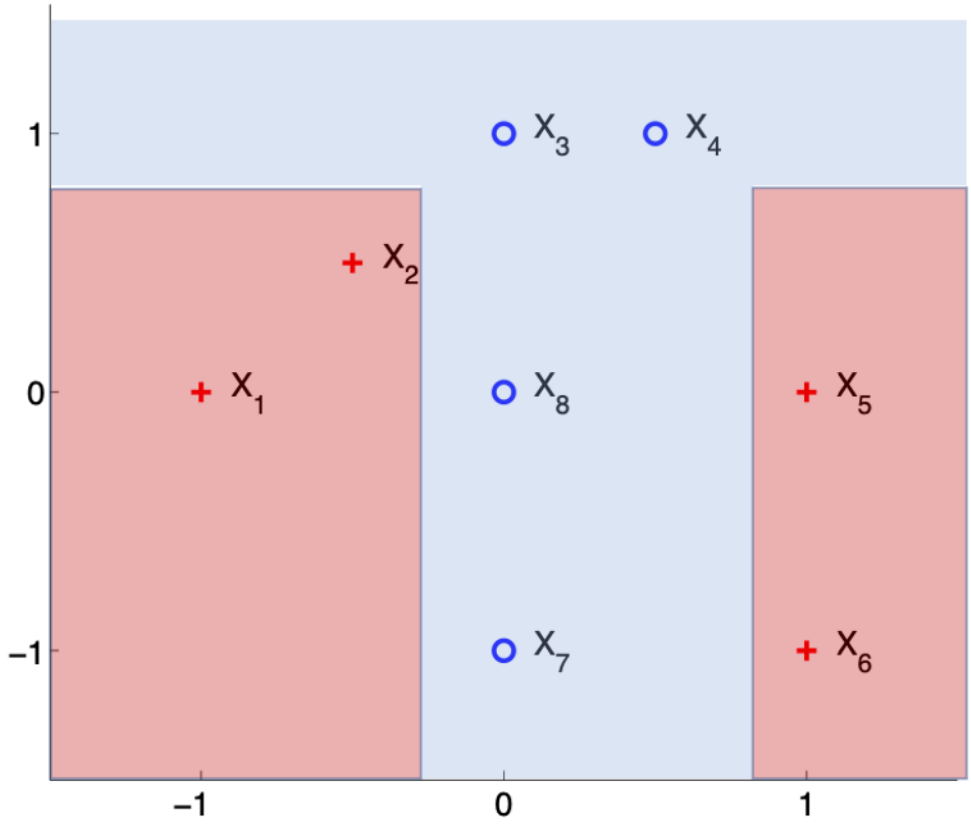


Figure 2: **Final Adaboost classifier**.

2. **Linear regression: bias-variance tradeoff, CV, and variable selection** (30 points)

Consider a dataset with n data points (x^i, y^i), $x^i \in \mathbb{R}^n$, following from the following linear model:

$$y^i = \beta^{*T} x^i + \epsilon^i, \quad i = 1, \ldots, m,$$

where ϵ^i are i.i.d. Gaussian noise with zero mean and variance σ^2, and β^* is the true parameter. Consider the ridge regression as follows:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \left\{ \frac{1}{m} \sum_{i=1}^{m} (y^i - \beta^T x^i)^2 + \lambda \|\beta\|_2^2 \right\}, \tag{1}$$

where $\lambda \geq 0$ is the regularized parameter.

(a) (5 points) Find the closed form solution for $\hat{\beta}(\lambda)$ and its distribution conditioning on $\{x^i\}$ (i.e., treat them as fixed).

Answer:

$$\begin{aligned} \frac{\partial}{\partial \beta} \frac{1}{m} \sum_{i=1}^{m} (y^i - \beta^T x^i)^2 + \lambda \|\beta\|_2^2 &= \frac{1}{m} \sum_{i=1}^{m} -2x^i (y^i - \beta^T x^i) + 2\lambda\beta \\ &= \frac{1}{m} \sum_{i=1}^{m} \{-2x^i y^i + 2\beta (x^i)^2\} + 2\lambda\beta \end{aligned} \tag{2}$$

By setting the above equation equal to zero, we have:

$$\frac{1}{m} \sum_{i=1}^{m} \{-2x^i y^i + 2\hat{\beta}(x^i)^2\} + 2\lambda\hat{\beta} = 0 \tag{3}$$

$$\rightarrow \hat{\beta} \{ \frac{1}{m} \sum_{i=1}^{m} (x^i)^2 + \lambda \} = \frac{1}{m} \sum_{i=1}^{m} x^i y^i$$

$$\rightarrow \hat{\beta} = \frac{\frac{1}{m} \sum_{i=1}^{m} x^i y^i}{\frac{1}{m} \sum_{i=1}^{m} (x^i)^2 + \lambda}$$

$$\rightarrow \hat{\beta} = \frac{\frac{1}{m} \sum_{i=1}^{m} x^i (\beta^{*T} x^i + \epsilon^i)}{\frac{1}{m} \sum_{i=1}^{m} (x^i)^2 + \lambda}$$

$$\rightarrow \boxed{\hat{\beta} = \frac{\frac{1}{m} \sum_{i=1}^{m} \beta^* (x^i)^2 + \frac{1}{m} \sum_{i=1}^{m} x^i \epsilon^i}{\frac{1}{m} \sum_{i=1}^{m} (x^i)^2 + \lambda}}$$

In matrix format, we can write the above equation as:

$$\hat{\beta} = (\frac{1}{m} x x^T + \lambda I)^{-1} (\frac{1}{m} x x^T \beta^* + \frac{1}{m} x \epsilon) \tag{4}$$

In equation (4), m is just a scale factor, therefore we can write this equation as following:

$$\boxed{\hat{\beta} = (x x^T + \lambda I)^{-1} (x x^T \beta^* + x \epsilon)} \tag{5}$$

Now, let's calculate the expected value and variance of this estimate to find the distribution:

$$\begin{aligned} \mathbb{E}[\hat{\beta}] &= \mathbb{E}[(x x^T + \lambda I)^{-1} (x x^T \beta^* + x \epsilon)] \\ &= (x x^T + \lambda I)^{-1} (x x^T \beta^*) \end{aligned} \tag{6}$$

$$\begin{aligned}\mathbb{V}ar[\hat{\beta}] &= \mathbb{E}[(\hat{\beta} - \mathbb{E}(\hat{\beta}))(\hat{\beta} - \mathbb{E}(\hat{\beta}))^T] \\ &= \mathbb{E}[(\hat{\beta} - (xx^T + \lambda I)^{-1}(xx^T\beta^*))(\hat{\beta} - (xx^T + \lambda I)^{-1}(xx^T\beta^*))^T] \\ &= \mathbb{E}[(xx^T + \lambda I)^{-1}x\epsilon\epsilon^T x^T((xx^T + \lambda I)^{-1})^T] \\ &= \sigma^2(xx^T + \lambda I)^{-1}xx^T((xx^T + \lambda I)^{-1})^T\end{aligned} \tag{7}$$

(b) (5 points) Calculate the bias $\mathbb{E}[x^T\hat{\beta}(\lambda)] - x^T\beta^*$ as a function of λ and some fixed test point x.
Answer:

$$\begin{aligned}\mathbb{E}[x^T\hat{\beta}(\lambda)] - x^T\beta^* &= \mathbb{E}[x^T(xx^T + \lambda I)^{-1}(xx^T\beta^* + x\epsilon)] - x^T\beta^* \\ &= x^T(xx^T + \lambda I)^{-1}(xx^T\beta^*) - x^T\beta^* \\ &= x^T[(xx^T + \lambda I)^{-1}(xx^T) - I]\beta^*\end{aligned} \tag{8}$$

(c) (5 points) Calculate the variance term $\mathbb{E}\left[\left(x^T\hat{\beta}(\lambda) - \mathbb{E}[x^T\hat{\beta}(\lambda)]\right)^2\right]$ as a function of λ.
Answer:

$$\begin{aligned}\mathbb{E}[(x^T\hat{\beta} - \mathbb{E}(x^T\hat{\beta}))^2] &= x^T\mathbb{E}[(\hat{\beta} - \mathbb{E}(\hat{\beta}))^2]x \\ &= x^T\mathbb{V}ar[\hat{\beta}]x \\ &= \sigma^2x^T(xx^T + \lambda I)^{-1}xx^T((xx^T + \lambda I)^{-1})^Tx\end{aligned} \tag{9}$$

(d) (5 points) Use the results from parts (b) and (c) and the bias-variance decomposition to analyze the impact of λ in the mean squared error. Specifically, which term dominates when λ is small, and large, respectively?
Answer: The total loss is as following (without considering *noise* term):

$$\begin{aligned}Expected\ Loss &= Bias^2 + Variance \\ &= [x^T[(xx^T + \lambda I)^{-1}(xx^T) - I]\beta^*]^2 + \sigma^2x^T(xx^T + \lambda I)^{-1}xx^T((xx^T + \lambda I)^{-1})^Tx\end{aligned} \tag{10}$$

When λ is large, the bias term becomes large and hence it dominates the loss value. On the contrary, when λ is small, the variance term becomes large and dominates the total loss. Therefore, we should opt cross validation to find the optimal value of λ to minimize the overal loss.

(e) (5 points) Now suppose we have $m = 100$ samples. Write a pseudo-code to explain how to use cross validation to find the optimal λ.
Answer:

Algorithm 1 Cross validation

```
K ← number of folds
Divide samples into K folds
Λ ← [λ_1, λ_2, ..., λ_n]
for λ in Λ do
  for fold in K do
    Leave out fold and train model on rest of the data using λ
    Test model performance on fold (compute loss)
    loss_fold ← loss
  end for
  Loss_λ ← mean(loss)
end for
best λ ← argmin_λ Loss_λ(λ with minimum average loss)
```

(f) (5 points) Explain if we would like to perform variable selection, how should we change the regularization term in Equation (11) to achieve this goal.
Answer: Instead of using ℓ_2-norm, we have to use ℓ_1-norm. This ℓ_1-regularized algorithm (Lasso) can be formulated as following:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \left\{ \frac{1}{m} \sum_{i=1}^{m} (y^i - \beta^T x^i)^2 + \lambda \|\beta\|_1^2 \right\}, \tag{11}$$

Regularizer λ controls model complexity: larger λ restricts the model and less parameters will be selected.

3. **Random forest and one-class SVM for email spam classifier** (40 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 that website. The collection of non-spam emails came from filed work and personal emails, and hence the word 'george' and the area code '650' 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.

(a) (10 points) Build a CART model and visualize the fitted classification tree.
Answer: Figure (3) shows the decision tree which was trained on 80% of the data.

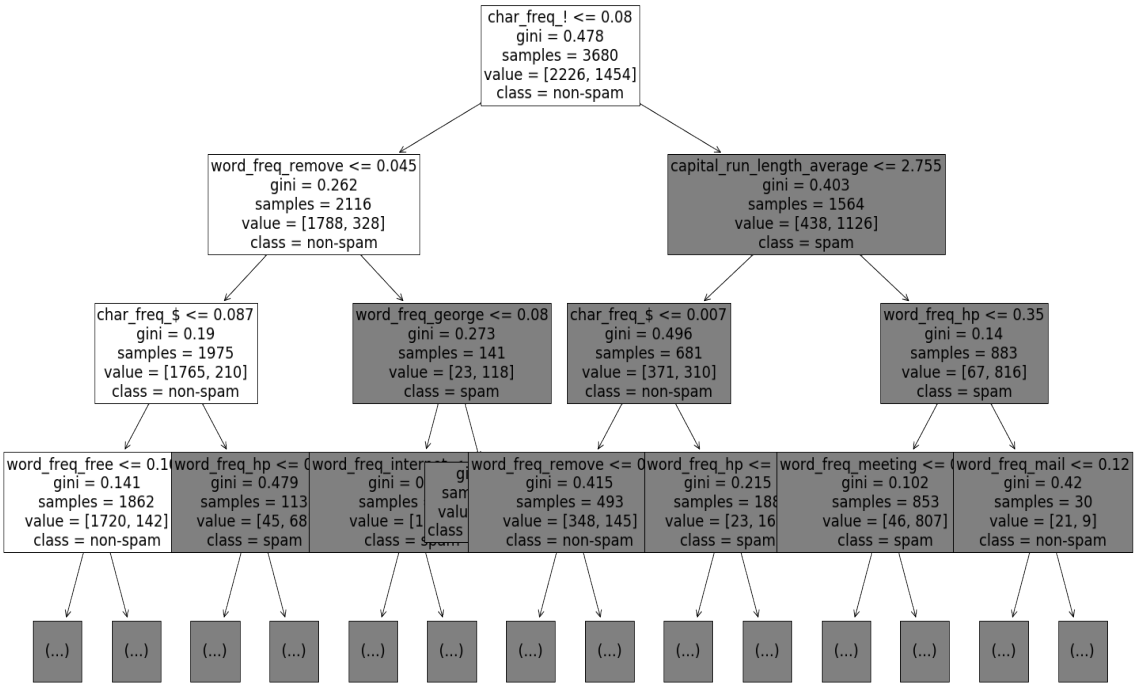


Figure 3: **Decision tree**.

(b) (15 points) Now also build a random forest model. Partition the data to use the first 80% for training and the remaining 20% 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 random forest classifier was trained and with almost 20 trees, the best performance was achieved (Figure 4). The best error rate achieved by random forest classifier was 0.040. However, decision tree classifier was only able to achieve 0.078 error rate.

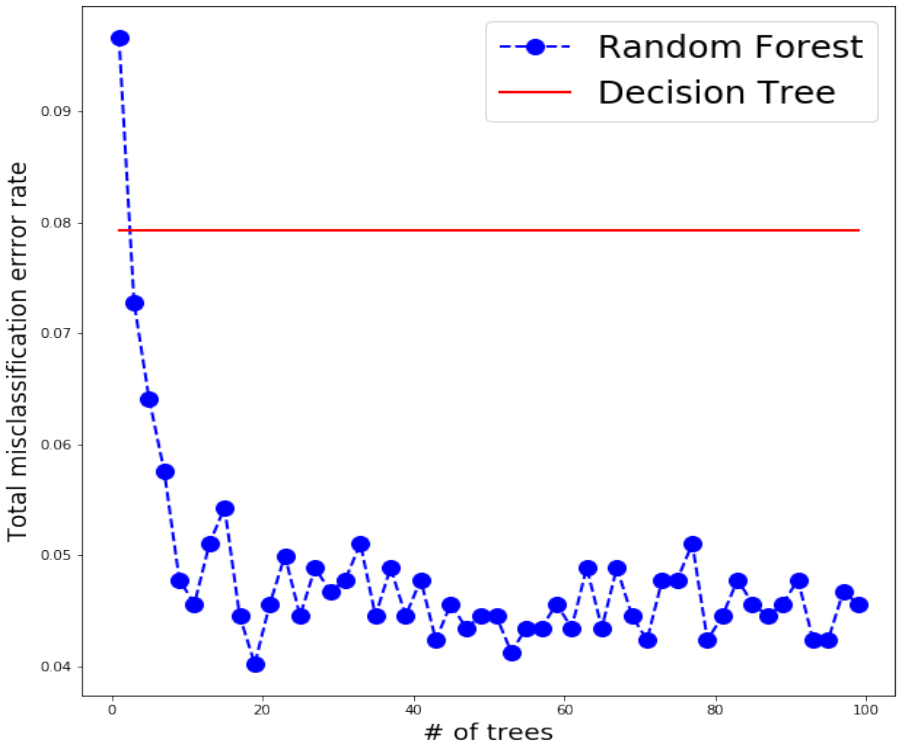


Figure 4: **Error rate as a function of number of estimators (trees) in random forest classifier**.

(c) (15 points) Now we will use a one-class SVM approach for spam filtering. Partition the data to use the first 80% for training and the remaining 20% for testing. Extract all *non-spam* emails from the training block (80% of data you have selected) to build the one-class kernel SVM using RBF kernel (you can turn the kernel bandwidth to achieve good performance). Then apply it on the 20% of data reserved for testing (thus this is a novelty detection situation), and report the total misclassification error rate on these testing data.
Answer: One class SVM could not perform very well on this data. The best misclassification rate achieved at *bandwidth* $= 0.5$ which was equal to 0.420. This is a high error rate possibly because of not providing the negative classes as additional information.