HW6

1. Conceptual Questions

a. Explain how we can control the data-fit complexity for regression trees.
The data-fit complexity for regression trees can be controlled by growing a large tree and stop when a minimum node size has been reached. Then prune the tree by minimizing the cost function:
$C_\alpha(S) = \Sigma_{j=1}^{|J|} \Sigma_{x_i \in R_j} (y_i - \hat{c}_j)^2 + \alpha |J|$

b. What's the main difference between boosting and bagging?
In boosting, weak learners learn sequentially and focus on improving the results of the previous learner. The errors from the previous learner are more heavily weighted than other data in current learner's training set. The learners are combined linearly. In bagging, no prior knowledge from other learners is required. Weak learners are run in parallel on bootstrap replicates of the training set. The learners are combined by taking their average.

c. Explain how OOB errors are constructed and how to use them to decide upon an ideal number of trees in random forest. Is OOB error test or training error and why?
Out of bag errors are constructed when bagging is used to train a random forest. Each tree is trained using a bootstrap sample. Some data is left out to create an out of bag sample. The OOB sample is then used as test data for the trees that do not contain the OOB sample in their training data. The majority vote or average prediction from the group of trees is used to determine the classification. OOB errors is the number of incorrectly predicted rows from the OOB sample. To decide upon an ideal number of trees, we observe when the OOB error stabilizes; we can stop training when OOB error is stable. Since the data used to test the model is different from the data used to train the model, OOB error would be considered a test error.

d. Explain what the "kernel trick" is and why it is used.
The kernel trick allows us to use the inner product of features rather than considering all the features explicitly. The feature space can grow very large, very quickly, can be computationally expensive and may require lots of data to fit all the polynomial terms. By using the kernel trick, we are able to simplify the feature space.

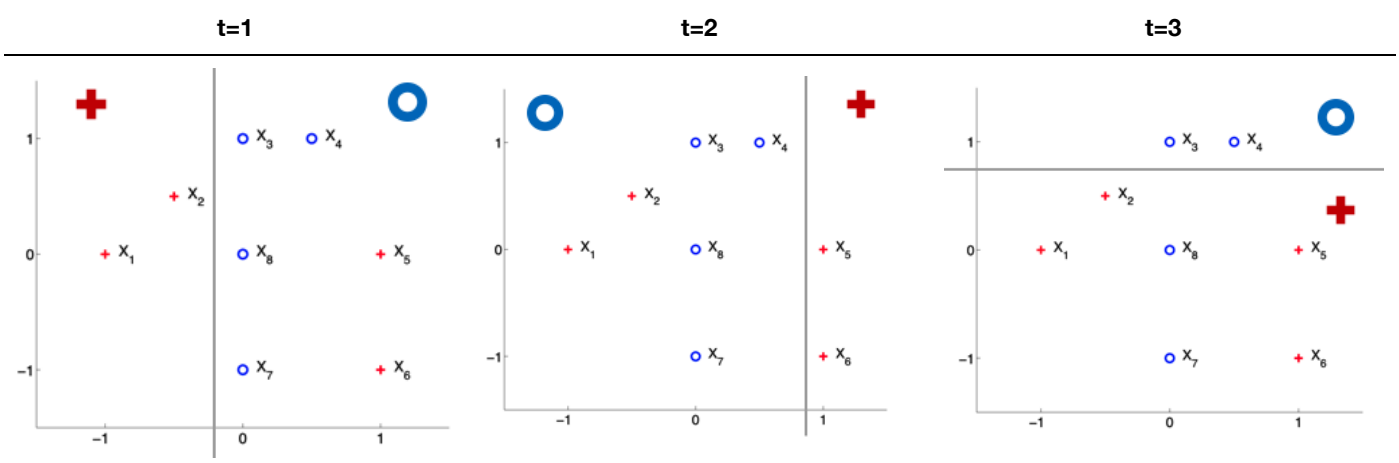
2. AdaBoost

a. For each iteration t=1,2,3 compute $\epsilon_t, \alpha_t, Z_t, D_t$ by hand.
I will show the calculations for the first iteration. Complete table of calculations can be found in *2_AdaBoost.xlsx*.

$D_1(i) = \frac{1}{m} = \frac{1}{8}, i = 1, 2, 3, 4, 5, 6, 7, 8$
$\epsilon_1 = \frac{1}{8}(1 + 1) = \frac{1}{4} \longrightarrow X_5, X_6$ are misclassified
$\alpha_1 = \frac{1}{2} ln(\frac{1-\epsilon_1}{\epsilon_1}) = \frac{1}{2} ln(\frac{1-0.25}{0.25}) \approx 0.5493$
$Z_t = \Sigma_{i=1}^{m} D_t(i) e^{-\alpha_t y^i h_t(x^i)} = \frac{1}{8}[6e^{-0.5493} + 2e^{0.5493}] \approx 0.8660$
$D_2(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t y^i h_t(x^i)} \longrightarrow D_2(1) = D_2(2) = D_2(3) = D_2(4) = D_2(7) = D_2(8) = 0.0833, D_2(5) = D_2(6$

Decision Stumps

t=1	t=2	t=3



Final Classification

$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) = sign(0.25h_1(x) + 0.8047h_2(x) + 1.0986h_3(x))$$
$$H(1) = sign(0.25 * 1 + 0.8047 * -1 + 1.0986 * 1) = 1$$
$$H(2) = sign(0.25 * 1 + 0.8047 * -1 + 1.0986 * 1) = 1$$
$$H(3) = sign(0.25 * -1 + 0.8047 * -1 + 1.0986 * -1) = -1$$
$$H(4) = sign(0.25 * -1 + 0.8047 * -1 + 1.0986 * -1) = -1$$
$$H(5) = sign(0.25 * -1 + 0.8047 * 1 + 1.0986 * 1) = 1$$
$$H(6) = sign(0.25 * -1 + 0.8047 * 1 + 1.0986 * 1) = 1$$
$$H(7) = sign(0.25 * -1 + 0.8047 * -1 + 1.0986 * 1) = -1$$
$$H(8) = sign(0.25 * -1 + 0.8047 * -1 + 1.0986 * 1) = -1$$

t	ϵt	αt	Zt	$Dt(1)$	$Dt(2)$	$Dt(3)$	$Dt(4)$	$Dt(5)$	$Dt(6)$	$Dt(7)$	$Dt(8)$	ht(x1)	ht(x2)	ht(x3)	ht(x4)	ht(x5)	ht(x6)	ht(x7)	ht(x8)
0												1	1	-1	-1	1	1	-1	-1
1	0.2500	0.5493	0.8660	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	1	1	-1	-1	-1	-1	-1	-1
2	0.1667	0.8047	0.7454	0.0833	0.0833	0.0833	0.0833	0.2500	0.2500	0.0833	0.0833	-1	-1	-1	-1	1	1	-1	-1
3	0.1000	1.0986	0.6000	0.2500	0.2500	0.0500	0.0500	0.1500	0.1500	0.0500	0.0500	1	1	-1	-1	1	1	1	1
H(x)												0.8432	0.8432	-2.4526	-2.4526	1.3540	1.3540	-0.2554	-0.2554
sign(H(x))												1	1	-1	-1	1	1	-1	-1

b. What is the training error of this AdaBoost model? Give a short explanation for why AdaBoost outperforms a single decision stump.

Based on the final classification, the training error for this AdaBoost model is 0 - all data points were classified correctly. The average training error for all iterations is 0.1722 and the final training error achieved in t=3 is 0.1.

AdaBoost outperforms a single decision stump because it allows for future weak learners to learn from the weighted dataset from the previous weak learner, thus creating a single strong learner.

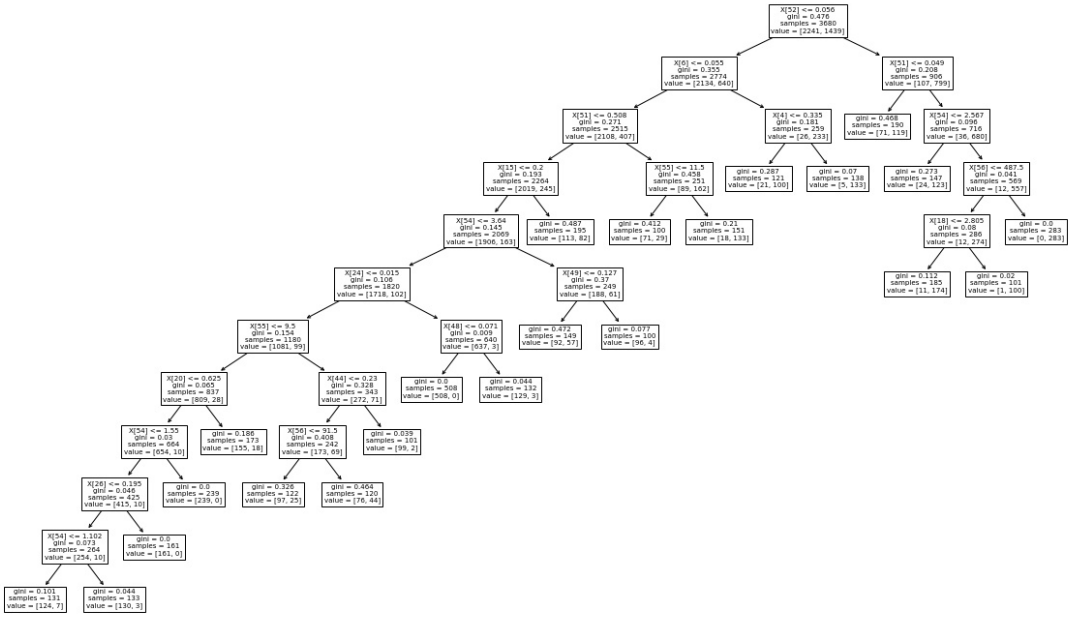
3. Random Forest and One-Class SVM for Email Spam Classifier

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.

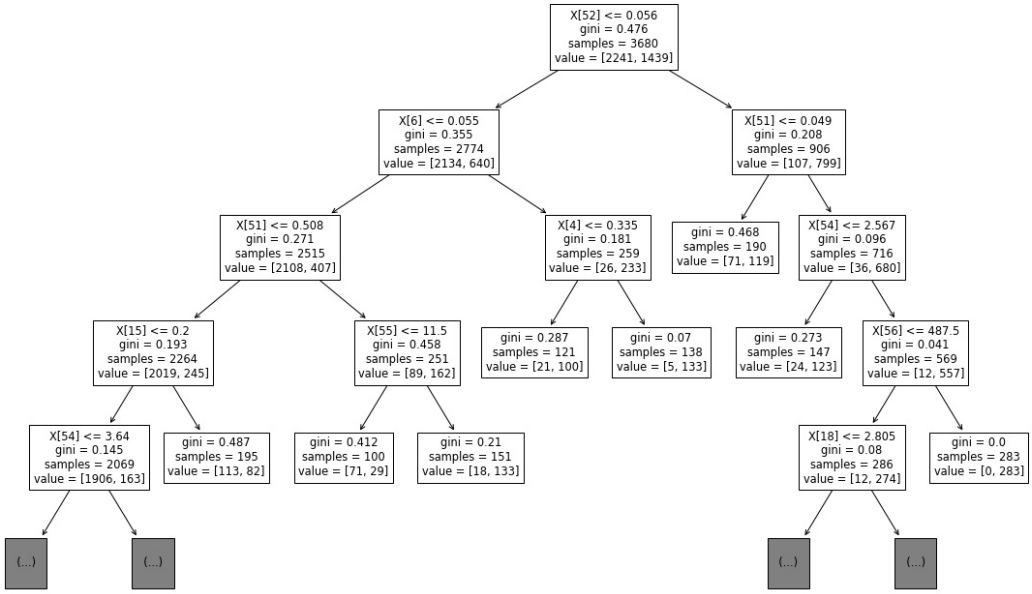
The collection of non-spam emails came from filed work and personal emails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. You are free to choose any package for this homework. Note: there may be some missing values. You can just fill in zero.

a. Randomly shuffle the data and partition to use 80% for training and the remaining 20% for testing. Build a CART model with the training data and visualize the fitted classification.

To create this tree, I configured the classifier to have min_samples_leaf = 100. Without this setting, the tree was very large and difficult to read and understand.



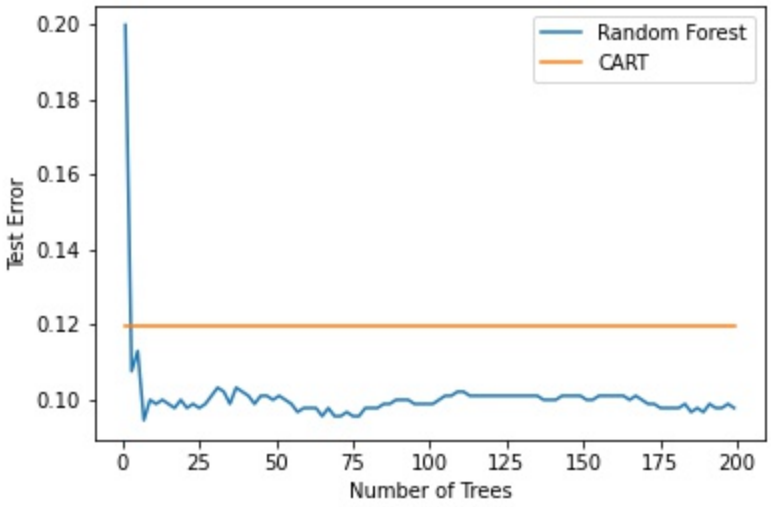
Below is a snapshot of the first 4 levels:



b. Now also build a random forest model. Use your train/test split from part a). 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).

CART Test Error = 0.11943539630836053
Random Forest Test Error = 0.09880564603691644

Test Error vs Number of Trees



c. Now we will use a one-class SVM approach for spam filtering. Use your train/test split from part a). 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 this testing data.

Total Misclassification Error Rate = 0.3669923995656895

4. Locally Weighted Linear Regression and Bias-Variance Tradeoff

Consider a data set with n data points (x_i, y_i) following the following linear model:
$y_i = \beta^{*T} x_u + \epsilon_i$
where $\epsilon_i \sim N(0, \sigma_i^2)$ are independent (but not identically distributed) Gaussian noise with zero mean and variance σ_i^2.

a. Show that the ridge regression which introduces a squared l2 norm penalty on the parameter in the maximum likelihood estimate of β can be written as follows:
$\hat{\beta}(\lambda) = argmin_\beta \{X(\beta - y)^T W X(\beta - y) + \lambda ||\beta||_2^2\}$ **for diagonal matrix W, matrix X and vector y. Please also explain how W, X and y are defined in terms of σ_i^2, x_i and y_i.**

I'm following steps from Prof. Xie's office hour from March 29.

Start with the log likelihood function, given that $y_i \sim N(B^{*T} x_i, \sigma_i^2)$:
$\Sigma_{i=1}^n log(f(y_i|\beta))$
$= \Sigma_{i=1}^n log(\frac{1}{\sqrt{2\pi\sigma_i^2}} e^{\frac{-(y_i - \beta^T x_i)^2}{2\sigma_i^2}})$
$\longrightarrow l(\beta) = \Sigma_{i=1}^n \frac{-1}{2} log(2\pi\sigma_i^2) - \Sigma_{i=1}^n \frac{(y_i - \beta^T x_i)^2}{2\sigma_i^2}$

To get the maximum likelihood estimate of β, we will add the ridge regression penalty and take the derivative of $l(\beta)$ and set to 0. We can drop the first term because it doesn't depend on β.

$max_\beta\ l(\beta) = -\Sigma_{i=1}^n \frac{(y_i - \beta^T x_i)^2}{2\sigma_i^2} + \lambda||\beta||_2^2$

$\longrightarrow min_\beta\ l(\beta) = \Sigma_{i=1}^n \frac{(y_i - \beta^T x_i)^2}{2\sigma_i^2} + \lambda||\beta||_2^2$

Let $y = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}, X = \begin{bmatrix} x_1^T \\ \dots \\ x_n^T \end{bmatrix}$ and $X\beta = \begin{bmatrix} x_1^T\beta \\ \dots \\ x_n^T\beta \end{bmatrix}$ and W = covariance matrix = $\begin{bmatrix} \frac{1}{2\sigma_1^2} \dots 0 \\ \dots \\ 0 \dots \frac{1}{2\sigma_n^2} \end{bmatrix}$

Then $\Sigma_{i=1}^n \frac{(y_i - \beta^T x_i)^2}{2\sigma_i^2} \equiv (y - X\beta)^T W (y - X\beta)$

and the expression $min_\beta\ l(\beta) = \Sigma_{i=1}^n \frac{(y_i - \beta^T x_i)^2}{2\sigma_i^2} + \lambda||\beta||_2^2 \equiv min_\beta\ l(\beta) = (y - X\beta)^T W (y - X\beta) + \lambda||\beta||_2^2$

b. Find the closed-form solution for β(λ) and its distribution conditioning on {x_i}.

To find closed form, we must take the derivative of $l(\beta) = (y - X\beta)^T W (y - X\beta) + \lambda||\beta||_2^2$ and set to 0.

$\frac{\partial}{\partial\beta}(y - X\beta)^T W (y - X\beta) + \lambda||\beta||_2^2 = -2WX^T(y - X\beta) + 2\lambda\beta$

$0 = -2WX^T y + 2WX^T X\beta + 2\lambda\beta$

$2WX^T y = \beta(2WX^T X + 2\lambda)$

$\longrightarrow \hat{\beta} = \frac{WX^T y}{WX^T X + \lambda}$

Finding the distribution:

$E[\hat{\beta}] = E[\frac{WX^T y}{WX^T X+\lambda}] = \frac{WX^T}{WX^T X+\lambda} * E[y] = \frac{WX^T}{WX^T X+\lambda} * (X\beta^*)$

$Var(\hat{\beta}) = Var(\frac{WX^T y}{WX^T X+\lambda}) = \frac{WX^T}{WX^T X+\lambda} * Var(y) = \frac{WX^T}{WX^T X+\lambda}\sigma^2$, where $\sigma^2 = \begin{bmatrix} \sigma_1^2 \\ \dots \\ \sigma_n^2 \end{bmatrix}$

Thus, $\hat{\beta} \sim N(\frac{WX^T X\beta^*}{WX^T X+\lambda}, \frac{WX^T\sigma^2}{WX^T X+\lambda})$.

c. Derive the bias as a function of λ and some fixed test point x.

Bias = $E[\hat{y}] - y = E[X\hat{\beta}] - X\beta^*$

$= E[X\frac{WX^T X\beta^*}{WX^T X+\lambda}] - X\beta^*$

$= X\frac{WX^T X\beta^*}{WX^T X+\lambda} - X\beta^*$

$= \frac{XWX^T X\beta^* - X\beta^*(WX^T X+\lambda)}{WX^T X+\lambda}$

$= \frac{XWX^T X\beta^* - X\beta^* WX^T X - X\beta^*\lambda}{WX^T X+\lambda}$

$= \frac{-X\beta^*\lambda}{WX^T X+\lambda}$

d. Derive the variance term as a function of λ and some fixed text point x.

Variance = $Var(\hat{y}) = Var(x^T\hat{\beta}) = x^T Var(\hat{\beta})x$

$= \frac{x^T WX^T\sigma^2 x}{WX^T X+\lambda}$

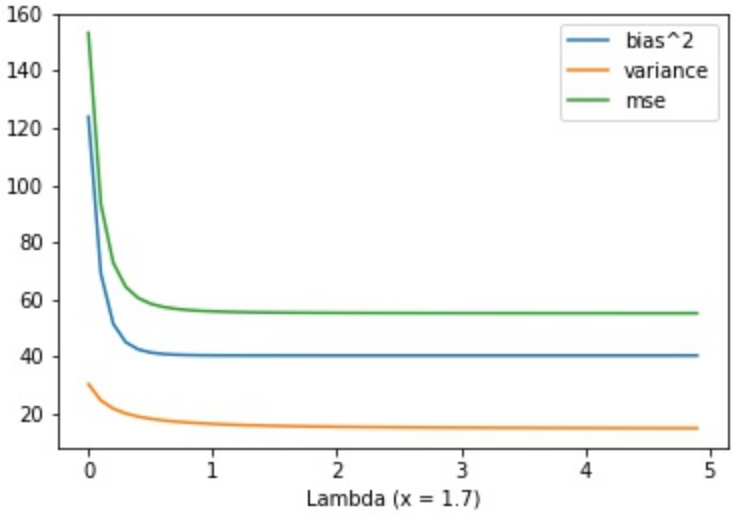
e. Now assuming the data are one-dimensional, we provide a toy training dataset of two samples: x_1 = 1.5 and x_2 = 1, and the test sample x = 1.7. The true parameter β_0^* = 1, β_1^* = 1, the noise variance is given by σ_1^2 = 2, σ_2^2 = 1. Plot the MSE (Bias square plus variance) as a function of the regularization parameter

λ.

$$X = \begin{bmatrix} 1.5 \\ 1 \end{bmatrix},\ \beta^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ W = \begin{bmatrix} \frac{1}{2\sigma_1^2} & 0 \\ 0 & \frac{1}{2\sigma_2^2} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{2} \end{bmatrix},\ \sigma^2 = \begin{bmatrix} \sigma_1^2 \\ \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$MSE = Bias^2 + Variance = (\frac{-X\beta^*\lambda}{WX^TX+\lambda})^2 + \frac{x^TWX^T\sigma^2x}{WX^TX+\lambda}$$

I know that my derivations are not correct because I cannot get a scalar value from my bias and variance calculations. In particular, I cannot get a scalar value for the denominator.
For the sake of this problem and to generate some graphs, I will plot the spectral norm of MSE, bias and variance matrices I get as a result of my calculation.



f. Now change the test sample to be a x = 2.5, and keep everything else the same as in the previous question. Plot the MSE (Bias square plus variance) as a function of the regularization parameter λ, and comment on the difference from the previous result.

As stated in part e, my derivations are incorrect so I cannot properly comment on the difference between x=1.7 and x=2.5.