ISYE 6740 Homework 6
Spring 2025
Prof. Yao Xie
Total 100 points

Provided Data:

- Q3: Random forest and one-class SVM for email spam classifier [spambase.data]
- Q4: Locally weighted linear regression and bias-variance tradeoff [data.mat]

As usual, please submit a report with sufficient explanation of your answers to each the questions, together with your code, in a zip folder.

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

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

   Answer: Bagging builds models in parallel on different bootstrap samples and averages their predictions to reduce variance. Boosting builds models sequentially, where each new model attempts to correct the errors of the previous model, reducing both bias and variance. Random Forest is a bagging method that uses decision trees and aggregates their predictions.

   1.2. List several ways to prevent overfitting in CART.

   Answer:

   - Limit the maximum depth of the tree (max_depth)
   - Use a minimum number of samples required to split a node (min_samples_split)
   - Limit the minimum number of samples per leaf node (min_samples_leaf)
   - Use pruning techniques (pre-pruning or post-pruning)
   - Use cross-validation to tune hyperparameters

   1.3. 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: We control data-fit complexity in a regression tree by limiting the ability of the tree to grow too deep or create overly specific splits. This prevents overfitting and helps the tree generalize better. Common hyperparameters for controlling complexity include:

   - max_depth — maximum depth of the tree
   - min_samples_split — minimum number of samples needed to make a split
   - min_samples_leaf — minimum number of samples required at a leaf node

1.4. 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: OOB (Out-of-Bag) errors are calculated by using the data not included in each bootstrap sample (about 1/3 of the dataset) to evaluate that tree's prediction. By averaging these OOB errors across all trees, we get a reliable estimate of the model's generalization error. As we increase the number of trees, we observe the OOB error — when it stabilizes, it suggests that adding more trees won't significantly improve performance. OOB error is considered more like a test error because it's evaluated on data not used to train the corresponding tree.

1.5. Explain what the bias-variance tradeoff means in the linear regression setting.

Answer: The bias-variance tradeoff reflects the balance between two sources of prediction error:

- Bias: Error from erroneous assumptions in the learning algorithm. In linear regression, a model with too few features (e.g., underfitting) has high bias.
- Variance: Error from sensitivity to small fluctuations in the training set. Including too many predictors or high-degree polynomials increases variance (overfitting).

The goal is to find a model complexity that minimizes the total expected error, which is the sum of bias squared, variance, and irreducible error.

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 mathematically, showing all of your calculation steps. Draw the decision stumps on the figure (you can draw this by hand).

Iteration 1 ($t = 1$)

$$D_1(i) = \frac{1}{m} = \frac{1}{8}, \quad \text{for } i = 1, \ldots, 8$$

$$h_1(x) : x > 0 \Rightarrow +1, \text{ else } -1$$

$$\epsilon_1 = D_1(3) + D_1(4) = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

$$\alpha_1 = \frac{1}{2} \ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \frac{1}{2} \ln\left(\frac{0.75}{0.25}\right) = \frac{1}{2} \ln(3) \approx 0.5493$$

$$Z_1 = \frac{1}{8}\left[6e^{-0.5493} + 2e^{0.5493}\right] \approx 0.8660$$

$$D_2(i) = \frac{D_1(i)}{Z_1} e^{-\alpha_1 y_i h_1(x_i)} \Rightarrow \begin{cases} 0.0833, & \text{correctly classified} \\ 0.1667, & \text{misclassified } (X_3, X_4) \end{cases}$$

Iteration 2 ($t = 2$)

$$h_2(x) : y > 0.25 \Rightarrow +1, \text{ else } -1$$

$$\epsilon_2 = D_2(1) + D_2(2) + D_2(4) + D_2(6) = 0.1667 + 0.1667 + 0.1667 + 0.1000 = 0.6001$$

$$\alpha_2 = \frac{1}{2} \ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) \approx -0.2028$$

$$Z_2 = \sum_{i=1}^{8} D_2(i) \cdot e^{-\alpha_2 y_i h_2(x_i)} \approx 0.9639$$

$$D_3(i) = \frac{D_2(i)}{Z_2} e^{-\alpha_2 y_i h_2(x_i)} \Rightarrow \begin{cases} 0.2273, & X_2 \text{ (most emphasized)} \\ 0.0789, & X_3,\ X_5,\ X_6 \\ \text{others} \approx 0.13 \end{cases}$$

Iteration 3 ($t = 3$)

$$h_3(x) : x > 0.75 \Rightarrow +1, \text{ else } -1$$

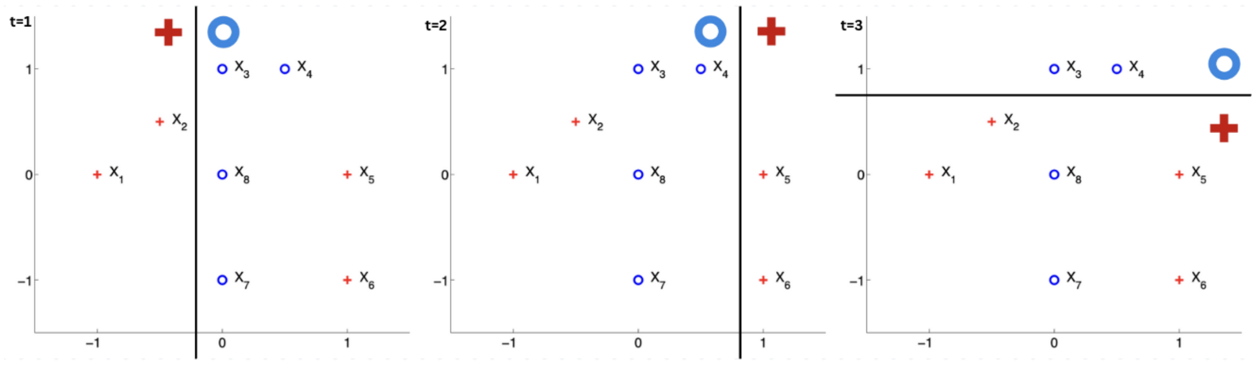
$$\epsilon_3 = D_3(5) + D_3(6) = 0.0789 + 0.0789 = 0.1578$$

$$\alpha_3 = \frac{1}{2} \ln\left(\frac{1-\epsilon_3}{\epsilon_3}\right) \approx 0.8672$$

$$Z_3 = \sum_{i=1}^{8} D_3(i) \cdot e^{-\alpha_3 y_i h_3(x_i)} \approx 0.9594$$

$$D_4(i) = \frac{D_3(i)}{Z_3} e^{-\alpha_3 y_i h_3(x_i)}$$

Decision stumps used at each step are shown in the figure:



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:

AdaBoost correctly classifies all 8 training points using the weighted majority of the three decision stumps, so the training error is:

$$\text{Training error} = \frac{0}{8} = 0\%$$

AdaBoost outperforms a single decision stump because it sequentially focuses on misclassified points, adjusting the distribution of weights to prioritize harder examples. Each weak learner contributes to the final decision based on its accuracy, allowing the ensemble to reduce both bias and variance more effectively than any individual stump.

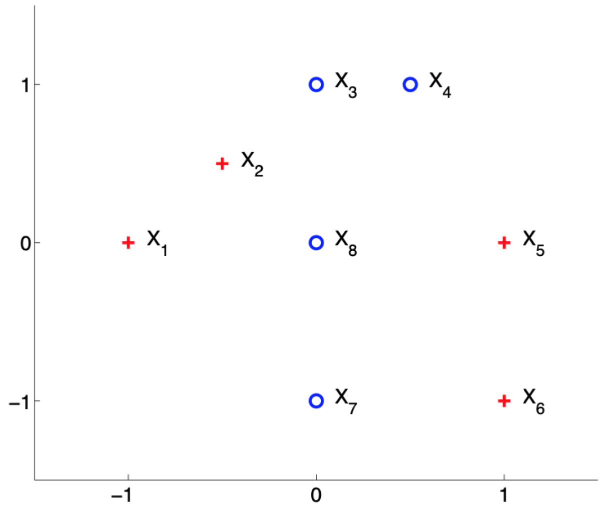


Figure 1: A small dataset for binary classification with AdaBoost.

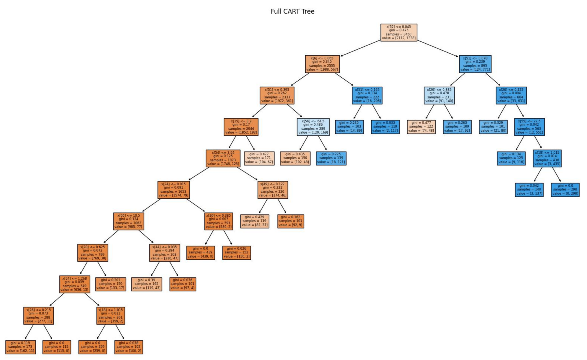
Table 1: Values of AdaBoost parameters at each timestep.

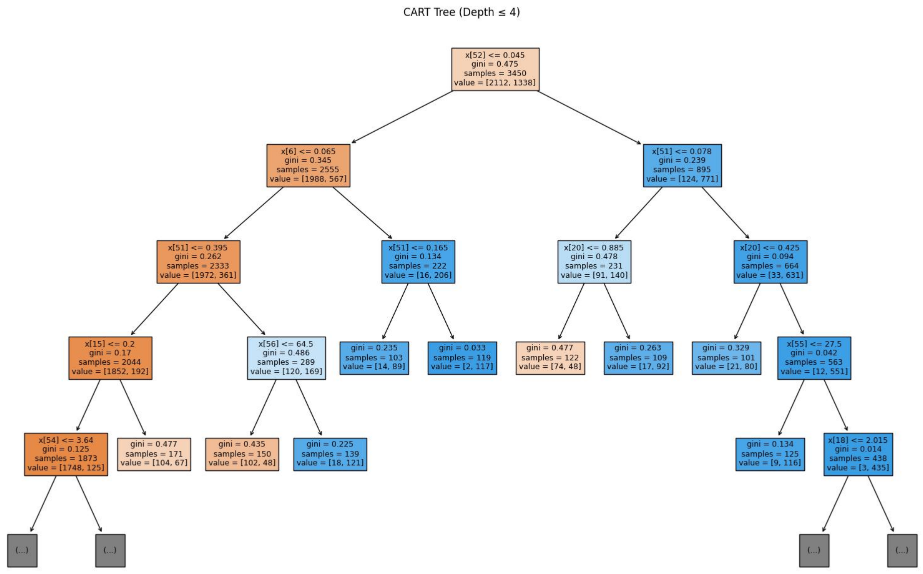
t	ϵ_t	α_t	Z_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$
1	0.3750	0.2554	0.9682	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.6334	-0.2734	0.9639	0.1667	0.1667	0.1000	0.1667	0.1000	0.1000	0.1000	0.1000
3	0.3589	0.2901	0.9594	0.1316	0.2273	0.0789	0.1316	0.0789	0.0789	0.1364	0.1364

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

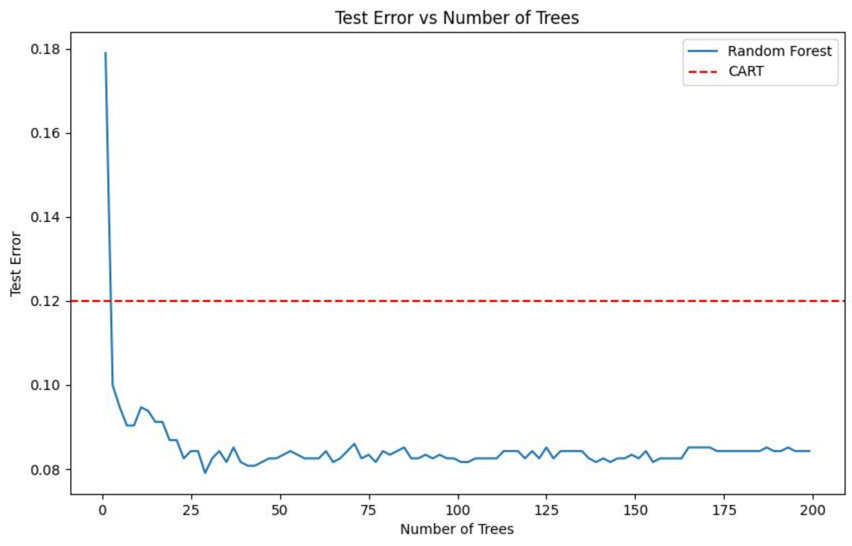
   Your task for this question is to build a spam classifier using the UCR email spam dataset `https://archive.ics.uci.edu/ml/datasets/Spambase` came from the postmaster and individuals who had filed spam. Please use the data provided in the assignment. Headers are not necessary for this problem, but can be obtained by the 'spambase.names' file from the url. 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.

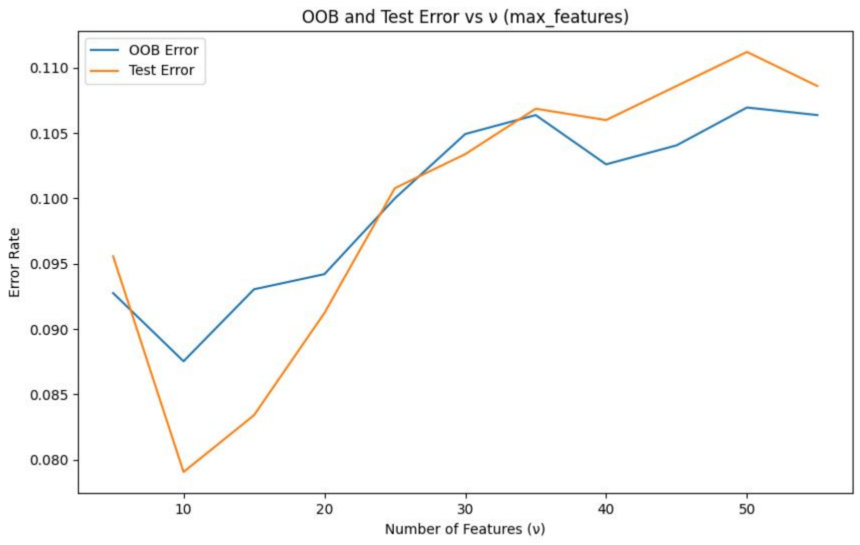



CART Tree (Depth ≤ 4)
x[52] <= 0.045 gini = 0.475 samples = 3450 value = [2112, 1338]
x[6] <= 0.065 gini = 0.345 samples = 2555 value = [1988, 567]
x[51] <= 0.078 gini = 0.239 samples = 895 value = [124, 771]
x[51] <= 0.395 gini = 0.262 samples = 2333 value = [1972, 361]
x[51] <= 0.165 gini = 0.134 samples = 222 value = [16, 206]
x[20] <= 0.885 gini = 0.478 samples = 231 value = [91, 140]
x[20] <= 0.425 gini = 0.094 samples = 664 value = [33, 631]
x[15] <= 0.2 gini = 0.17 samples = 2044 value = [1852, 192]
x[56] <= 64.5 gini = 0.486 samples = 289 value = [120, 169]
gini = 0.235 samples = 103 value = [14, 89]
gini = 0.033 samples = 119 value = [2, 117]
gini = 0.477 samples = 122 value = [74, 48]
gini = 0.263 samples = 109 value = [17, 92]
gini = 0.329 samples = 101 value = [21, 80]
x[55] <= 27.5 gini = 0.042 samples = 563 value = [12, 551]
x[54] <= 3.64 gini = 0.125 samples = 1873 value = [1748, 125]
gini = 0.477 samples = 171 value = [104, 67]
gini = 0.435 samples = 150 value = [102, 48]
gini = 0.225 samples = 139 value = [18, 121]
gini = 0.134 samples = 125 value = [9, 116]
x[18] <= 2.015 gini = 0.014 samples = 438 value = [3, 435]
(...)
(...)
(...)
(...)

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).



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.



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:

To perform spam detection using a one-class SVM, the dataset was first shuffled and partitioned such that 75% was used for training and 25% for testing. Only non-spam emails from the training set were extracted to train the one-class SVM model, treating the task as a novelty detection problem. An RBF kernel was selected for the SVM due to its flexibility in capturing non-linear boundaries.

During evaluation, the trained model was applied to the entire test set, which included both spam and non-spam emails. The SVM predicted whether each test point was similar to the non-spam class it had learned. Points classified as outliers were treated as spam. To optimize the model's performance, the kernel bandwidth (`gamma`) parameter was tuned. After testing several configurations, using `gamma = 'auto'` resulted in the best balance between sensitivity and specificity.

The final misclassification error rate on the test set was:

$$\text{One-Class SVM Test Error: } \boxed{0.3666}$$

This result indicates that the model misclassified approximately 36.66% of the test emails. Given that the model was trained only on non-spam emails without access to any spam examples, this level of error is expected. While the performance is not competitive with supervised classifiers, the one-class SVM approach

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. For the coding portion, Any packages can be used.

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). Your final derivation must define your X, W, and Y vectors/matrices.

Answer:

The objective is to minimize the weighted least squares loss function centered at a point $x \in \mathbb{R}^p$:

$$\widehat{\beta} = \arg\min_{\beta_0, \beta_1} \sum_{i=1}^{n} \left(y_i - \beta_0 - (x - x_i)^T \beta_1 \right)^2 K_h(x - x_i)$$

Let $z_i = x_i - x$ be the shifted features relative to the prediction point. Define:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad \phi_i = \begin{bmatrix} 1 \\ z_i \end{bmatrix} \in \mathbb{R}^{p+1}$$

The loss becomes:

$$\sum_{i=1}^{n} \left(y_i - \phi_i^T \beta \right)^2 K_h(x - x_i)$$

This is a standard weighted least squares problem of the form:

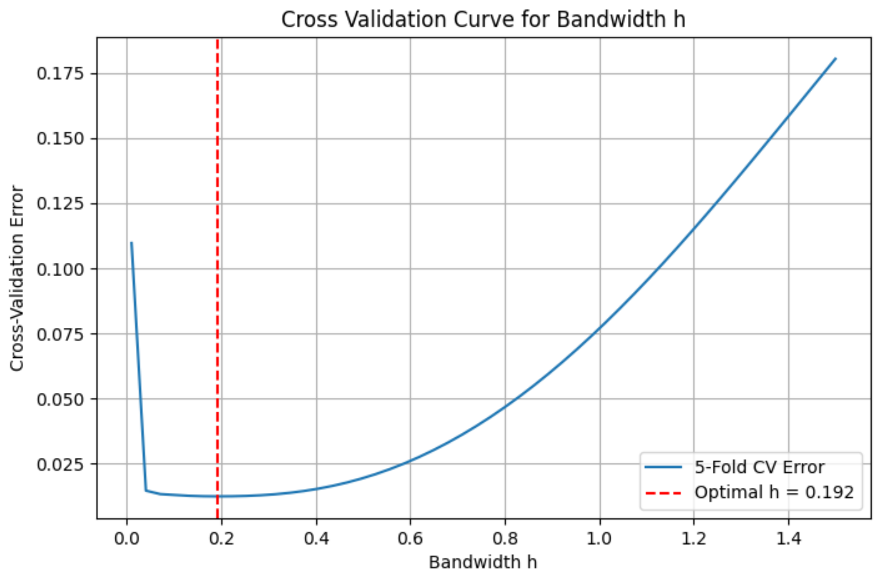
$$\min_{\beta} \sum_{i=1}^{n} w_i \left(y_i - \phi_i^T \beta \right)^2$$

Define the following matrices:
- $X \in \mathbb{R}^{n \times (p+1)}$ is the design matrix, with i-th row $X_i = \phi_i^T = [1 \quad (x_i - x)^T]$ - $Y \in \mathbb{R}^n$ is the response vector: $Y = [y_1, \dots, y_n]^T$ - $W \in \mathbb{R}^{n \times n}$ is the diagonal matrix of weights, where $W_{ii} = K_h(x - x_i)$

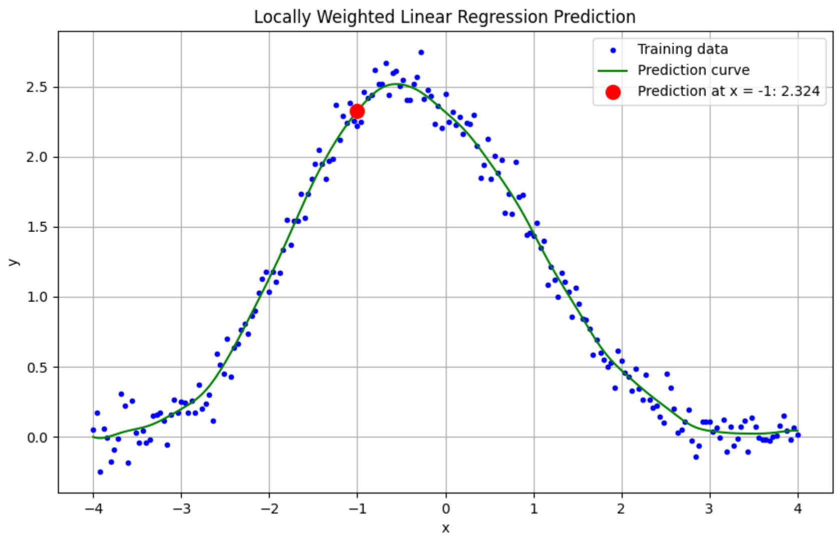
The solution to the weighted least squares problem is then given by:

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

4.2. (5 points) Use all data in the data.mat file (Do not split your data into train/test) 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.



4.3. (5 points) Using the tuned hyper-parameter h to make a prediction for $x = -1$. Provide the predicted y value, and report a plot showing your training data, prediction curve, and a marker indicating your prediction.



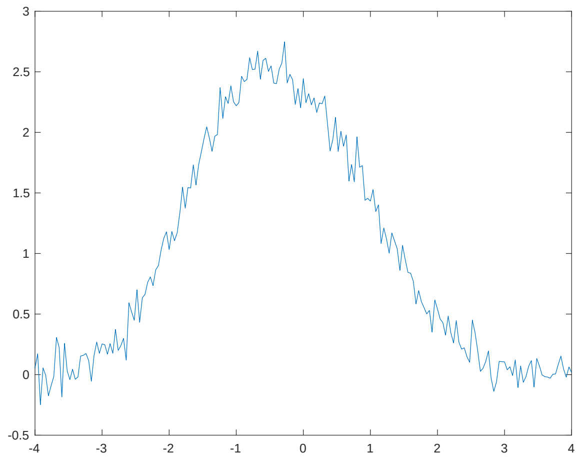


Figure 2: Illustration of noisy curve.