IDS/ACM/CS 158 FUNDAMENTALS OF STATISTICAL LEARNING PROBLEM SET 1

Please submit your solution as a single PDF file, that contains both the written-up and published code parts, via Gradescope by **9pm, Friday, April 24**. An example of the submission process is shown here: `https://www.gradescope.com/get_started#student-submission`

- For theoretical problems, please use a pen, not a pencil: it is hard to read scanned submission written by a pencil.
- For coding problems:
  - Unless otherwise stated, you are not allowed to use high-level built-in functions that implement statistical learning methods, since in many problems you are asked to implement them yourself. If in doubt, ask before using.
  - If you decide to use MATLAB, please convert you scripts to PDF using `publish('your script.m','pdf')`.
  - If you decide to use Python, please use the Jupyter Notebook Template written by the Head TA (available on Piazza, under resources). Only py files without running results logged will not be accepted. If you do not know how to use the Jupyter Notebook, contact the Head TA or attend an office hour.
- After uploading your submission to Gradescope, please label all pages.
- In addition to submitting your full solution via Gradescope, submit a zip-file that contains all your code via a Google form (the link for submission is available on Piazza).

Problem 1. (10 points) K-NN and Linear Regression for Regression.
To get started, in this problem you will implement and test two basic learning methods for solving regression problems: the k-NN regression method and the linear regression method (see Lecture 2 for theoretical details).

(a) (4 points) Write a function `knn_regression`, which implements the k-NN method. It must have three arguments:

1. K, the number of neighbors,
2. D, a matrix that contains the training data. The i^{th} row of D represents the i^{th} observation (x_i^T, y_i), where $x_i \in \mathbb{R}^p$ is the i^{th} training input and $y_i \in \mathbb{R}$ is the corresponding output,
3. X, a column p-vector that represents a new input.

The function must have one output: Y, the prediction of the output from input X.

(b) (4 points) Write a function `linreg_regression`, which implements the linear regression method. It must have two arguments:

1. D, a matrix that contains the training data. The i^{th} row of D represents the i^{th} observation (x_i^T, y_i), where $x_i \in \mathbb{R}^p$ is the i^{th} training input and $y_i \in \mathbb{R}$ is the corresponding output,
2. X, a column p-vector that represents a new input.

The function must have one output: Y, the prediction of the output from input X.

(c) (2 points) Write a script `ps1problem1` that compares the two methods on two different data sets, dataset1 and dataset2, as follows. First, download the training data `dataset1_train.csv` and the test data `dataset1_test.csv` from `https://piazza.com/caltech/spring2020/idsacmcs158/resources`. The files are self-explanatory. The `knn_regression` and `linreg_regression` functions from parts (a) and (b) should get access to the training data through their argument D, to the test inputs through their argument X, and they should not see the test outputs. Use the k-NN method with $K = 5$ neighbors. For each method, compute the *average test error*, which is an empirical estimate of the expected prediction error (under the L_2 loss function) and given by

$$\overline{\text{Err}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2, \tag{1}$$

where n is the sample size of the test data, (x_i^T, y_i) is the i^{th} test observation ($x_i \in \mathbb{R}^p$ is the i^{th} test input and $y_i \in \mathbb{R}$ is the corresponding output), and $\hat{f}(x_i)$ is the prediction of the output from input x_i obtained by the method. Finally, compute the ratio of the test errors for the two methods:

$$R = \overline{\text{Err}}_{knn}/\overline{\text{Err}}_{linreg}. \tag{2}$$

If $R > 1$, then linear regression outperforms the k-NN method, otherwise the k-NN method is more efficient. Repeat the above for the training data `dataset2_train.csv` and the test data `dataset2_test.csv`, which are also available at `https://piazza.com/caltech/spring2020/idsacmcs158/resources`. Compare the two values of R obtained for these two cases (dataset1 and dataset2), and write the results as comments in the script.

Problem 2. (10 POINTS) THE CURSE OF DIMENSIONALITY: THEORY.
In Lecture 2, we discussed that the k-NN method suffers from the curse of dimensionality because training inputs sparsely populate high-dimensional input spaces and, as a consequence, they lie close to the boundary of the training sample (and this is bad because making predictions using k-NN leads to extrapolation from the training inputs instead of interpolation between them). In this problem, you will investigate this "boundary effect" a bit further.

(a) (5 points) Let B_p be the p-dimensional unit ball centered at the origin, and suppose we want to use the k-NN method for predicting the output at its center. Let $X_1, \ldots, X_N$ be N training inputs uniformly distributed in B_p, $X_1, \ldots, X_N \sim \mathcal{U}(B_p)$. Let D be the Euclidean distance from the origin to the nearest neighbor, $D = \min\{\|X_1\|, \ldots, \|X_N\|\}$. Derive an expression for the expected distance $\bar{D} = \mathbb{E}[D]$ and plot $\bar{D}$ versus the dimension p for $N = 10^2, 10^3$, and 10^4 and $p = 1, \ldots, 1000$. The (counter-intuitive) fact that $\bar{D} \to 1$ as $p \to \infty$ is a manifestation of the curse of dimensionality: in high dimensions there are no training inputs in a reasonably small neighborhood of the target input. Please submit your derivation of $\bar{D}$ and the plot of $\bar{D}$ versus p (just the plot, not need to submit the code that generates the plot).

(b) (5 points) The "boundary effect" is a general high-dimensional effect, it is not specific to uniform sampling on bounded domains. Let $X_1, \ldots, X_N \sim \mathcal{N}(0, I_p)$ be N training inputs distributed according to the multivariate normal distribution. Here I_p is the $p \times p$ identity matrix. Let $X \sim \mathcal{N}(0, I_p)$ be a new input where we want to make a prediction. Let $a = X/\|X\|$ be the unit vector in the direction of X and $\text{pr}_a X_i = (a^T X_i)a$ be the projections of training inputs on this direction. Find the expected squared distance from the origin to $\text{pr}_a X_i$ and to X. The (counter-intuitive) fact that $\mathbb{E}[\|\text{pr}_a X_i\|^2] \ll \mathbb{E}[\|X\|^2]$ when p is large is a manifestation of the curse of dimensionality: most training inputs are far from the target input, and they see themselves as lying on the boundary of the training set.

Problem 3. (10 POINTS) THE CURSE OF DIMENSIONALITY: SIMULATION.
The theoretical arguments discussed in Lecture 2 and in Problem 2 explain why the k-NN method suffers from the curse of dimensionality. But it is important to see how this happens in a particular example[1] In this problem, you will see how the performance of the k-NN method gets affected by the curse of dimensionality as the dimension of the input space increases, and how liner regression can suppress this curse provided that the true relationship between the input and output is approximately linear.

Let p be the dimension of the input space. First, generate the training data of the sample size $N = 10^3$ and test data of the same sample size using the following model:

$$\begin{aligned} &X_1, \ldots, X_p \sim \mathcal{N}(0, 1), \\ &Y = X_1 + \ldots + X_p + Z, \quad \text{where } Z \sim \mathcal{N}(0, 1). \end{aligned} \tag{3}$$

[1]"All theory is gray, my friend. But forever green is the tree of life." — J.W. von Goethe, Faust.

Then use your functions knn_regression with $K = 5$ neighbors and linreg_regression that you wrote for Problem 1 to compute the average test error (1) for k-NN and linear regression. Remember that the test data is used only for computing the test error, not for training k-NN and linear regression (the training data is used for that purpose). Repeat this for $p = 1, \dots, 100$ and create two plots:

1. The k-NN average test error versus the dimension p, and
2. The linear regression average test error versus the dimension p. In Lecture 2, we have shown that if $Y = X^T\beta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then the expected mean-squared error at a particular input X is

$$\mathbb{E}[\mathrm{MSE}(X)] \approx \sigma^2 + \frac{\sigma^2}{N}p. \tag{4}$$

In our case (3), $\beta = [1, \dots, 1]^T$ and $\sigma = 1$, and the average test error is an empirical approximation of the expected mean-squared error. So, plot the "theoretical" line (4) on the same figure to see how it approximates the average test error.

Please write a script ps1problem3 that implements these tasks and produces the two required plots.

Problem 4. (10 POINTS) K-NN AND LINEAR REGRESSION FOR CLASSIFICATION.
In this problem, you will implement and test the k-NN classifier and the linear regression method for solving classification problems (see Lecture 3 for theoretical details).

(a) (4 points) Write a function knn_classification, which implements the k-NN classifier. It must have three arguments:

1. K, the number of neighbors,
2. D, a matrix that contains the training data. The i^{th} row of D represents the i^{th} observation (x_i^T, g_i), where $x_i \in \mathbb{R}^p$ is the i^{th} training input and $g_i \in \mathbb{R}$ is the group (class) of x_i,
3. X, a column p-vector that represents a new input.

The function must have one output: G, the prediction of the group of X.

(b) (4 points) Write a function linreg_classification, which implements the linear regression method for classification. You can assume that that there are 2 different groups, $\mathfrak{G} = \{0, 1\}$. The function must have two arguments:

1. D, a matrix that contains the training data. The i^{th} row of D represents the i^{th} observation (x_i^T, g_i), where $x_i \in \mathbb{R}^p$ is the i^{th} training input and $g_i \in \mathbb{R}$ is the group (class) of x_i,
2. X, a column p-vector that represents a new input.

The function must have one output: G, the prediction of the group of X.

(c) (2 points) Write a script ps1problem4 that compares the two classification methods on two different data sets, dataset3 and dataset4, as follows. First, download the training data dataset3_train.csv and the test data dataset3_test.csv from `https://piazza.com/caltech/spring2020/idsacmcs158/resources`. The files are self-explanatory. The `knn_regression` and `linreg_regression` functions from parts (a) and (b) should get access to the training data through their argument$D$, to the test inputs through their argument X, and they should not see the test outputs. Use the k-NN method with $K = 1$ neighbor. For each method, compute the *misclassification rate*, which is an empirical estimate of the expected prediction error (under the zero-one loss function) and given by

$$\bar{R} = \frac{1}{n}\sum_{i=1}^{n} I(g_i \neq \hat{f}(x_i)), \tag{5}$$

where n is the sample size of the test data, (x_i^T, g_i) is the i^{th} test observation ($x_i \in \mathbb{R}^p$ is the i^{th} test input and $g_i \in \mathbb{R}$ is the corresponding group), $\hat{f}(x_i)$ is the prediction of the group of x_i obtained by the method, and I is the indicator function: $I(a \neq b) = 1$ if $a \neq b$ and $I(a \neq b) = 0$ if $a = b$. Repeat the above for the training data dataset4_train.csv and the test data dataset4_test.csv, which are also available at `https://piazza.com/caltech/spring2020/idsacmcs158/resources`. Compare the misclassification rates of k-NN and linear regression obtained for these two cases (dataset3 and dataset4), and write the results as comments in the script.

Remark: Fig. 1 shows the training data for datasets 3 and 4. Before running your script, try to think: what method, k-NN (with $K = 1$) or linear regression, would perform better on each of the two datasets and why? Check you intuition with the numerical results by running your script.

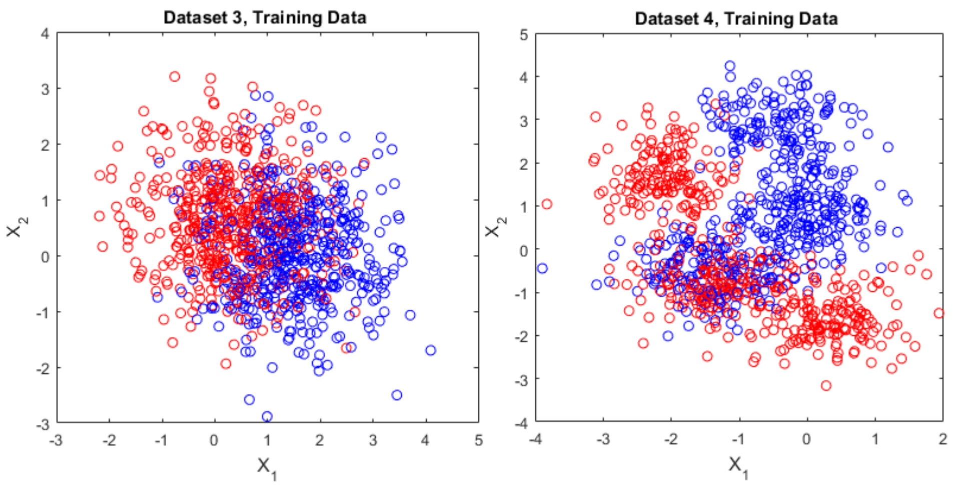

FIGURE 1. The training data for datasets 3 and 4. Red and blue colors indicate inputs from class 0 and class 1.

Problem 5. (10 POINTS) TRAINING VS TESTING ERROR AND THE BIAS-VARIANCE TRADE-OFF. In Lecture 3, among other things, we have discussed two important conceptual points:

- When selecting between different learning methods, we want to choose the method with the smallest *test error*, not the smallest *training error*. In other words, we want the method that performs well on previously unseen data, rather than on the data used to train the method. Small training error does not guarantee that the test error is also small. Moreover, very small training error is an indicator of *overfitting* that leads to large test error.
- The test error at a new input can be decomposed into the sum of three fundamental quantities: *irreducible error* due to the randomness of the output for a given input, the *squared bias* and the *variance* of the predicted output due to the randomness of the training data. While the irreducible error is not under our control, we can control (to some degree) the bias and the variance. More flexible learning methods (roughly speaking, methods with more parameters or methods that can fit many different functional forms) tend to have lower bias but higher variance. Finding the correct level of flexibility is thus crucial for learning from the data.

In this problem, you will observe and explore these points in a particular example via simulation.

(a) (5 points) First, generate the training D_{train} and test data D_{test}, both of size $N = 10^3$, using the following model:

$$\begin{aligned} &X_1, X_2, X_3 \sim N(0,1), \\ &Y = \sin(X_1) + e^{X_2} + \log(|X_3|) + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\sigma^2),\ \sigma = 1. \end{aligned} \tag{6}$$

Using the `knn_regression` function that you wrote in Problem 1, for each $k = 1, \ldots, K = N = 10^3$, compute the average training error:

$$\overline{\text{err}}_k = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_k(x_i))^2, \quad (x_i, y_i) \in D_{\text{train}}, \tag{7}$$

and the average test error:

$$\overline{\text{Err}}_k = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_k(x_i))^2, \quad (x_i, y_i) \in D_{\text{test}}. \tag{8}$$

Here $\hat{f}_k(x)$ is the prediction of the output from input x obtained by the k-NN regression method with k neighbors. Recall that k controls the flexibility of the k-NN method: the larger k is, the less flexible the method. Plot both $\overline{\text{err}}_k$ and $\overline{\text{Err}}_k$ versus N/k (measure of flexibility).

(b) (5 points) In Lecture 3, we have shown that if $Y = f(X) + \epsilon$, where $\mathbb{E}[\epsilon] = 0$ and $\mathbb{V}[\epsilon] = \sigma^2$, then the test error $\text{Err}(X)$ at any input X is

$$\text{Err}(X) = \sigma^2 + (f(X) - \mathbb{E}[\hat{f}(X)])^2 + \mathbb{V}[\hat{f}(X)], \tag{9}$$

where the expectation and the variance are with respect to the training data. Let's observe this bias-variance decomposition in simulation using model (6) and the k-NN regression method. First, generate $T = 10^3$ training data sets $D_1, \dots, D_T$, each of size $N = 10^2$, using model (6), and generate a new input

$$X = (X_1, X_2, X_3)^T, \quad X_1, X_2, X_3 \sim N(0, 1). \tag{10}$$

Using the `knn_regression` function that you wrote in Problem 1, for each $k = 1, \dots, K = N = 10^2$, estimate

1. The test error at X:

$$\text{Err}_k(X) = \mathbb{E}[(Y - \hat{f}_k(X))^2] \approx \frac{1}{T}\sum_{t=1}^{T}(f(X) + \epsilon_t - \hat{f}_k(D_t, X))^2, \tag{11}$$

where $f(X) = \sin(X_1) + e^{X_2} + \log(|X_3|)$, $\epsilon_t \sim \mathcal{N}(0, 1)$, and $\hat{f}_k(D_t, X)$ is the prediction of the output from input X obtained by the k-NN method with k neighbors and training data D_t.

2. The squared bias of $\hat{f}_k(X)$:

$$\mathbb{B}[\hat{f}_k(X)]^2 = (f(X) - \mathbb{E}[\hat{f}_k(X)])^2 \approx \left(f(X) - \frac{1}{T}\sum_{t=1}^{T}\hat{f}_k(D_t, X)\right)^2. \tag{12}$$

3. The variance of $\hat{f}_k(X)$:

$$\mathbb{V}[\hat{f}_k(X)] \approx \frac{1}{T}\sum_{t=1}^{T}\left(\hat{f}_k(D_t, X) - \frac{1}{T}\sum_{s=1}^{T}\hat{f}_k(D_s, X)\right)^2. \tag{13}$$

Plot the estimated $\text{Err}_k(X)$, $\mathbb{B}[\hat{f}_k(X)]^2$, $\mathbb{V}[\hat{f}_k(X)]$, and $\sigma^2 + \mathbb{B}[\hat{f}_k(X)]^2 + \mathbb{V}[\hat{f}_k(X)]$ versus N/k on the same figure. For different inputs X and different training sets $D_1, \dots, D_T$, these four curves my look differently quantitatively, but qualitatively, the test error should have U-shape, the bias should decrease and the variance should increase with N/k, and, thanks to (9), the first curve should approximately follow the forth curve, $\text{Err}_k(X) \approx \sigma^2 + \mathbb{B}[\hat{f}_k(X)]^2 + \mathbb{V}[\hat{f}_k(X)]$.

Please write a script `ps1problem5` that implements these tasks and produces the two required plots.