HOMEWORK 3

Venkata Saikiranpatnaik Balivada
Net ID: vbalivada
Wisc ID: 9084550244

October 10, 2023

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Late submissions may not be accepted. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework.
Code and plots are available in the github repo https://github.com/pao214/cs760_hw3.

1 Questions (50 pts)

1. (9 pts) Explain whether each scenario is a classification or regression problem. And, provide the number of data points (n) and the number of features (p).

   (a) (3 pts) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in predicting CEO salary with given factors.
   This is a $\boxed{regression}$ problem since salary is a continuous value. We have 500 data points one from each firm. So, $\boxed{n = 500}$. And apart from the true label (CEO salary), we have 3 features, so $\boxed{p = 3}$.

   (b) (3 pts) We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.
   This is a $\boxed{classification}$ problem since success or failure is discrete. We have 20 data points, one for each product, so $\boxed{n = 20}$. Each data point comes with $\boxed{p = 13}$ features apart from the true label.

   (c) (3 pts) We are interesting in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.
   This is a $\boxed{regression}$ problem since percentage change is continuous. Assuming, there are 52 weeks in a year, we have $\boxed{n = 52}$ data points, one for each week. Moreover, since we consider three markets, $\boxed{p = 3}$.

2. (6 pts) The table below provides a training data set containing six observations, three predictors, and one qualitative response variable.

X_1	X_2	X_3	Y
0	3	0	Red
2	0	0	Red
0	1	3	Red
0	1	2	Green
-1	0	1	Green
1	1	1	Red

Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K-nearest neighbors.

(a) (2 pts) Compute the Euclidean distance between each observation and the test point, $X_1 = X_2 = X_3 = 0$.

Extended the table in question with another column containing Euclidean distance from $(0, 0, 0)$ in Table 1.

X_1	X_2	X_3	Y	Distance
0	3	0	Red	3
2	0	0	Red	2
0	1	3	Red	3.16
0	1	2	Green	2.24
-1	0	1	Green	1.41
1	1	1	Red	1.73

Table 1: Euclidean distance from (0, 0, 0)

(b) (2 pts) What is our prediction with $K = 1$? Why?
This is simply the label of the closest point. The closest point is $(-1, 0, 1)$ with label $\boxed{Green}$.

(c) (2 pts) What is our prediction with $K = 3$? Why?
Here, we take the majority label of the closest 3 points, namely, $(-1, 0, 1), (1, 1, 1), (2, 0, 0)$ which is Green, Red, and Red respectively. Clearly, the prediction is $\boxed{Red}$.

3. (12 pts) When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large.

(a) (2pts) Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?
The algorithm always looks at 10% of the interval irrespective of whether or not we are the boundary. So, we are picking 10% of the total points in expectation, i.e. the fraction is $\boxed{0.1}$.

(b) (2pts) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, $X1$ and $X2$. We assume that predict a test observation's response using only observations that $(X1, X2)$ are uniformly distributed on [0, 1] × [0, 1]. We wish to are within 10% of the range of $X1$ and within 10% of the range of $X2$ closest to that test observation. For instance, in order to predict the response for a test observation with $X1 = 0.6$ and $X2 = 0.35$, we will use observations in the range [0.55, 0.65] for $X1$ and in the range [0.3, 0.4] for $X2$. On average, what fraction of the available observations will we use to make the prediction?
Say, our test point is $(x1, x2)$. We pick all points whose $x1 \in [a1, b1]$ and $x2 \in [a2, b2]$ where both intervals have length 0.1 (by question description). The probability that any random point will be used for kNN is $P(a1 \leq x1 \leq b1, a2 \leq x2 \leq b2) = P(a1 \leq x \leq b1)P(a2 \leq x \leq b2)$ (assuming independence). Geometrically, this is the ratio of area used for classification to the total area. Each probability is 0.1 and hence the fraction is $0.1 \times 0.1 = \boxed{10^{-2}}$

(c) (2pts) Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?
Generalizing the above argument, the fraction can be computed as $P(a1 \leq x1 \leq b1, ..., a100 \leq x100 \leq b100) = \Pi_{i=1}^{100} P(ai \leq xi \leq bi) = 0.1^{100} = \boxed{10^{-100}}$

(d) (3pts) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

As we can see, the fraction of points we can pick decreases exponentially with p when we consider a fixed fraction along each feature. Exponential reduction is quite fast and it is nearly improbable that we find a point when the probability is as low as 10^{-100}, for example. Geometrically, even if one feature of a point is far enough away, we are too far away and this is a very likely possibility with lots of features.

(e) (3pts) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p =1, 2, and 100, what is the length of each side of the hypercube? Comment what happens to the length of the sides as $\lim_{p\to\infty}$.

Say, the length of each side of the hypercube is l. Using the above generalization, the fraction becomes l^p. We want this to be equal to $10\% = 0.1$. This means $l^p = 0.1 \implies l = 0.1^{\frac{1}{p}}$. As $p \to \infty$, $\frac{1}{p} \to 0$, and hence $0.1^{\frac{1}{p}} \to 0.1^0 = \boxed{1}$. Hence, the length of the hypercube spans the whole feature range.

4. (6 pts) Supoose you trained a classifier for a spam detection system. The prediction result on the test set is summarized in the following table.

		Predicted class	
		Spam	not Spam
Actual class	Spam	8	2
	not Spam	16	974

Calculate

(a) (2 pts) Accuracy Here, we assume that Spam is positive and non-spam is negative. So, there are 8 true positives, 16 false positives, 2 false negatives, and 974 true negatives. The accuracy is $\frac{TP+TN}{TP+FP+FN+TN} = \frac{8+974}{8+16+2+974} = \boxed{0.98}$.

(b) (2 pts) Precision Precision is $\frac{TP}{TP+FP} = \frac{8}{8+16} = \boxed{0.33}$.

(c) (2 pts) Recall Recall is $\frac{TP}{TP+FN} = \frac{8}{8+2} = \boxed{0.8}$.

5. (9pts) Again, suppose you trained a classifier for a spam filter. The prediction result on the test set is summarized in the following table. Here, "+" represents spam, and "-" means not spam.

Confidence positive	Correct class
0.95	+
0.85	+
0.8	-
0.7	+
0.55	+
0.45	-
0.4	+
0.3	+
0.2	-
0.1	-

(a) (6pts) Draw a ROC curve based on the above table.

See Figure 1 for the ROC curve. See the ipynb file in the github repo for the code generating this plot.

(b) (3pts) (Real-world open question) Suppose you want to choose a threshold parameter so that mails with confidence positives above the threshold can be classified as spam. Which value will you choose? Justify your answer based on the ROC curve.

The figure is similar to a step function where FPR remains constant for a period while TPR increases and then FPR increases after we reach a mini-peak. Clearly, we want to be picking one of these peaks since otherwise we can increase TPR without increasing FPR. We have three such peaks. The peak on the right has a very high TPR, but also has a reasonably high FPR, which means more than $\frac{1}{2}$ of

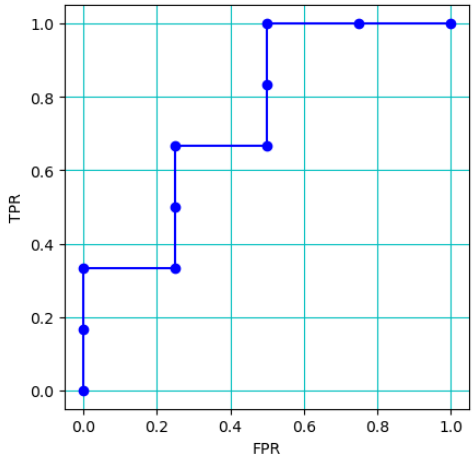

Figure 1: ROC figure for the provided data of a spam filter.

non-spam emails are being classified as spam (not ideal). The lowest peak has a very low FPR, but also has too low of a TPR for my liking. The inbox will be full of spam. There is no right answer here, but I am comfortable taking the middle peak where the TPR is close to 70% while the FPR is below 30%. On the other hand, if we have reason to believe that the users are tech-savvy enough to be able to do spam classification on their own, the left peak is ideal since you miss no non-spam emails.

6. (8 pts) In this problem, we will walk through a single step of the gradient descent algorithm for logistic regression. As a reminder,

$$\hat{y} = f(x, \theta)$$
$$f(x; \theta) = \sigma(\theta^\top x)$$
$$\text{Cross entropy loss } L(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$
$$\text{The single update step } \theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x; \theta), y)$$

(a) (4 pts) Compute the first gradient $\nabla_\theta L(f(x; \theta), y)$.

Before computing the gradient, let us substitute the functional forms into the loss function to get

$$\begin{aligned}
&-(y \log(\frac{1}{1 + \exp(-\theta^T x)}) + (1 - y) \log(1 - \frac{1}{1 + \exp(-\theta^T x)})) \\
&= y \log(1 + \exp(-\theta^T x)) - (1 - y) \log(\frac{\exp(-\theta^T x)}{1 + \exp(-\theta^T x)}) \\
&= y \log(1 + \exp(-\theta^T x)) - (1 - y) \log(\frac{1}{1 + \exp(\theta^T x)}) \\
&= y \log(1 + \exp(-\theta^T x)) + (1 - y) \log(1 + \exp(\theta^T x))
\end{aligned}$$

We now differentiate this expression with θ.

$$
\begin{aligned}
\nabla_\theta L &= \nabla_\theta(y\log(1+\exp(-\theta^T x)) + (1-y)\log(1+\exp(\theta^T x))) \\
&= \nabla_\theta(y\log(1+\exp(-\theta^T x))) + \nabla_\theta((1-y)\log(1+\exp(\theta^T x))) \\
&= y\frac{1}{1+\exp(-\theta^T x)}\nabla_\theta(1+\exp(-\theta^T x)) + (1-y)\frac{1}{1+\exp(\theta^T x)}\nabla_\theta(1+\exp(\theta^T x)) \\
&= -x^T y\frac{\exp(-\theta^T x)}{1+\exp(-\theta^T x)} + x^T(1-y)\frac{\exp(\theta^T x)}{1+\exp(\theta^T x)} \\
&= -x^T(y(1-\frac{1}{1+\exp(-\theta^T x)}) + (1-y)\frac{1}{1+\exp(-\theta^T x)}) \\
&= x^T(-y(1-f)+(1-y)f) \\
&= \boxed{x^T(f-y)}
\end{aligned}
$$

(b) (4 pts) Now assume a two dimensional input. After including a bias parameter for the first dimension, we will have $\theta \in \mathbb{R}^3$.

$$\text{Initial parameters} : \theta^0 = [0,0,0]$$

$$\text{Learning rate } \eta = 0.1$$

$$\text{data example} : x = [1,3,2], y = 1$$

Compute the updated parameter vector θ^1 from the single update step.

Plugging in the values, our function value is 0.5, gradient is $[-0.5,-1.5,-1]$. After the gradient update with a learning rate of 0.1, the parameters now become $\boxed{[0.05, 0.15, 0.1]}$.

2 Programming (50 pts)

1. (10 pts) Use the whole D2z.txt as training set. Use Euclidean distance (i.e. $A = I$). Visualize the predictions of 1NN on a 2D grid $[-2 : 0.1 : 2]^2$. That is, you should produce test points whose first feature goes over $-2, -1.9, -1.8, \ldots, 1.9, 2$, so does the second feature independent of the first feature. You should overlay the training set in the plot, just make sure we can tell which points are training, which are grid.

   The expected figure looks like this.

   The figure 2 shows grid points, labeled positive and negative.

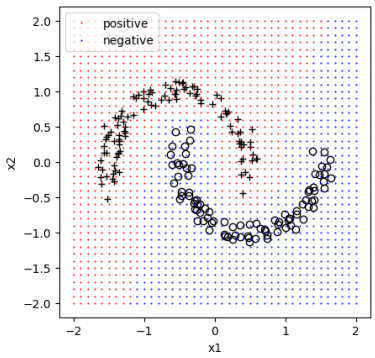

Figure 2: 1NN on a grid of data

Spam filter Now, we will use 'emails.csv' as our dataset. The description is as follows.

- Task: spam detection
- The number of rows: 5000

- The number of features: 3000 (Word frequency in each email)
- The label (y) column name: 'Predictor'
- For a single training/test set split, use Email 1-4000 as the training set, Email 4001-5000 as the test set.
- For 5-fold cross validation, split dataset in the following way.
  - Fold 1, test set: Email 1-1000, training set: the rest (Email 1001-5000)
  - Fold 2, test set: Email 1000-2000, training set: the rest
  - Fold 3, test set: Email 2000-3000, training set: the rest
  - Fold 4, test set: Email 3000-4000, training set: the rest
  - Fold 5, test set: Email 4000-5000, training set: the rest

2. (8 pts) Implement 1NN, Run 5-fold cross validation. Report accuracy, precision, and recall in each fold.

   See table 2.

Fold	Accuracy	Precision	Recall
1	0.82	0.65	0.82
2	0.85	0.69	0.87
3	0.86	0.72	0.84
4	0.85	0.72	0.82
5	0.78	0.61	0.76

Table 2: Accuracy, Precision, and Recall for all 5 folds: 1NN

3. (12 pts) Implement logistic regression (from scratch). Use gradient descent (refer to question 6 from part 1) to find the optimal parameters. You may need to tune your learning rate to find a good optimum. Run 5-fold cross validation. Report accuracy, precision, and recall in each fold.

   See table 3 for results. Weights are initialized using random numbers from a normal distribution, the learning rate is 0.005 and run for 5000 epochs.

Fold	Accuracy	Precision	Recall
1	0.94	0.90	0.89
2	0.94	0.91	0.86
3	0.92	0.92	0.80
4	0.93	0.89	0.85
5	0.91	0.85	0.84

Table 3: Accuracy, Precision, and Recall for all 5 folds: Logistic Regression

4. (10 pts) Run 5-fold cross validation with kNN varying k (k=1, 3, 5, 7, 10). Plot the average accuracy versus k, and list the average accuracy of each case.
   Expected figure looks like this.

   See figure 3 for the plot and values in table 4.

k	Accuracy
1	0.8336
3	0.8416
5	0.8414
7	0.846
10	0.8548

Table 4: Cross validation Accuracy for different values of k in kNN

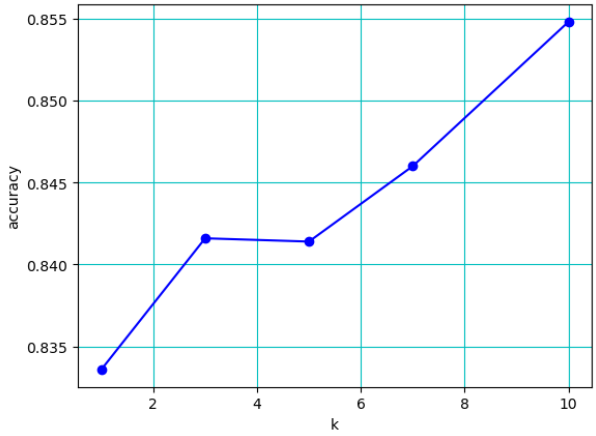

Figure 3: k vs Accuracy in kNN 5-fold cross validation

5. (10 pts) Use a single training/test setting. Train kNN (k=5) and logistic regression on the training set, and draw ROC curves based on the test set.
Expected figure looks like this. Note that the logistic regression results may differ.

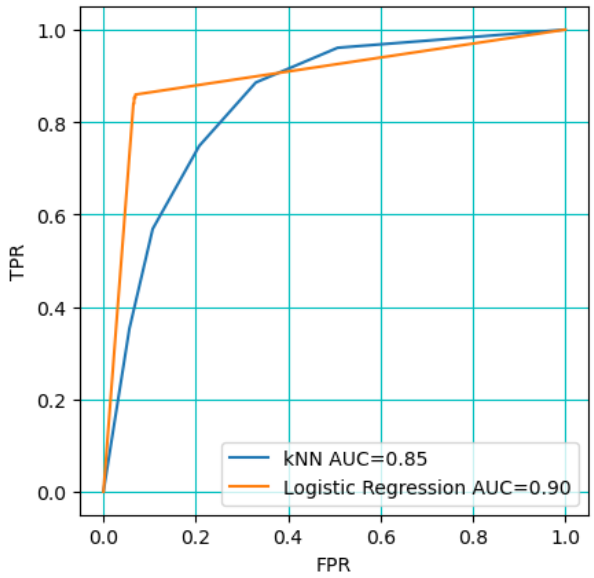

Figure 4: ROC curve comparing kNN with k = 5 and Logistic Regression

See figure 4 for the ROC curve plot for both kNN and Logistic Regression.