HOMEWORK 3

>>Hemanth Sridhar Nakshatri<<
>>nakshatri@wisc.edu<<

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.

The HW can be found in the following link: https://github.com/hemanth-nakshatri/ECE760-Machine-Learning/tree/main/hw/hw3. In case this doesn't work, all the HWs can be found in this github repo: https://github.com/hemanth-nakshatri/ECE760-Machine-Learning

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.

   - **Problem type**: Regression
   - **Reason**: CEO salary is a conntinuos variable. (*it could have been classification if we were categorizing something like high or low salary etc*)
   - ***n*** = 500
   - ***p*** = 3 (profit, no. of employees, industry)

   (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.

   - **Problem type**: Classification
   - **Reason**: We are predicting success or failure (Boolean/categorical variable)
   - ***n*** = 20
   - ***p*** = 13 (price, marketing budget, competetion price, and ten more)

   (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.

   - **Problem type**: Regression
   - **Reason**: We are predicting percentage change in US dollar. This is a continuos variable.
   - ***n*** = 52 (No. of weeks in 2012)
   - ***p*** = 3 (% changes in US market, British market and the German market)

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, $X_1 = X_2 = X_3 = 0$.

Test point = (0,0,0)

Euclidean distance is given by $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$

Distances of each point from (0,0,0) are as follows,

- 1: (0,3,0) = 3.0
- 2: (2,0,0) = 2.0
- 3: (0,1,3) = 3.1623
- 4: (0,1,2) = 2.2361
- 5: (-1,0,1) = 1.4142
- 6: (1,1,1) = 1.732

(b) (2 pts) What is our prediction with $K = 1$? Why?

For $K = 1$, the prediction is **Green** since the nearest point (-1,0,1) belongs to class Green.

(c) (2 pts) What is our prediction with $K = 3$? Why?

For $K = 3$, we consider 3 of the nearest points i.e (-1,0,1), (1,1,1) and (2,0,0). We consider the majority class as the final prediction.

In our case there are 2 Reds and 1 Green. Thus the prediction for $K = 3$ is **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?

For one dimension, **10%** of the available observations will be used for predictions. (Length of line = 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?

For two dimensions, we use 10% for each dimension. Thus we use **0.1 * 0.1 = 0.01 or 1%** of the total observations. (Area of square = 0.01)

(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?

For each dimension, we use **0.1 * 0.1 * ..... 0.1** fraction of features.
$\implies (\mathbf{0.1})^{\mathbf{100}}$.

(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 increase the number of features (p), the fraction of available observations used for prediction reduces drastically. In high-dimensional spaces, observations are far away from test observations. Thus KNN performance deteriorates in high-dimensional spaces due to low number of "near" observations.

(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 $\lim_{n\to\infty}$.

Ans: length of side as $p \to \infty = 1$

For p = *1* (a line), to capture 10% of the data, we need 10% of the line's length i.e 0.1.
For p = *2* (a square), we need to capture 10% of the data in a square of side s and area s^2.
$s^2 = 10\%\ of\ the\ unit\ square.$
Thus, $s = \sqrt{10}$.
Similarly, extending this to $p \to \infty$, we have a hypercube whose volume is 10% of unit square. Thus each side will be $(\mathbf{0.1})^{\mathbf{1/\infty}} \implies (\mathbf{1})$

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
$Accuracy = (TP + TN)/(TP + TN + FP + FN)$
$Accuracy = \frac{(8+974)}{8+974+2+16}$

Accuracy = 0.982 or 98.2%

(b) (2 pts) Precision
$Precision = TP/(TP + FP)$
$Precsion = 8/(8 + 2)$
Precision = 0.8 or 80%

(c) (2 pts) Recall
$Recall = TP/(TP + FN)$
$Recall = 8/(8 + 16)$
Recall = 0.3333 or 33.33%

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.

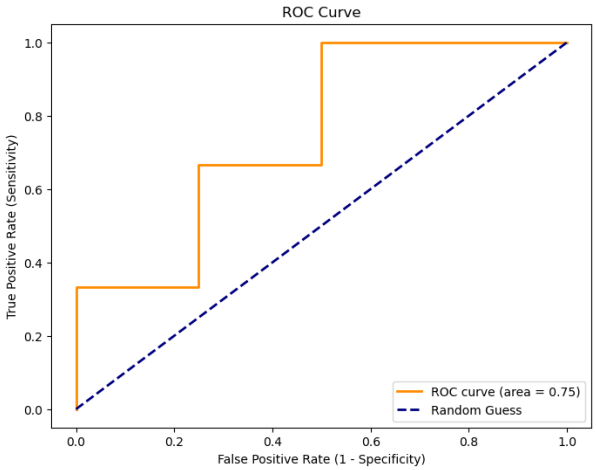

Figure 1: ROC curve for Q5

See Fig. 1 for the ROC 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 threshold can be set as **0.55** above which the mails are classified as spam. From Fig. 2, it can be seen that True positive rate increases significantly with negligible increase in False positive rate. Further, the true positive rate reaches the maximum at that threshold.

Another point of view would be to select threshold at **0.3** since you would get good true positive rate with minimal false positive rate and there's a stagnation of true positive rate till threshold = 0.55.

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 $\nabla_\theta L(f(x; \theta), y)$.

Logisitic regression model: $\hat{y} = \sigma(\theta^T x)$

Derviative of loss wrt to $\hat{y}$,

$\frac{\partial L}{\partial \hat{y}} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}})$

Derivative of sigmoid function,
$\sigma(z) = 1/(1 + e^{-z})$
$\sigma'(z) = \sigma(z) * (1 - \sigma(z))$

Derivative of $\hat{\mathbf{y}}$ wrt θ
$\frac{\partial \hat{y}}{\partial \theta} = \sigma(\theta^T x)(1 - \sigma(\theta^T x)) \cdot x$

Thus, $\nabla_\theta(f(x;\theta), y)$ is given by,
$\nabla_\theta(f(x;\theta), y) = \frac{\partial L}{\partial \theta} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}})(\sigma(\theta^T x)(1 - \sigma(\theta^T x)) \cdot x)$
Simplifies to,
$\nabla_\theta(\mathbf{f}(\mathbf{x};\theta), \mathbf{y}) = -(\mathbf{y} - \hat{\mathbf{y}}) \cdot \mathbf{x}$

(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.
$\hat{y} = \sigma(\theta^T x)$
$\hat{y} = \sigma([0, 0, 0]^T \cdot [1, 3, 2])$
$\hat{y} = \sigma(0) = \mathbf{0}$

The first update is given by,
$\theta^1 = \theta^0 - \eta \nabla_\theta L(f(x;\theta), y)$
$\theta^1 = [0, 0, 0] - 0.1 \cdot -(1 - 0) \cdot [1, 3, 2]$
$\theta^1 = [0, 0, 0] + 0.1 \cdot [1, 3, 2]$
$\theta^{\mathbf{1}} = [\mathbf{0.1}, \mathbf{0.3}, \mathbf{0.2}]$

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.

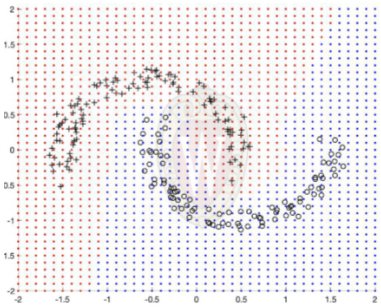

Figure 2 denotes the training points and the approximate boundary for the classes for 1NN.

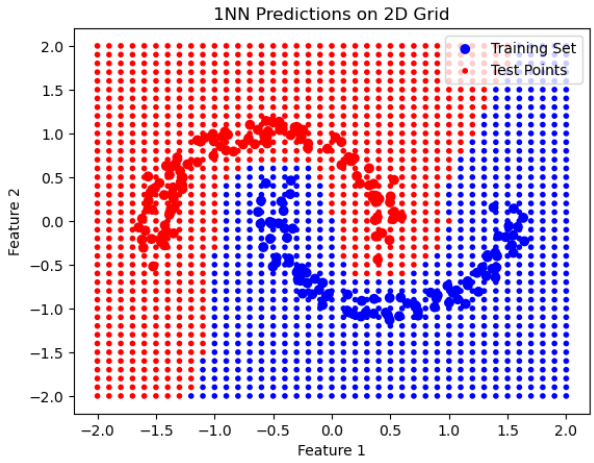

Figure 2: Plot of train and test for 1NN

	Accuracy	**Precision**	**Recall**
Fold1	0.825	0.654	0.818
Fold2	0.853	0.686	0.866
Fold3	0.862	0.721	0.838
Fold4	0.851	0.716	0.816
Fold5	0.775	0.606	0.758
Mean	**0.8332**	**0.6767**	**0.8193**

Table 1: 5-Fold validation results for 1NN on emails.csv

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.

Refer Table. 1 for answers. *(Clickable refrences)*

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.

Refer Table. 2 for answers. *(Clickable refrences)*

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.

	Accuracy	**Precision**	**Recall**
Fold1	0.965	0.910	0.975
Fold2	0.968	0.974	0.974
Fold3	0.984	0.959	0.986
Fold4	0.97	0.933	0.965
Fold5	0.971	0.940	0.967
Mean	**0.971**	**0.931**	**0.973**

Table 2: 5-Fold validation results for Logistic Regression on emails.csv

k	**Accuracy**
1	0.8332
3	0.8424
5	0.8418
7	0.8454
10	0.8557

Table 3: k vs accuracy for k-NN (Question 4)

Refer Table 3 and Fig. 3 for answer. *(Clickable refrences)*

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.

   Refer Fig. 4 for answers. The logistic regression has been run for 1000 epochs with learning rate of 0.01.

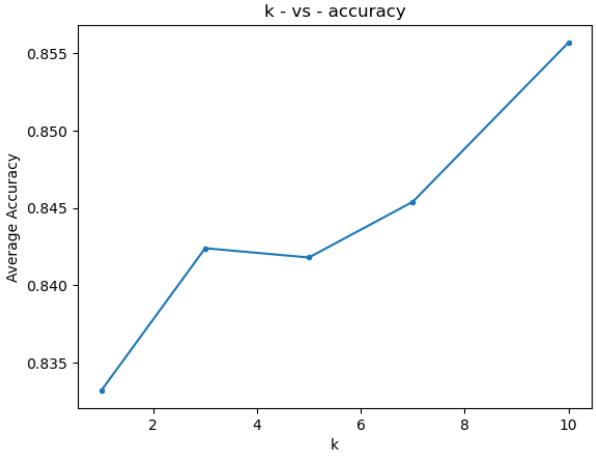

Figure 3: k-vs-accuracy plot

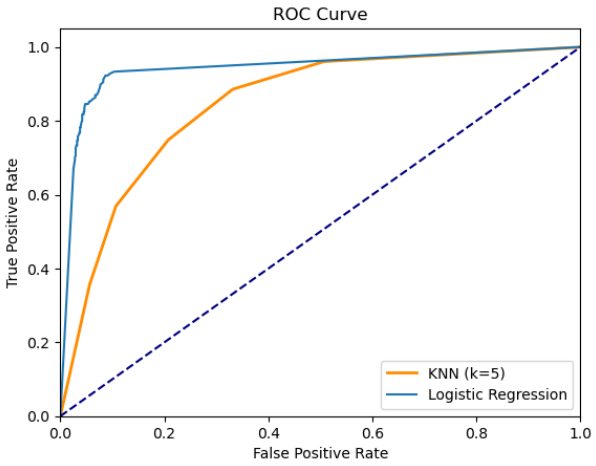

Figure 4: KNN (k=5) vs Logistic Regression ROC curves