HOMEWORK 3

Nick Boddy
nboddy

GitHub Repo: https://github.com/Nick-Boddy/CS760-HW3-kNN-Log-Regr

2023-10-09

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.

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 regression problem because we wish to predict a continuous value, a salary. In this case we have n = 500 data points and p = 3 features (profit, # 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.

   This is a classification problem because we are predicting a discrete value - a class, either success or failure. Here, we will have n = 20 data points and p = 13 features (price, marketing budget, competition price, and 10 other features).

   (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 regression problem since we are predicting a continuous value - a percentage change in the value of a US dollar. Here we will have n = 52 data points and p = 3 features (% change in the US market, % change in the UK market, and % change in the DE 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, $X_1 = X_2 = X_3 = 0$.

X_1	X_2	X_3	$D_{(0,0,0)}$
0	3	0	$\sqrt{0^2+3^2+0^2} = 3$
2	0	0	$\sqrt{2^2+0^2+0^2} = 2$
0	1	3	$\sqrt{0^2+1^2+3^2} \approx 3.16$
0	1	2	$\sqrt{0^2+1^2+2^2} \approx 2.24$
-1	0	1	$\sqrt{(-1)^2+0^2+1^2} \approx 1.41$
1	1	1	$\sqrt{1^2+1^2+1^2} \approx 1.73$

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

With $K = 1$, we take the single nearest neighbor and predict its value. The nearest neighbor is $[-1, 0, 1]$, so we predict **Green**

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

With $K = 3$, we now consider the three nearest neighbors: $[-1, 0, 1], [1, 1, 1], [2, 0, 0]$, which vote Green, Red, and Red, respectively. Plurality would take **Red** as the prediction.

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?

Since X is uniformly distributed on $[0, 1]$, on average, $\frac{1}{10}$ of the available observations are used in the prediction.

(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?

Since $X1$ and $X2$ are both uniformly distributed across $[0, 1]$, we take, on average, 10% of 10% of the available observations, which is $\frac{1}{100}$

(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?

With $p \geq 1$ features, each uniformly distributed, if we only consider observations in the nearest 10% of each and every feature, on average we only consider $\frac{1}{10^p}$ of the available observations when making a prediction.

With $p = 100$, we consider $\frac{1}{10^{100}}$ of the available observations.

(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 explained in part (c), with each new feature, we remove 90% of the available observations for predictions (at least when only taking those within 10% of each feature). Thus, with large p, we lose an exponential number of observations when it comes to prediction. This is an inherent drawback of KNN.

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

We want to use about 10% of training observations. Assuming each feature is uniformly distributed on $[0, 1]$, we want to consider the length l of the sides of the hypercube that contains $\approx 10\%$ of training observations. Because the range of each feature is 1, the hypercube of the entire feature space will always have a volume of 1. Therefore, the volume of the hypercube containing $\approx 10\%$ of the training observations will be $0.1(1) = 0.1$.

$p = 1 : l = 0.1$

$p = 2 : l = \sqrt{0.1}$

$p = 100 : l = \sqrt[100]{0.1}$

$l = \sqrt[p]{0.1} = 0.1^{1/p}$

$\lim_{p\to\infty} 0.1^{1/p} = 1$

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 = $\frac{TP+TN}{TP+FP+TN+FN} = \frac{8+974}{8+16+974+2} = 0.982$

(b) (2 pts) Precision
Precision = $\frac{TP}{TP+FP} = \frac{8}{8+16} = 0.\overline{3}$

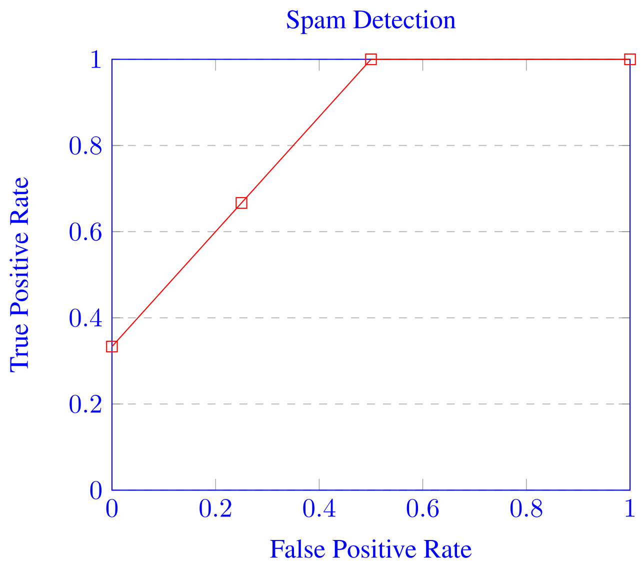
(c) (2 pts) Recall
Recall = $\frac{TP}{TP+FN} = \frac{8}{8+2} = 0.8$

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.

Confidence positive	Correct class	TPR	FPR
0.95	+		
0.85	+	2/6	0/4
0.8	-		
0.7	+		
0.55	+	4/6	1/4
0.45	-		
0.4	+		
0.3	+	6/6	2/4
0.2	-		
0.1	-	6/6	4/4



(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.

Personally speaking, I would assign a much higher misclassification cost for False Positives than for False Negatives. If a real email (that is not spam) were to be misclassified as spam, I would likely never see it. This could have dire consequences since that email may be very important. On the other hand, if a spam email is misclassified as not spam, it's not a big deal to me - I'm fairly good at determining what is and what is not spam and can just simply mark it as spam myself.
Therefore, I would choose a confidence threshold of 0.85. This is because the ROC curve indicates that for this confidence threshold, we have a False Positive Rate of 0, while still maintaining a True Positive Rate of $0.\overline{3}$.

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

We start by finding an expression for $L(f(x;\theta), y)$:

$$L(f(x;\theta), y) = -[y \log f(x;\theta) + (1-y)\log(1-f(x;\theta))]$$

$$= -y \log\left(\frac{1}{1+e^{-\theta^\top x}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\theta^\top x}}\right)$$

$$= -y\left(\log 1 - \log\left(1+e^{-\theta^\top x}\right)\right) + (y-1)\log\left(\frac{1+e^{-\theta^\top x}-1}{1+e^{-\theta^\top x}}\right)$$

$$= y\log\left(1+e^{-\theta^\top x}\right) + (y-1)\left(\log\left(e^{-\theta^\top x}\right) - \log\left(1+e^{-\theta^\top x}\right)\right)$$

$$= y\log\left(1+e^{-\theta^\top x}\right) + (y-1)\log\left(e^{-\theta^\top x}\right) - (y-1)\log\left(1+e^{-\theta^\top x}\right)$$

$$= \log\left(1+e^{-\theta^\top x}\right) + (y-1)(-\theta^\top x)$$

Now that we have simplified our $L(f(x;\theta),y)$, we shall compute $\nabla_\theta L(f(x;\theta),y)$:

$$\nabla_\theta L(f(x;\theta),y) = \nabla_\theta\left[\log\left(1+e^{-\theta^\top x}\right) + (y-1)(-\theta^\top x)\right]$$

$$= \nabla_\theta\left[\log\left(1+e^{-\theta^\top x}\right) + \theta^\top x - y\left(\theta^\top x\right)\right]$$

$$= \frac{1}{\left(1+e^{-\theta^\top x}\right)}\left(-xe^{-\theta^\top x}\right) + x - xy$$

$$= x\left(\frac{-e^{-\theta^\top x}}{\left(1+e^{-\theta^\top x}\right)} + 1 - y\right)$$

$$= x\left(-\left(\frac{1+e^{-\theta^\top x}-1}{\left(1+e^{-\theta^\top x}\right)}\right) + 1 - y\right)$$

$$= x\left(-1 + \frac{1}{1+e^{-\theta^\top x}} + 1 - y\right)$$

$$= x\left(\sigma(\theta^\top x) - y\right)$$

(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.

From our update step equation, we get:

$$\theta^1 = \theta^0 - 0.1x\left(\sigma(\theta^\top x) - y\right)$$

$$= \begin{bmatrix}0 & 0 & 0\end{bmatrix} - 0.1\begin{bmatrix}1 & 3 & 2\end{bmatrix}\left(\sigma\left(\begin{bmatrix}0\\0\\0\end{bmatrix}\cdot\begin{bmatrix}1 & 3 & 2\end{bmatrix}\right) - 1\right)$$

$$= \begin{bmatrix}0 & 0 & 0\end{bmatrix} - \begin{bmatrix}0.1 & 0.3 & 0.2\end{bmatrix}\left(\frac{1}{2} - 1\right)$$

$$= \begin{bmatrix}0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0.05 & 0.15 & 0.1\end{bmatrix}$$

$$= \begin{bmatrix}0.05 & 0.15 & 0.1\end{bmatrix}$$

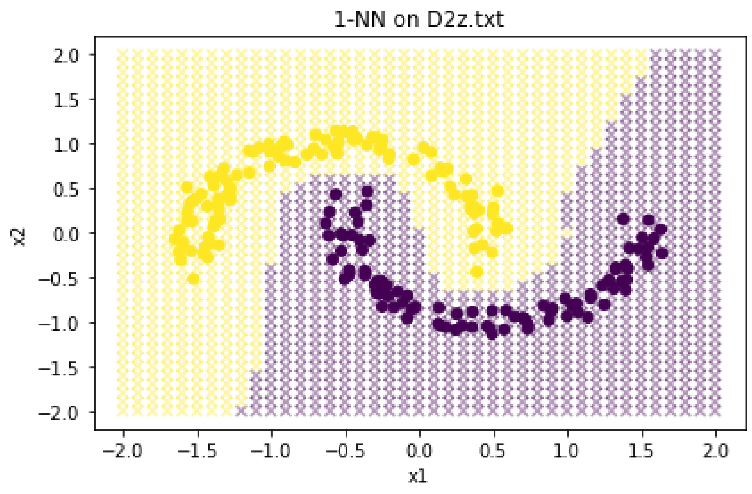
2 Programming (50 pts)

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.

   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.

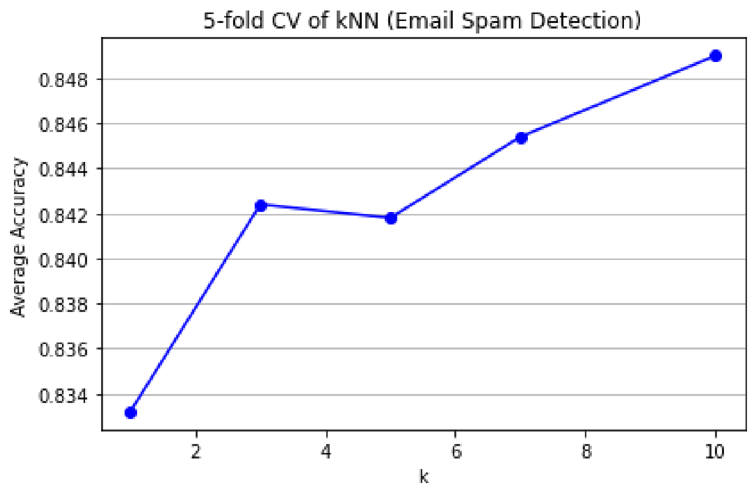
Fold	Accuracy	Precision	Recall
Fold 1	0.825	0.655	0.818
Fold 2	0.853	0.686	0.866
Fold 3	0.862	0.721	0.838
Fold 4	0.851	0.716	0.816
Fold 5	0.775	0.606	0.758

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.

Fold	Accuracy	Precision	Recall
Fold 1	0.929	0.912	0.832
Fold 2	0.931	0.909	0.834
Fold 3	0.913	0.919	0.761
Fold 4	0.908	0.888	0.786
Fold 5	0.890	0.863	0.761

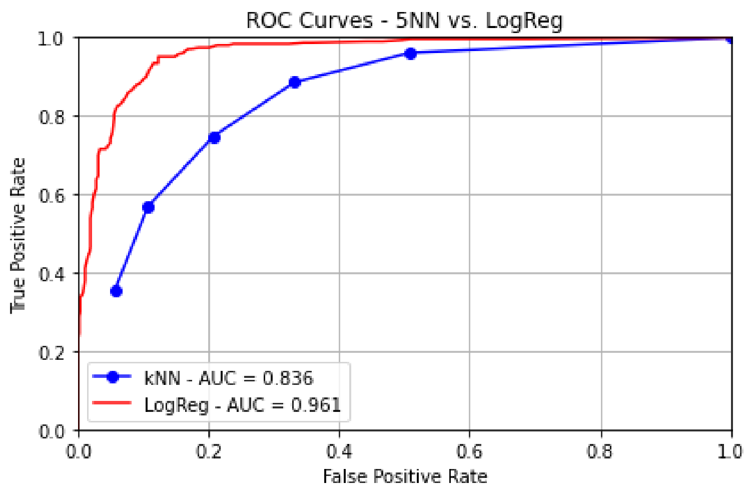
4. (10 pts) Run 5-fold cross validation with kNN varying k (k=1, 3, 5, 7, 10). Plot the average accuracy versus k, and list the average accuracy of each case.
   Expected figure looks like this.

k	Average Accuracy
1	0.8332
3	0.8424
5	0.8418
7	0.8454
10	0.8490



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.


ROC Curves - 5NN vs. LogReg
True Positive Rate
False Positive Rate
kNN - AUC = 0.836
LogReg - AUC = 0.961
0.0
0.2
0.4
0.6
0.8
1.0