ISYE 6740, Spring 2024, Homework 4
100 points

Yuxi Chen

1. Optimization (35 points).

Consider a simplified logistic regression problem. Given m training samples (x^i, y^i), $i = 1, \ldots, m$. The data $x^i \in \mathbb{R}^2$, and $y^i \in \{0, 1\}$. To fit a logistic regression model for classification, we solve the following optimization problem, where $\theta \in \mathbb{R}$ is a parameter we aim to find:

$$\max_\theta \ell(\theta), \tag{1}$$

where the log-likelihood function

$$\ell(\theta) = \sum_{i=1}^{m} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) + (y^i - 1)\theta^T x^i \right\}.$$

1. (10 points) Show step-by-step mathematical derivation for the gradient of the cost function $\ell(\theta)$ in (1).

   Using the chain rule to derive:

$$\ell(\theta) = \sum_{i=1}^{m} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) + (y^i - 1)\theta^T x^i \right\}$$

$$\Rightarrow \frac{\partial \ell}{\partial \theta} = \frac{\partial}{\partial \theta} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) \right\} + \frac{\partial}{\partial \theta} \left\{ (y^i - 1)\theta^T x^i \right\}$$

$$\Rightarrow \frac{\partial}{\partial \theta} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) \right\} = \frac{\exp\{-\theta^T x^i\} x^i}{1 + \exp\{-\theta^T x^i\}}$$

$$\Rightarrow \frac{\partial}{\partial \theta} \left\{ (y^i - 1)\theta^T x^i \right\} = (y^i - 1)x^i$$

$$\Rightarrow \frac{\partial \ell(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left\{ \frac{\exp\{-\theta^T x^i\} x^i}{1 + \exp\{-\theta^T x^i\}} + (y^i - 1)x^i \right\}$$

2. (5 points) Write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does.

   Pseudo-code:

```
1. Initialize parameter($\theta^0$)
2. Determine learning rate($\gamma$) and max iterations, where iterations is $t$
3. Update $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_i (y_i - 1)x_i + \frac{\exp(-{\theta^t}^T x_i)x_i}{1+\exp(-{\theta^t}^T x_i)}$
4. While $||\theta^{t+1} - \theta^t|| > \epsilon$ and $t <$ max iterations
```

3. (5 points) Write the pseudo-code for performing the **stochastic gradient descent** algorithm to solve the training of logistic regression problem (1). Please explain the difference between gradient descent and stochastic gradient descent for training logistic regression.

   Pseudo-code:

```
1. Initialize parameter($\theta^0$)
2. Determine learning rate($\gamma$) and max batches($k$)
3. Each batch, randomly sample of a subset $S_k$ of data points $(x_i, y_i), \sum_{i \in S_k}$
4. For each $S_k$, update $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_{i \in S_k} (y^i - 1)x^i + \frac{\exp(-{\theta^t}^T xi)xi}{1+\exp(-{\theta^t}^T xi)}$
5. While $||\theta^{t+1} - \theta^t|| > \epsilon$ and $k <$ max batches
```

   The difference between gradient descent and stochastic gradient descent is determined by the method used to updated θ. Gradient updates θ over the entire dataset all at once, whereas stochastic gradient descent does so in batches and can be performed faster and more efficiently with a larger dataset.

4. (15 points) We will **show that the training problem in basic logistic regression problem is concave.** Derive the Hessian matrix of $\ell(\theta)$ and based on this, show the training problem (1) is concave. Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

   Given the problem:

$$\ell(\theta) = \sum_{i=1}^{m} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) + (y^i - 1)\theta^T x^i \right\}$$

$$\Rightarrow \frac{\partial \ell(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left\{ \frac{\exp\{-\theta^T x^i\} x^i}{1 + \exp\{-\theta^T x^i\}} + (y^i - 1)x^i \right\}$$

$$\Rightarrow \frac{\partial^2 \ell(\theta)}{\partial \theta^2} = -\sum_{i=1}^{m} \frac{\exp\{-\theta^T x^i\} {x^i}^2}{(1 + \exp\{-\theta^T x^i\})^2} < 0$$

   To determine if the problem concave we need to take the second derivative and if it is less than 0, we can state that it is concave. Using gradient descent, we will converge to the maximum point, which will be the unique global optimizer.

2. Bayes Classifier for spam filtering (35 points)

In this problem, we will use the Bayes Classifier algorithm to fit a spam filter by hand. This will enhance your understanding to the Bayes classifier and build intuition. This question does not involve any programming but only derivation and hand calculation.

Spam filters are used in all email services to classify received emails as "Spam" or "Not Spam". A simple approach involves maintaining a vocabulary of words that commonly occur in "Spam" emails and classifying an email as "Spam" if the number of words from the dictionary that are present in the email is over a certain threshold. We are given the vocabulary consists of 15 words

$V = \{$secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza$\}$.

We will use V_i to represent the ith word in V. As our training dataset, we are also given 3 example spam messages,

- million dollar offer for today
- secret offer today

- secret is secret

and 4 example non-spam messages

- low price for valued customer today
- play secret sports today
- sports is healthy
- low price pizza today

Recall that the Naive Bayes classifier assumes the probability of an input depends on its input feature. The feature for each sample is defined as $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}]^T$, $i = 1, \dots, m$ and the class of the ith sample is $y^{(i)}$. In our case the length of the input vector is $d = 15$, which is equal to the number of words in the vocabulary V. Each entry $x_j^{(i)}$ is equal to the number of times word V_j occurs in the i-th message.

1. (5 points) Calculate class prior $\mathbb{P}(y = 0)$ and $\mathbb{P}(y = 1)$ from the training data, where $y = 0$ corresponds to spam messages, and $y = 1$ corresponds to non-spam messages. Note that these class prior essentially corresponds to the frequency of each class in the training sample. Write down the feature vectors for each spam and non-spam messages.

   Given 3 spam messages, and 4 non-spam messages. We have $P(y = 0) = \frac{3}{7}$ and $P(y = 1) = \frac{4}{7}$.

   Given $V = \{$1:secret, 2:offer, 3:low, 4:price, 5:valued, 6:customer, 7:today, 8:dollar, 9:million, 10:sports, 11:is, 12:for, 13:play, 14:healthy, 15:pizza$\}$

index:	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
	$P(y = 0) = \frac{3}{7}$
million dollar offer for today:	[0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0]
secret offer today:	[1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
secret is secret:	[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
	$P(y = 1) = \frac{4}{7}$
low price for valued customer today:	[0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0]
play secret sports today:	[1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0]
sports is healthy:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0]
low price pizza today:	[0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]

2. (15 points) Assuming the keywords follow a multinomial distribution, the likelihood of a sentence with its feature vector x given a class c is given by

$$\mathbb{P}(x|y = c) = \frac{n!}{x_1! \cdots x_d!} \prod_{k=1}^{d} \theta_{c,k}^{x_k}, \quad c = \{0, 1\}$$

   where $n = x_1 + \cdots x_d$, $0 \le \theta_{c,k} \le 1$ is the probability of word k appearing in class c, which satisfies

$$\sum_{k=1}^{d} \theta_{c,k} = 1, \quad c = \{0, 1\}.$$

   Given this, the complete log-likelihood function for our training data is given by

$$\ell(\theta_{0,1}, \dots, \theta_{0,d}, \theta_{1,1}, \dots, \theta_{1,d}) = \sum_{i=1}^{m} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k}$$

(In this example, $m = 7$.) Calculate the maximum likelihood estimates of $\theta_{0,1}$, $\theta_{0,7}$, $\theta_{1,1}$, $\theta_{1,15}$ by maximizing the log-likelihood function above.
(Hint: We are solving a constrained maximization problem and you will need to introduce Lagrangian multipliers and consider the Lagrangian function.)

Using the lagrangian function:

$$L(x, \lambda) = f(x) + \lambda g(x)$$

$$\Rightarrow \sum_{i=1}^{m} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{c,k} + \lambda_c \, (\sum_{k=1}^{d} \theta_{c,k})$$

To find the maximum likelihood estimates we will need to take the derivative of the lagrangian function, given the two classes, $c = \{0, 1\}$.

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_{c,k}} = \sum_{c=\{0,1\}} \frac{x_k^{(i)}}{\theta_{c,k}} + \lambda_c = 0$$

Solving for λ:

$$-\lambda_c = \sum_{c=\{0,1\}} \frac{x_k^{(i)}}{\theta_{c,k}}$$

Solving for $\theta_{c,k}$:

$$\theta_{c,k} = \sum_{c=\{0,1\}} \frac{x_k^{(i)}}{\lambda_c}$$

Given that:

$$\sum_{k=1}^{d} \theta_{c,k} - 1 = 0$$

$$\Rightarrow \sum_{k=1}^{d} \theta_{c,k} = 1$$

Putting it all together:

$$\theta_{c,k} = \frac{\sum_{c=\{0,1\}} x_k^{(i)}}{\sum_{c=\{0,1\}} \frac{x_k^{(i)}}{\theta_{c,k}}}$$

Solving for the $\theta_{0,1}$, $\theta_{0,7}$, $\theta_{1,1}$, $\theta_{1,15}$:
$\theta_{0,1} = \frac{3}{11}$
$\theta_{0,7} = \frac{2}{11}$
$\theta_{1,1} = \frac{1}{17}$
$\theta_{1,15} = \frac{1}{17}$

3. (15 points) Given a test message "today is secret", using the Naive Bayes classier that you have trained in Part (a)-(b), to calculate the posterior and decide whether it is spam or not spam.

From the message "today is secret", we know to use $\theta_{c,7}$, $\theta_{c,11}$ and $\theta_{c,7}$. In this case, we will use Bayes Rule.

$$q_c(x) = P(y = c|x) = \frac{P(x|y = c)P(y = c)}{P(x|y = 0)P(y = 0) + P(x|y = 1)P(y = 1)}$$

Given the values from part (a)-(b), for part(b) the values were verified by counting the occurrence in each message:
$P(y = 0) = \frac{3}{7}$
$P(y = 1) = \frac{4}{7}$

$P(x|y=0) = \theta_{0,7}\theta_{0,11}\theta_{0,1} = \frac{2}{11} * \frac{1}{11} * \frac{3}{11}$
$P(x|y=1) = \theta_{1,7}\theta_{1,11}\theta_{1,1} = \frac{3}{17} * \frac{1}{17} * \frac{1}{17}$

To decide whether the message is spam or not we can use Bayes decision rule. Which is, if $q_1(x) > q_0(x)$, y=1, thus spam. Otherwise, y=0, not spam.

Solving for $q_0(x)$:

$$P(y=0|x) = \frac{P(x|y=0)P(y=0)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)}$$
$$\Rightarrow \frac{(\frac{2}{11} * \frac{1}{11} * \frac{3}{11})(\frac{3}{7})}{(\frac{2}{11} * \frac{1}{11} * \frac{3}{11})(\frac{3}{7}) + (\frac{3}{17} * \frac{1}{17} * \frac{1}{17})(\frac{4}{7})}$$
$$\Rightarrow \frac{14739}{17401} \approx 0.84702$$

Solving for $q_1(x)$:

$$P(y=1|x) = \frac{P(x|y=1)P(y=1)}{P(x|y=0)P(y=0) + P(x|y=1)P(y=1)}$$
$$\Rightarrow \frac{(\frac{3}{17} * \frac{1}{17} * \frac{1}{17})(\frac{4}{7})}{(\frac{2}{11} * \frac{1}{11} * \frac{3}{11})(\frac{3}{7}) + (\frac{3}{17} * \frac{1}{17} * \frac{1}{17})(\frac{4}{7})}$$
$$\Rightarrow \frac{2662}{17401} \approx 0.15297$$

Since $q_1(x) < q_0(x)$, we can conclude that the message is spam.

3. Comparing classifiers: Divorce classification/prediction (30 points)

In lectures, we learn different classifiers. This question is compare them on two datasets. Python users, please feel free to use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar libraries in other languages of your choice to perform the tasks.

This dataset is about participants who completed the personal information form and a divorce predictors scale. The data is a modified version of the publicly available at `https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set` (by injecting noise so you will not get the exactly same results as on UCI website). The dataset **marriage.csv** is contained in the homework folder. There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce").

We are going to compare the following classifiers (**Naive Bayes, Logistic Regression, and KNN**). Use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train_test_split to split the dataset.

Remark: Please note that, here, for Naive Bayes, this means that we have to estimate the variance for each individual feature from training data. When estimating the variance, if the variance is zero to close to zero (meaning that there is very little variability in the feature), you can set the variance to be a small number, e.g., $\epsilon = 10^{-3}$. We do not want to have include zero or nearly variance in Naive Bayes. This tip holds for both Part One and Part Two of this question.

1. (15 points) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

   NB Testing Accuracy: 0.9705882352941176
   LR Testing Accuracy: 0.9705882352941176
   KNN Testing Accuracy: 0.9705882352941176

   From the testing accuracy obtained from three different classifiers. We observed the same results of 97.05% accuracy, unless we run additional testing. Such as, specificity, precision, or AUC. The high accuracy among the three classifiers can be attributed to the data. This indicates that the data must have a linear separation without additional overlapping or noise.

2. (15 points) Now perform PCA to project the data into two-dimensional space. Build the classifiers (**Naive Bayes, Logistic Regression, and KNN**) using the two-dimensional PCA results. Plot the data points and decision boundary of each classifier in the two-dimensional space. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

   In the figures below, we have a decision boundary for each classifier separating the data points. The decision boundary for naives bayes, which has gaussian distribution has a slightly curved quadratic shape. The decision boundary for logistic regression has a linear line shape, Lastly, since KNN is non parametric, its shapes adjusts to the data. In this case, it has a more angular shape, which can be smoothed out as we increase the k value. The shape of the boundary line follows what is expected for these classifier, mainly because the data fits well. If we had any outliers or noise it can affect the shape.

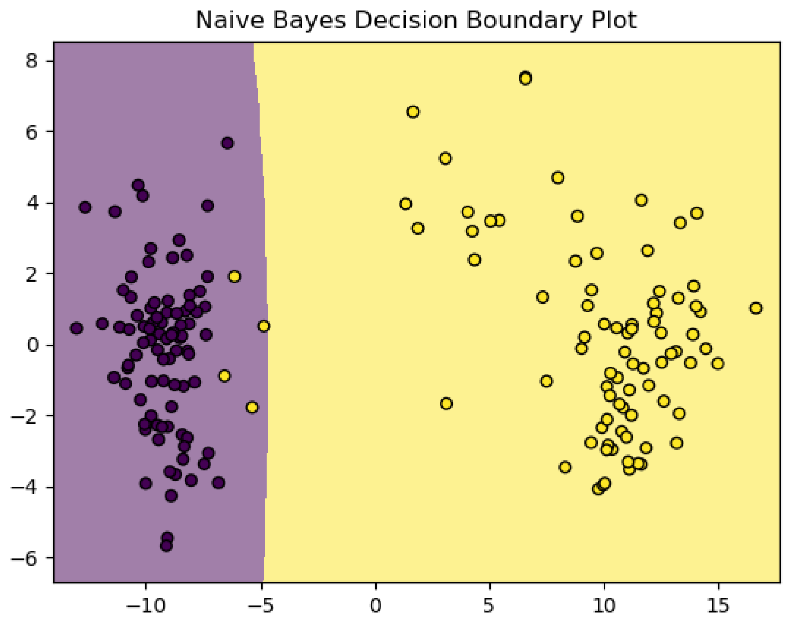

Figure 1: Naives Bayes Plot

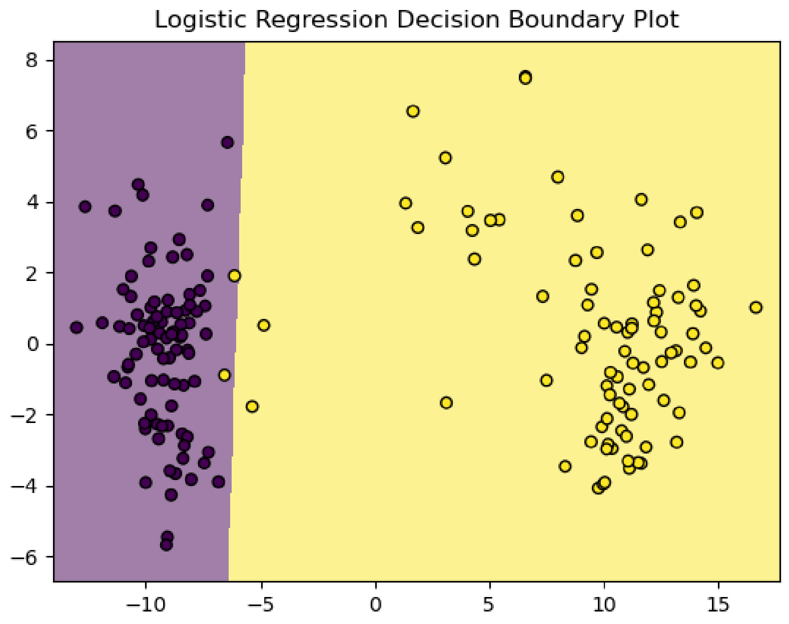

Figure 2: Logistic Regression Plot

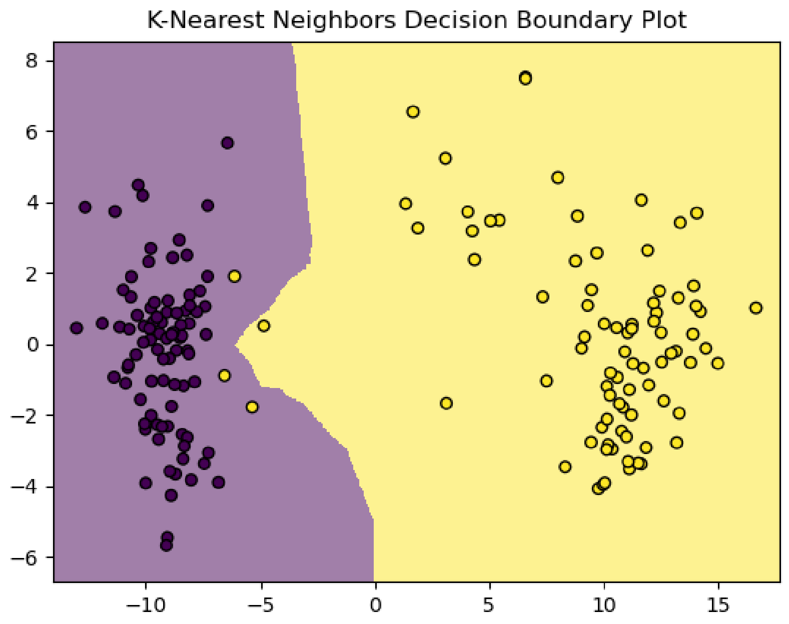

Figure 3: KNN Plot