ISYE 6740 Homework 4

Total 100 points + 15 bonus points.

Farshad Rafiei

1. **Basic optimization.** (40 points.)

   Consider a simplified logistic regression problem. Given m training samples (x^i, y^i), $i = 1, \dots, m$. The data $x^i \in \mathbb{R}$ (note that we only have one feature for each sample), 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

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

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

   Answer:

$$\begin{aligned}\frac{\partial \ell}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{i=1}^{m} \left\{ -\log(1 + \exp\{-\theta x^i\}) + (y^i - 1)\theta x^i \right\} \\ &= \sum_{i=1}^{m} \left\{ \frac{\partial}{\partial \theta}(-\log(1 + \exp\{-\theta x^i\})) + \frac{\partial}{\partial \theta}((y^i - 1)\theta x^i) \right\} \\ &= \sum_{i=1}^{m} \left\{ \frac{\exp\{-\theta x^i\} x^i}{1 + \exp\{-\theta x^i\}} + (y^i - 1)x^i \right\}\end{aligned} \tag{2}$$

   (b) (10 points) Write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does. (pseudo-code means you will write down the steps of the algorithm, not necessarily any specific programming language.)

   Answer: Algorithm (1) shows the pseudocode for performing gradient descent to find the optimizer $\theta^\star$.

   (c) (10 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.

   Answer: Algorithm (2) shows the pseudocode for stochastic gradient descent to solve the training of the logistic regression problem. The difference between algorithm (1) and (2) is in the updating step. In stochastic gradient descent, new θ is found only based on a sample, and not the whole data. On contrary, gradient descent uses whole dataset to update the θ at each step, which is not desirable since in large datasets this will result in a huge computation cost.

Algorithm 1 Gradient descent algorithm

```
$\theta^0 \leftarrow 0$ (*initialize* $\theta$)
$\theta^1 \leftarrow \gamma_1 \sum_{i=1}^{m} \{ \frac{\exp\{-\theta x^i\} x^i}{1+\exp\{\theta x^i\}} + (y^i - 1)x^i \}$
**while** $|| \theta^{t+1}$ - $\theta^t|| < \epsilon$ **do**
    $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_{i=1}^{m} \{ \frac{\exp\{-\theta x^i\} x^i}{1+\exp\{\theta x^i\}} + (y^i - 1)x^i \}$
**end while**
```

Algorithm 2 Stochastic gradient descent algorithm

```
$\theta^0 \leftarrow 0$ (*initialize* $\theta$)
$\theta^1 \leftarrow \gamma_1 \sum_{i \in S_1} \{ \frac{\exp\{-\theta x^i\} x^i}{1+\exp\{\theta x^i\}} + (y^i - 1)x^i \}$
**while** $|| \theta^{t+1}$ - $\theta^t|| < \epsilon$ **do**
    *randomly sample subset* $S_t$ *data point* $(x^i, y^i)$, $i \in S_t$
    $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_{i \in S_t} \{ \frac{\exp\{-\theta x^i\} x^i}{1+\exp\{\theta x^i\}} + (y^i - 1)x^i \}$
**end while**
```

(d) (10 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 (note that in this case, since we only have one feature, the Hessian matrix is just a scalar). Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

Answer:

$$\begin{aligned}
\frac{\partial^2 \ell}{\partial \theta^2} &= \frac{\partial}{\partial \theta} \sum_{i=1}^{m} \{ \frac{\exp\{-\theta x^i\} x^i}{1+\exp\{-\theta x^i\}} + (y^i - 1)x^i \} \\
&= \frac{\partial}{\partial \theta} \sum_{i=1}^{m} \{ (1 - \frac{1}{1+\exp\{-\theta x^i\}}) x^i + (y^i - 1)x^i \} \\
&= \frac{\partial}{\partial \theta} \sum_{i=1}^{m} \{ (y^i - \frac{1}{1+\exp\{-\theta x^i\}}) x^i \} \\
&= -\sum_{i=1}^{m} \frac{x^i}{1+\exp\{-\theta x^i\}} x^i \\
&= -\sum_{i=1}^{m} \frac{(x^i)^2}{1+\exp\{-\theta x^i\}}
\end{aligned} \tag{3}$$

The last line of equation (3) always takes negative values (since $(x^i)^2$ and $1 + \exp\{-\theta x^i\}$ are positive values) and therefore it indicates that the second derivative of log-likelihood function is always negative. That means that the training problem is concave and starting from any initial value and moving towards the gradients of likelihood function, we will end up with global optimizer, which is the max point on concave function. Since there is no other local maxima (or minima), the (stochastic) gradient descent algorithm will always achieve a unique global optimizer.

2. **Comparing Bayes, logistic, and KNN classifiers.** (60 points)

In lectures, we learn three different classifiers. This question is to implement and compare them. 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.

Part One (Divorce classification/prediction). (30 points)

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").

Build three classifiers using (**Naive Bayes, Logistic Regression, 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.*

(a) (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.

Answer: Instead of reporting the performance for each classifier on a single test data, I created 100 different test dataset by randomly splitting the train and test dataset. Following shows the average over all test accuracies obtained from this procedure.
The average test accuracy for **Naive Bayes classifier** is **97.97%**.
The average test accuracy for **Logistic Regression classifier** is **97.91%**.
The average test accuracy for **Nearest Neighbor ($k = 25$) classifier** is **97.97%**.
All of the classifiers perform very well on this dataset with Logistic Regression performance being slightly poor when comparing to the other two. Since basis of Logistic Regression classification is linear discrimination of the data points, the non-linear distribution of the points could be possible explanation for weaker performance of this classifier for this dataset. However, again, all of these classifiers perform very well on this dataset.

(b) (15 points) Now perform PCA to project the data into two-dimensional space. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

Answer: decision boundaries are shown in Figure (1). The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.
Logistic Regression classifier has linear boundaries. Whereas Naive Bayes classifier has curved (non-linear) decision boundaries. They both act relatively same, and has similar performance in separating the data with different classes. Decision boundary for Nearest Neighbor ($k = 25$) classifier is relatively different. The boundary is nonlinear and highly dependent on the position of the data points scattered all over the space. This classifier performs as well as the other two classifiers for this dataset, though.

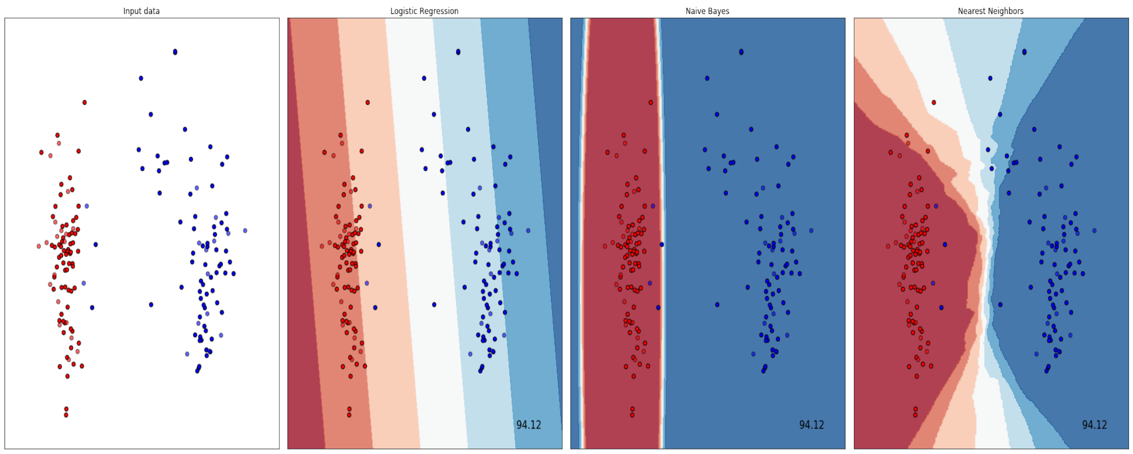

Figure 1: **Data points scatter plot and decision boundary visualization**. Data points are shown for *Divorce* dataset after projecting the points into two dimensional space using PCA. Scatter plots contain a visualization of decision boundaries which separates the data points based on Logistic regression, Naive Bayes and Nearest Neighbor ($k = 25$) classification methods.

Part Two (Handwritten digits classification). (30 points) Repeat the above using the **MNIST Data** in our previous homework. Here, give "digit" 6 label $y = 1$, and give "digit" 2 label $y = 0$. All the pixels in each image will be the feature (predictor variables) for that sample (i.e., image). Our goal is to build classifier to such that given a new test sample, we can tell is it a 2 or a 6. Using the first 80% of the samples for training and remaining 20% for testing.

(a) (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.

Answer: Similar to part one of this question, instead of reporting the test accuracy for single setting of train-test split, we performed the split 100 times and are reporting the average accuracy.
The average test accuracy for **Naive Bayes** classifier is **78.60%**.
The average test accuracy for **Logistic Regression** classifier is **97.84%**.
The average accuracy for **Nearest Neighbor** classifier is **99.08%**.
Naive Bayes classifier does not perform as well as the other two classifiers. It assumes that the features are independent of each other and probably because of this assumption, it fails to perform well on discriminating the digits in MNIST dataset. Intuitively, the location of the brighter points in a digit image, are dependent on each other. Therefore, Naive Bayes may not be the best option while working with MNIST dataset. On the other hand Nearest Neighbor works the best, apparently because of the similarity of data points in high dimensional spaces. The distance between the points with same labels are potentially less than the distance of those from opposite classes. In this setting, Logistic Regression work very well on this dataset as well, potentially because of the linearly separability of the data points in high dimensional spaces.

(b) (15 points) Now perform PCA to project the data into two-dimensional space. Plot the data points and decision boundary of each classifier. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

Answer: decision boundaries are shown in Figure (2). The plots show training points in solid colors and testing points semi-transparent. The lower right shows the classification accuracy on the test set.

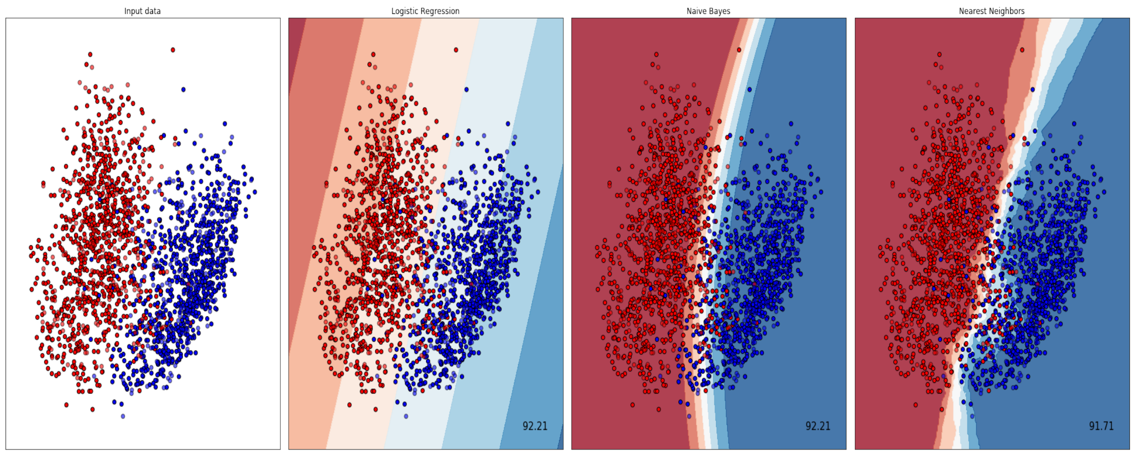

Figure 2: **Data points scatter plot and decision boundary visualization**. Data points are shown for *MNIST* dataset after projecting the points into two dimensional space using PCA. Scatter plots contain a visualization of decision boundaries which separates the data points based on Logistic regression, Naive Bayes and Nearest Neighbor ($k = 25$) classification methods.

Here the data sound to be linearly separable when mapped into two dimensional space. For a specific split of train-test data, you can observe that all models perform equally well in classification of MNIST dataset. The boundaries of all of the models look very similar in this setting. Even the boundary created by Nearest Neighbor ($k = 25$) classifier looks approximately linear despite its highly non-linear nature. Looking at the boundaries in detail, linear boundary of Logistic Regression, curved boundary of Naive Bayes and highly non-linear boundary of Nearest Neighbor could be discerned.

3. **Naive Bayes for spam filtering**. (15 points)

In this problem, we will use the Naive Bayes algorithm to fit a spam filter by hand. This will enhance your understanding to 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

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

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
- secret offer today
- secret is secret

and 4 example non-spam messages

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

- low price pizza

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 $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]^T$, $i = 1, \ldots, m$ 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.

(a) (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.

Answer: The class priors are simply correspond to the frequency of each class in the training sample. Therefore, $\mathbb{P}(y = 0) = \frac{3}{7}$ and $\mathbb{P}(y = 1) = \frac{4}{7}$.
Feature vectors are summerized in Table (1).

Table 1: Feature vectors for spam and non-spam messages

Message	secret	offer	low	price	valued	customer	today	dollar	million	sports	is	for	play	healthy	pizza
spam	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
spam	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
spam	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0
non-spam	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0
non-spam	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0
non-spam	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
non-spam	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1

(b) (5 points) In the Naive Bayes model, assuming the keywords are independent of each other (this is a simplification), the likelihood of a sentence with its feature vector x given a class c is given by

$$\mathbb{P}(x|y = c) = \prod_{k=1}^{d} \theta_{c,k}^{x_k}, \quad c = \{0, 1\}$$

where $0 \le \theta_{c,k} \le 1$ is the probability of word k appearing in class c, which satisfies

$$\theta_{0,k} = 1 - \theta_{1,k}, \quad \forall k.$$

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

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

(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: you can introduce Lagrangian multipliers, or directly substitute the $\theta_{0,k} = 1 - \theta_{1,k}$ into the objective function so you do not need to worry about the constraint.)

Answer: To compute the requested maximum likelihood estimates, we have the take the first derivative of log likelihood function mentioned above and set it equal to zero. We substitute the $\theta_{0,k} = 1 - \theta_{1,k}$ into the likelihood function and solve the problem with respect to $\theta_{1,k}$. Therefore:

$$
\begin{aligned}
\frac{\partial \ell}{\partial \theta_{1,k}} &= \frac{\partial}{\partial \theta_{1,k}} \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k} \\
&= \frac{\partial}{\partial \theta_{1,k}} \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \{(1-y^{(i)}) \log \theta_{0,k} + y^{(i)} \log \theta_{1,k}\} \\
&= \frac{\partial}{\partial \theta_{1,k}} \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \{(1-y^{(i)}) \log (1-\theta_{1,k}) + y^{(i)} \log \theta_{1,k}\} \\
&= \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \{\frac{y^{(i)}}{\theta_{1,k}} - \frac{1-y^{(i)}}{1-\theta_{1,k}}\} \\
&= \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \{\frac{y^{(i)}(1-\theta_{1,k}) - (1-y^{(i)})\theta_{1,k}}{\theta_{1,k}(1-\theta_{1,k})}\} \\
&= \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \{\frac{y^{(i)}-\theta_{1,k}}{\theta_{1,k}(1-\theta_{1,k})}\}
\end{aligned} \tag{4}
$$

By setting the last line of equation (4), equal to zero, we will have:

$$
\theta_{1,k} = \frac{\sum_{i=1}^{m} x_k^{(i)} (y^{(i)} = 1)}{\sum_{i=1}^{m} x_k^{(i)}} \tag{5}
$$

And generally we have:

$$
\boxed{\theta_{c,k} = \frac{\sum_{i=1}^{m} x_k^{(i)} (y^{(i)} = c)}{\sum_{i=1}^{m} x_k^{(i)}}} \tag{6}
$$

Now, we compute the requested likelihood estimates:
$\theta_{0,1} = p(secret|spam) = \frac{3}{4} = 0.75$

$\theta_{0,7} = p(today|spam) = \frac{1}{2} = 0.5$

$\theta_{1,1} = p(secret|non-spam) = \frac{1}{4} = 0.25$

$\theta_{1,15} = p(pizza|non-spam) = \frac{1}{1} = 1$

(c) (5 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.

Answer: $P(spam|today\ is\ secret) = p(spam)p(today|spam)p(is|spam)p(secret|spam) = \frac{3}{7}\frac{1}{2}\frac{1}{2}\frac{3}{4} = \frac{9}{112}$
$P(non-spam|today\ is\ secret) = p(non-spam)p(today|non-spam)p(is|non-spam)p(secret|non-spam) = \frac{4}{7}\frac{1}{2}\frac{1}{2}\frac{1}{4} = \frac{4}{112}$
According to the probabilities found above, we can normalize them based on Bayes theorem. We have $p(spam|today\ is\ secret) = \frac{9}{9+4} = 0.69$ and $p(non-spam|today\ is\ secret) = \frac{4}{9+4} = 0.31$. The conclusion is that the message is most probably a ***spam*** message.