ISYE 6740 Homework 4

Total 100 points + 15 bonus points.

Shasha Liao

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

   Consider a simplified logistic regression problem. Given m training samples (x^i, y^i), $i = 1, \ldots, m$. The data $x^i \in \mathbb{R}$ (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).

   Solution: Since $\theta \in \mathbb{R}$, the gradient of $\ell(\theta)$ is just the derivative of $\ell(\theta)$. Then by Chain rule, we have

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

   (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.)

   Solution: We may adjust the learning rate γ base on the rate of convergence.

```
Data: $(x^i, y^i)$ for $i = 1, ..., m$.
Result: $\theta^*$
Initialization: $\theta^0, \gamma, \epsilon$;
do
    $\theta^{t+1} \leftarrow \theta^t + \gamma \sum_{i=1}^{m} \left\{ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)x^i \right\}$;
while $\|\theta^{t+1} - \theta^t\| > \epsilon$;
```

Algorithm 1: Batch Gradient Descent

(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.

Solution: We may tune the learning rate γ^0 base on the rate of convergence.

```
**Data:** $(x^i, y^i)$ for $i = 1, ..., m$.
**Result:** $\theta^*$
Initialization: $\theta^0, \gamma^0, \epsilon, K$;
Randomly split the indices of our data points into subsets $S_k$, $k = 1, 2, ..., K$ ;
do
    for $k = 1, 2, ..., K$ do
        $\theta^{t+1} \leftarrow \theta^t + \gamma^t \sum_{i \in S_k} \left\{ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)x^i \right\}$;
        $\gamma^{t+1} \leftarrow \frac{\gamma^0}{t+1}$
    end
while $\| \sum_{i=1}^m \left\{ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)x^i \right\} \| > \epsilon$;
```

Algorithm 2: Stochastic Gradient Descent

Comparing GD(Gradient Descent) and SGD(Stochastic Gradient Descent):

- Way of Convergence: GD increases the likelihood function value at each iteration as long as we choose a suitable learning rate, while SGD may not increase the likelihood function value at each step but will decrease it in an oscillating fashion.
- Speed of Convergence: For a large dataset, GD converges very slowly since it utilize all the data for one single update of θ, while SGD can converge much faster as it only utilize a small subset of the data for each update of θ.
- Conclusion: If the data set is small, I prefer GD as the computation can still be fast and it increase the likelihood at each step. If the data set is large, I prefer SGD for a faster rate of convergence.

(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.

Solution: From part (a), we have

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

So

$$
\begin{aligned}
\ell''(\theta) &= \sum_{i=1}^{m} \frac{d}{d\theta} \left\{ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)x^i \right\} \\
&= \sum_{i=1}^{m} \frac{d}{d\theta} \frac{-x^i}{1+\exp\{-\theta x^i\}} \\
&= \sum_{i=1}^{m} \frac{-(x^i)^2}{(1+\exp\{-\theta x^i\})^2} \\
&< 0.
\end{aligned}
$$

Therefore, $\ell(\theta)$ is a concave function and has a unique global maximum point.

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.

*If we have a feature x_i with constant value, then Naive Bayes will suffer from dividing by zero when it tries to fit a Gaussian distribution for x_i since the variance of x_i is zero. In Sklearn GaussianNB, this issue is avoided by using a parameter **var_smoothing**, which is the portion of the largest variance of all features that is added to variances for calculation stability. If **var_smoothing** is too large, the fitted Naive Bayes model will suffer from under-fitting because large variances lead to a simple model.*

(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.

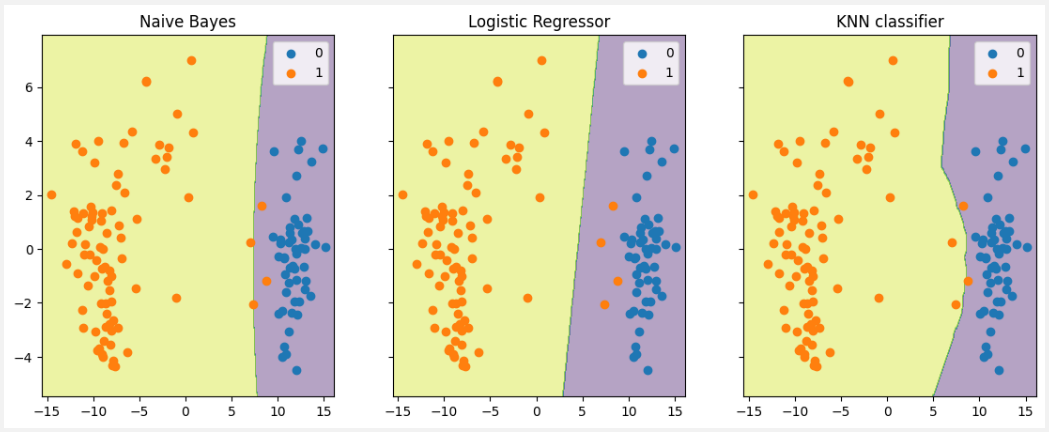
Solution:

Classifier	Testing accuracy
Naive Bayes	1.00
Logistic Regression	0.85
KNN	1.00

From the table above, we see that Naive Bayes and KNN did a much better job than Logistic Regression. I guess the reason is that the two classes in the testing set are not linear separable in the 54 dimensional feature space, which explains that the nonlinear classifiers like Naive Bayes and KNN perform better than the linear classifier Logistic Regression.

(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.

Solution:



Clearly, Logistic Regressor has a perfect linear decision boundary, while both Naive Bayes and KNN has nonlinear decision boundaries.

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.

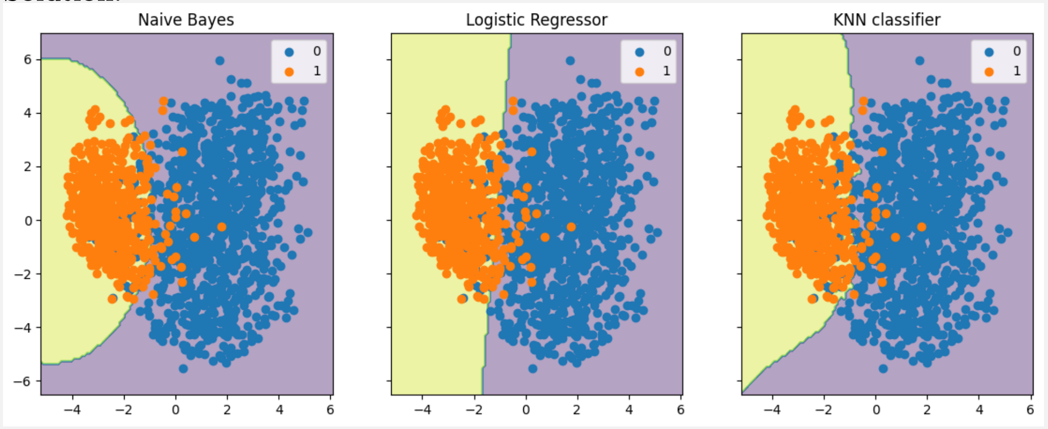
Solution:

Classifier	Testing accuracy
Naive Bayes	0.97
Logistic Regression	0.96
KNN	1.00

KNN has the highest accuracy this time. This might due to the highly nonlinearity of the actual boundary between the two classes in testing set. Even though Naive Bayes also has nonlinear boundaries, it is restricted to piecewise quadratic decision boundaries. So, Naive Bayes is not as flexible as KNN. The results shows that the more nonlinearity a decision boundary has, the higher accuracy it achieves.

(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.

Solution:



Naive Bayes has a piecewise quadratic decision boundary. Logistic Regressor has a linear decision boundary. KNN has a highly nonlinear decision boundary.

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

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

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.

(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 $0 \leq \theta_{c,k} \leq 1$ 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.)

(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.