HW 4

1. Optimization

1. Show step-by-step mathematical derivation for the gradient of the cost function l(θ) in (1).
from slide 40 from module 9

From logistic regression, we can derive the following probabilities:
$P(y = 1|x, \theta) = \frac{1}{1+e^{-\theta x}}$
$P(y = 0|x, \theta) = \frac{e^{-\theta x}}{1+e^{-\theta x}}$

Using the definition of the cost function, apply the log product rule:
$l(\theta) := log\Pi_{i=1}^{m} P(y^i|x^i, \theta)$
$= log(\Sigma_{i=1}^{m} P(y^i|x^i, \theta))$
$= \Sigma_{i=1}^{m} log(P(y^i|x^i, \theta))$
$= \Sigma_{i=1}^{m} y_i log(P(y = 1|x^i, \theta)) + (1 - y_i)log(P(y = 0|x^i\theta))$

Plugging in the definitions from above and simplifying:
$= \Sigma_{i=1}^{m} y_i log(\frac{1}{1+e^{-\theta x^i}}) + (1 - y_i)log(\frac{e^{-\theta x^i}}{1+e^{-\theta x^i}})$
$= \Sigma_{i=1}^{m} y_i[log(1) - log(1 + e^{-\theta x^i})] + (1 - y_i)[log(e^{-\theta x^i}) - log(1 + e^{-\theta x^i})]$
$= \Sigma_{i=1}^{m} y_i[-log(1 + e^{-\theta x^i}) + (1 - y_i)[-\theta x^i - log(1 + e^{-\theta x^i})]$
$= \Sigma_{i=1}^{m} -log(1 + e^{-\theta^T x^i}) + (y^i - 1)\theta^T x^i$

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

Initialize θ^0 and number of max iterations. t = 1

while $||\theta^{t+1} - \theta^t|| > \epsilon$ and t < max_iterations:
$\theta^{t+1} = \theta^t + \gamma_t \Sigma_i (y^j - 1)x^i + \frac{exp(\theta^{t^T} x^i)x^i}{1+exp(\theta^{t^T} x^i)x^i}$
t+=1

3. 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.

The stochastic gradient descent uses a small subset of data at each iteration, unlike in gradient descent where the full data set is used, to compute the gradient.

Initialize θ^0 and number of max iterations. t = 1
i = 0

while $||\theta^{t+1} - \theta^t|| > \epsilon$ and t < max_iterations:

select subset of data, S_i and compute $\theta^{t+1} using$:
$\theta^{t+1} = \theta^t + \gamma_t \Sigma_i (y^j - 1)x^i + \frac{exp(\theta^{t^T} x^i)x^i}{1+exp(\theta^{t^T} x^i)x^i}$

i+=1
t+=1

4. We will show that the training problem in basic logistic regression problem is concave. Derive the Hessian matrix of l(θ) 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.

$l(\theta)$ is concave if $\nabla^2 < 0$ (the Hessian is negative semidefinite).
$l(\theta) = \Sigma_{i=1}^{m} -log(1 + e^{-\theta^T x^i}) + (y^j - 1)\theta^T x^i$
$\frac{\partial l(\theta)}{\partial \theta} = \Sigma_{i=1}^{m} \frac{x^i}{1+e^{x^i \theta^T}} + x^i(y^j - 1)$
$\frac{\partial^2 l(\theta)}{\partial \theta^2} = \Sigma_{i=1}^{m} - \frac{x^{i2} e^{\theta^T x^i}}{(e^{\theta^T x^i} + x^i)^2} < 0$, thus $l(\theta)$ is concave.

Since this is a concave function, there only exists a single global optimum and no local maximums. By using gradient descent, we can take small steps in the direction of the negative gradient. When the gradient is 0 or is very small, we know that we have arrived at the unique global optimizer.

2. Comparing Classifiers

2.1 Divorce Classification

Part A

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.

1. Logistic Regression
    - Train Accuracy = 1
    - Test Accuracy = 0.9412
2. KNN, k=5
    - Train Accuracy = 0.9779
    - Test Accuracy = 0.9706
3. Naive Bayes
    - Train Accuracy = 0.9779
    - Test Accuracy = 0.9706

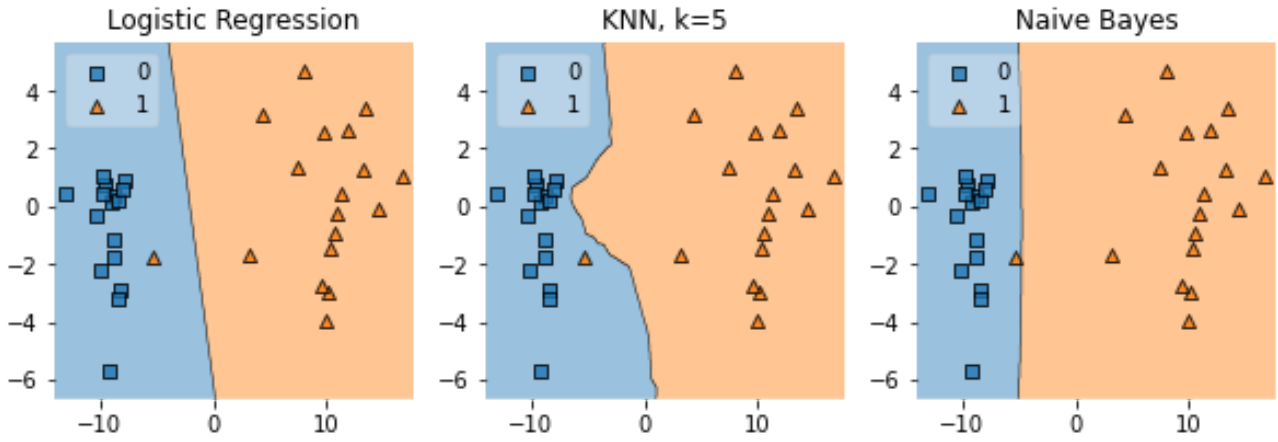
KNN and Naive Bayes peform best on the test data. KNN probably performs well because you'd expect that divorced couples would share some of the same characteristics and thus, would be close 'neighbours' to each other. I didn't expect Naive Bayes to work well because it assumes all features are independent given the label. Looking at the description of the features, I would say the some of them are dependent on another. For example:

5. The time I spent with my wife is special for us.
6. We don't have time at home as partners.

Part B

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.



1. Logistic Regression
    - Train Accuracy = 0.9779
    - Test Accuracy = 0.9706
2. KNN, k=5
    - Train Accuracy = 0.9853
    - Test Accuracy = 0.9706
3. Naive Bayes
    - Train Accuracy = 0.9779
    - Test Accuracy = 0.9706

Applying PCA, we see an improvement in the test accuracy for Logistic Regression, making it perform equally as well as KNN and Naive Bayes. The decision boundaries for Logistic Regression and Naive Bayes are linear while the KNN decision boundary is irregular and noisy.

2.2 Handwritten Digits Classification

Part A

Report confusion matrix, precision, recall, and F-1 score for each of the classifiers.

```
KNN, k=3
Training Accuracy: 0.9363833333333333
Test Accuracy: 0.9366
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       980
           1       0.90      1.00      0.94      1135
           2       0.97      0.91      0.94      1032
           3       0.92      0.93      0.92      1010
           4       0.95      0.91      0.93       982
           5       0.93      0.91      0.92       892
           6       0.96      0.97      0.97       958
           7       0.94      0.94      0.94      1028
           8       0.97      0.87      0.92       974
           9       0.91      0.93      0.92      1009

    accuracy                           0.94     10000
   macro avg       0.94      0.94      0.94     10000
weighted avg       0.94      0.94      0.94     10000

[[ 966    1    1    0    0    3    7    1    1    0]
 [   0 1130    1    1    1    1    1    0    0    0]
 [  16   32  937    9    5    1    3   21    8    0]
 [   3   11   10  936    1   26    0   10    8    5]
 [   1   22    0    0  896    0    9    1    2   51]
 [   6    8    1   33    3  815   11    3    3    9]
 [   8    5    0    1    6    5  933    0    0    0]
 [   0   35    5    1    4    0    0  964    0   19]
 [  15    9    9   27   11   28    2   12  847   14]
 [   6    8    2   10   20    0    2   19    0  942]]

Logistic Regression
Training Accuracy: 0.9392666666666667
Test Accuracy: 0.9256
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       980
           1       0.96      0.98      0.97      1135
           2       0.93      0.90      0.91      1032
           3       0.90      0.92      0.91      1010
           4       0.94      0.94      0.94       982
           5       0.90      0.87      0.88       892
           6       0.94      0.95      0.95       958
           7       0.93      0.92      0.93      1028
           8       0.88      0.88      0.88       974
           9       0.91      0.92      0.91      1009

    accuracy                           0.93     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.93      0.93      0.93     10000

[[ 955    0    2    4    1   10    4    3    1    0]
 [   0 1110    5    2    0    2    3    2   11    0]
 [   6    9  930   14   10    3   12   10   34    4]
 [   4    1   16  925    1   23    2   10   19    9]
 [   1    3    7    3  921    0    6    5    6   30]
 [   9    2    3   35   10  777   15    6   31    4]
 [   8    3    8    2    6   16  912    2    1    0]
 [   1    7   23    7    6    1    0  947    4   32]
 [   9   11    6   22    7   29   13   10  855   12]
 [   9    8    1    9   21    7    0   21    9  924]]
```

```
SVM
Training Accuracy: 0.9142
Test Accuracy: 0.9121
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       980
           1       0.95      0.98      0.97      1135
           2       0.90      0.90      0.90      1032
           3       0.87      0.90      0.89      1010
           4       0.89      0.92      0.91       982
           5       0.87      0.86      0.87       892
           6       0.94      0.95      0.94       958
           7       0.91      0.92      0.91      1028
           8       0.91      0.84      0.88       974
           9       0.91      0.86      0.88      1009

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000

[[ 959    0    2    2    0   11    5    1    0    0]
 [   0 1116    4    5    0    4    1    3    2    0]
 [  14   13  931   17   16    2   15    9   14    1]
 [   6    3   19  908    2   31    2   11   23    5]
 [   2    4    8    1  908    1   13    8    1   36]
 [  12    3    5   58   10  766   11    4   18    5]
 [   8    4   13    0    9   14  906    0    4    0]
 [   2   13   23    7    7    1    0  942    3   30]
 [  10    8   15   29   17   43    9   10  820   13]
 [   5    9    9   11   46    6    0   45   13  865]]

Kernel SVM
Training Accuracy: 0.9526
Test Accuracy: 0.9545
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.98      0.99      0.98      1135
           2       0.96      0.94      0.95      1032
           3       0.94      0.96      0.95      1010
           4       0.93      0.96      0.95       982
           5       0.95      0.95      0.95       892
           6       0.96      0.96      0.96       958
           7       0.96      0.94      0.95      1028
           8       0.94      0.94      0.94       974
           9       0.94      0.92      0.93      1009

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000

[[ 964    0    2    0    0    8    4    1    1    0]
 [   0 1123    2    3    0    1    3    0    3    0]
 [   8    0  970   10    9    0    7    8   20    0]
 [   0    1    7  968    0   10    1   10   11    2]
 [   1    0    5    0  941    0    5    3    2   25]
 [   3    1    3   20    4  843    8    2    5    3]
 [   9    3    2    0    8   10  924    0    2    0]
 [   1   12   18    3    7    0    0  962    3   22]
 [   3    1    3   13    7   16    6    4  918    3]
 [  10    6    1    9   31    2    0    9    9  932]]
```

```
Neural Networks
Training Accuracy: 0.95985
Test Accuracy: 0.9572
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.98      0.98      0.98      1135
           2       0.96      0.95      0.95      1032
           3       0.95      0.96      0.96      1010
           4       0.95      0.96      0.96       982
           5       0.97      0.93      0.95       892
           6       0.95      0.96      0.96       958
           7       0.97      0.93      0.95      1028
           8       0.95      0.96      0.95       974
           9       0.94      0.95      0.95      1009

    accuracy                           0.96     10000
   macro avg       0.96      0.96      0.96     10000
weighted avg       0.96      0.96      0.96     10000

[[ 963    1    0    0    0    4    6    1    3    2]
 [   0 1112    3    4    0    0    3    2   11    0]
 [  12    2  976    7    7    0   11    9    8    0]
 [   0    0   10  973    0    4    2    8   12    1]
 [   1    0    3    0  944    0    9    2    2   21]
 [   8    1    1   20    2  829   12    3    9    7]
 [   9    3    3    0    7    9  924    0    3    0]
 [   2    9   19    7    7    0    0  960    0   24]
 [   5    1    3    5    6    6    9    5  931    3]
 [   7    6    1   11   16    2    1    4    1  960]]
```

Part B

Comment on the performance of the classifier and give your explanation why some of them perform better than the others.

Test Accuracy Summary

1. Neural Networks = 0.9572
2. Kernel SVM = 0.9545
3. KNN = 0.9366
4. Logistic Regression = 0.9256
5. SVM = 0.9121

The worst performing classifier is SVM; this is likely because the data can't be separated well using linear boundaries. The best performing classifier is the neural network and not far behind is the Kernel SVM. This is likely the case because of the same reason why linear SVM is the worst performing classifier - the data is better classified using non-linear boundaries.

3. Naive Bayes for Spam Filtering

Part 1

Calculate class prior P(y = 0) and 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.
Let 1 = spam, 0 = not spam.

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

i	Phrase	Feature Vector	y
1	million dollar offer	{0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0}	1
2	secret offer today	{1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}	1
3	secret is secret	{2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}	1
4	low price for valued customer	{0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}	0
5	play secret sports today	{1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0}	0
6	sports is healthy	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0}	0
7	low price pizza	{0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}	0

$P(y = 0) = \frac{4}{7}$
$P(y = 1) = \frac{3}{7}$

Part 2

Calculate the maximum likelihood estimates of $\theta_{0,1}, \theta_{0,7}, \theta_{1,8}, \theta_{1,15}$ by maximizing the log-likelihood function above.

$l(\theta_{0,1}, \ldots, \theta_{0,d}, \theta_{1,1}, \ldots, \theta_{1,d}) = \Sigma_{i=1}^{7} \Sigma_{k=1}^{15} x_k^i log(\theta_{y^i,k})$

$max \;\; l(\theta_{0,1}, \ldots, \theta_{0,d}, \theta_{1,1}, \ldots, \theta_{1,d}) = \Sigma_{i=1}^{7} \Sigma_{k=1}^{15} x_k^i log(\theta_{y^i,k})$
such that $\Sigma_{k=1}^{15} \theta_{y^i,k} = 1, y^i = 0, 1$

Add the Lagrangian multiplier:
$max \;\; l(\theta_{0,1}, \ldots, \theta_{0,d}, \theta_{1,1}, \ldots, \theta_{1,d}) = \Sigma_{i=1}^{7} \Sigma_{k=1}^{15} x_k^i log(\theta_{y^i,k}) + \lambda(\Sigma_{k=1}^{15} \theta_{y^i,k} - 1)$

Now take derivative wrt to specific θ and set to 0. I will use $\theta_{0,1}$ for this example:

$\frac{\partial l \theta \ldots}{\partial \theta_{0,1}} = \Sigma_{i=4,5,6,7} \frac{x_1^i}{\theta_{0,1}} + \lambda = 0 \longrightarrow$ need to sum over all x_k^i where $y^i = 0$ or i=4,5,6,7 and k=1.
$-\lambda = \frac{1}{\theta_{0,1}} \Sigma_{i=4,5,6,7} x_1^i$
$\theta_{0,1} = -\frac{\Sigma_{i=4,5,6,7} x_1^i}{\lambda} = -\frac{1}{\lambda}$

I got stuck here solving for λ. It has to be negative in order to cancel out the negative in the numerator. However, I know that the value for $\theta_{0,1}$ is supposed to be the probability of the word k appearing in class c. So if the numerator is the number times of word 1 (secret) appears in sentences with class = 0, I will assume that the denominator is the total number of words that appear in class = 0 in order to finish the question.

Thus, $\theta_{0,1} = \frac{1}{14}$

Following this same logic for the rest of the θ:
$\theta_{0,7} = -\frac{\Sigma_{i=4,5,6,7} x_7^i}{\lambda} = -\frac{1}{\lambda} = \frac{1}{14}$
$\theta_{1,8} = -\frac{\Sigma_{i=1,2,3} x_8^i}{\lambda} = -\frac{1}{\lambda} = \frac{1}{9}$
$\theta_{1,15} = -\frac{\Sigma_{i=1,2,3} x_1 5^i}{\lambda} = 0$

Part 3

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

q_1 = P(y=1|'today is secret') = P('today is secret'|y=1)P(y=1)/(P(today is secret'|y=0)P(y=0) + P(today is secret'|y=1)P(y=1))

q_0 = P(y=0|'today is secret') = P('today is secret'|y=0)P(y=0)/(P(today is secret'|y=0)P(y=0) + P(today is secret'|y=1)P(y=1))

From part 1:
$P(y = 0) = \frac{4}{7}$
$P(y = 1) = \frac{3}{7}$

To calculate P('today is secret'|y=1), need to sum $\theta_{1,7}, \theta_{1,11}, \theta_{1,1}$:
$\theta_{1,7}$ = (# times of word 7 (today) appears in sentences with class = 1)/(# words in sentences with class = 1) $= \frac{1}{9}$
$\theta_{1,11}$ = (# times of word 11 (is) appears in sentences with class = 1)/(# words in sentences with class = 1) $= \frac{1}{9}$
$\theta_{1,1}$ = (# times of word 1 (secret) appears in sentences with class = 1)/(# words in sentences with class = 1) $= \frac{3}{9}$
$\longrightarrow$ P('today is secret'|y=1) = $\frac{4}{9}$

To calculate P('today is secret'|y=0), need to sum $\theta_{0,7}, \theta_{0,11}, \theta_{0,1}$
$\theta_{0,7}$ = (# times of word 7 (today) appears in sentences with class = 0)/(# words in sentences with class = 0) $= \frac{1}{14}$
$\theta_{0,11}$ = (# times of word 11 (is) appears in sentences with class = 0)/(# words in sentences with class = 0) $= \frac{1}{14}$
$\theta_{0,1}$ = (# times of word 1 (secret) appears in sentences with class = 0)/(# words in sentences with class = 0) $= \frac{1}{14}$

$\longrightarrow$ P('today is secret'|y=0) = $\frac{3}{14}$

q_1 = P(y=1|'today is secret') = $\frac{\frac{4}{9}\frac{3}{7}}{\frac{4}{9}\frac{3}{7}+\frac{3}{14}\frac{4}{7}} = \frac{14}{23}$

q_0 = P(y=0|'today is secret') = $\frac{\frac{3}{14}\frac{4}{7}}{\frac{4}{9}\frac{3}{7}+\frac{3}{14}\frac{4}{7}} = \frac{9}{23}$

$\longrightarrow$ Since P(y=1|'today is secret') > P(y=0|'today is secret'), "today is secret" is **spam**.