1. Linear Classification and Nearest Neighbor Classification

Q 1 Data

You will use a synthetic data set for the classification task that you'll generate yourself. Generate two classes with 20 features each. Each class is given by a multivariate Gaussian distribution, with both classes sharing the same covariance matrix. You are provided with the mean vectors (DS1-m0 for mean vector of negative class and DS1-m1 for mean vector of positive class) and the covariance matrix (DS1-cov). Generate 2000 examples for each class, and label the data to be positive if they came from the Gaussian with mean m1 and negative if they came from the Gaussian with mean m0. Randomly pick (without replacement) 20% of each class (i.e., 400 data points per class) as test set, 20% of each class (i.e., 400 data points per class) as validation set set and train the classifiers on the remaining 60% data. When you report performance results, it should be on the test set. Call this dataset as DS1, and submit it with your code. Follow the instructions from Assignment 1 for data submission format.

See files in the hwk2_submit folder

Q 2 GDA

We first consider the GDA model as seen in class given the class variable, the data are assumed to be Gaussians with different means for different classes but with the same covariance matrix. This model can formally be specified as follows:

$$Y \sim \text{Bernoulli}(\pi),\ X \mid Y = j \sim N(\mu j, \Sigma).$$

Estimate the parameters of the GDA model using the maximum likelihood approach

.

(a) For DS1, report the best fit accuracy achieved by the classifier.

For the GDA model, the best accuracy achieved is 95,5%

```
Accuracy: 95.5% --- 2.7976250648498535s seconds ---
```

(b) Report the coefficients learn.

See files in the hwk2_submit folder

Q 3 K-NN

For DS1, use k-NN to learn a classifier. Repeat the experiment for different values of k and report the performance for each value. We will compare this non-linear classifier to the linear approach and find out how powerful linear classifiers can be.

(a) Does this classifier perform better than GDA or worse? Are there particular values of k which perform better? Why does this happen? Use validation accuracy for model selection.

> The classifier K-NN doesn't perform better than GDA. It's even a lot worse.
> With multiple of k (No multiple of 2 because we have 2 classes) tested, the best accuracy achieved is 54.37% with a k of 9. The

```
Accuracy with k=1: 52.33333333333333%
Accuracy with k=3: 53.041666666666664%
Accuracy with k=5: 53.833333333333336%
Accuracy with k=9: 54.37499999999999%
Accuracy with k=13: 53.708333333333336%
```

(b) Report the best fit accuracy achieved by this classifier.

```
Accuracy with k=9: 54.37499999999999%
```

Q 4

Now instead of having a single multivariate Gaussian distribution per class, each class is going to be generated by a mixture of 3 Gaussians. For each class, we'll define 3 Gaussians, with the first Gaussian of the first class sharing the covariance matrix with the first Gaussian of the second class and so on. For both the classes, fix the mixture probability as (0.1,0.42,0.48) i.e. the sample has arisen from first Gaussian with probability 0.1, second with probability 0.42 and so on. Mean for three Gaussians in the positive class are given as DS2-c1-m1, DS2-c1-m2, DS2-c1-m3. Mean for three Gaussians in the negative class are gives as DS2-c2-m1, DS2-c2-m2, DS2-c2-m3. Corresponding covariance matrices are given as DS2-cov-1, DS2-cov-2 and DS2-cov-3. Now sample from this distribution and generate the dataset like question 1. Call this dataset as DS2 and submit it with your code. Follow the instructions from Assignment 1 for data submission format.

> ***See files in the hwk2_submit folder***

Q 5

Now perform the experiments in questions 2 and 3 again, but now using DS2.

1. Estimate the parameters of the GDA model using the maximum likelihood approach.

   (a) For DS2, report the best fit accuracy achieved by the classifier.

   ```
   Accuracy: 51.83333333333333%
   ```

   (b) Report the coefficients learn.

   See files in the hwk2_submit folder

2. Does k-NN classifier perform better than GDA or worse? Are there particular values of k which perform better? Why does this happen?

The K-NN classifier does not perform better than the GDA with this data set. The data are to mixed up due to the 3 Gaussian distribution in each class. The best accuracy achieved is 52.16% (only 0.33% more than the GDA method) with a k of 5. Here, the k doesn't chance a lot the accuracy for the same reason.

```
Accuracy with k=1: 52.166666666666664%
Accuracy with k=3: 51.79166666666667%
Accuracy with k=5: 52.166666666666664%
Accuracy with k=9: 50.0%
Accuracy with k=13: 50.041666666666664%
```

3. Report the best fit accuracy achieved by this classifier.

```
Accuracy with k=5: 52.166666666666664%
```

Q 6

Comment on any similarities and differences between the performance of both classifiers on datasets DS1 and DS2?

For the GDA model, these is a drastic difference in the accuracy for the two datasets. 95.5% for DS1 and 51.8% for DS2. That is almost a 50% difference between the two and it's because the assumption of the density of $P(x \mid c_k)$ is Gaussian is true. (Only one Gaussian)

For the K-NN model. These no significant difference between the two datasets. Both are too close to each other to correctly call the new point with his neighbour.

Part 2 : MNIST Handwritten Digits Classification

Q 1

First we will consider Gaussian Na¨ive Bayes (GNB) model for this task. This is same as the GDA model from the previous question but with two modifications: The covariance matrices are not shared and they are diagonal (Na¨ive Bayes assumption).

(a) Write down the equations for computing the mean and diagonal covariance matrices for the class conditional densities and the prior class probabilities using the maximum likelihood approach.

Mean:

$$\mu_k = \frac{\sum_{i=1}^{n_k} \overrightarrow{x_{k_i}} + \alpha}{n_k + \alpha * f}$$

$\overrightarrow{x_{k_i}}$: $vector\ of\ feature\ of\ class\ k$
μ_k: $mean\ of\ class\ k$
α: $Laplace\ Smoothing\ parameter$
f: $Number\ of\ feature$
n_k: $Number\ of\ data\ of\ class\ k$

Diagonal covariance matrices $\sum$:

$$s_j^2 = \frac{1}{n_k - 1}\sum_{i=1}^{n}(x_{ij} - \mu_{kj})^2 \qquad for\ j\ \in \{1, \dots, p\}$$

s_j^2: $Variance\ squarred\ on\ the\ diagonal\ at\ jj$
μ_{ki}: $mean\ of\ class\ k\ of\ feature\ j$
f: $Number\ of\ feature$
n_k: $Number\ of\ data\ of\ class\ k$

Prior class probabilities:

Using Bayes Theorem:

$$P(C_k \mid X) = \frac{P(X \mid C_k)\, P(C_k)}{\sum_{j=1}^{k} P(X \mid C_j)\, P(C_j)} = \frac{e^{a_k}}{\sum_{j=1}^{k} e^{a_j}}$$

Where:

$$a_k = w_k^T x + w_0$$
$$w_k = \sum^{-1} \mu_k$$
$$w_0 = -\frac{1}{2}\mu_k^T \sum^{-1} \mu_k + \ln P(C_k)$$

SoftMax:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

For numerical stability, a better way to compute the SoftMax is:

$$\sigma(z)_i = \frac{e^{z_i - \max(z_i)}}{\sum_{j=1}^{K} e^{z_j - \max(z_i)}}$$

Using SoftMax (generalization of sigmoid for more than two class), we can find a vector with the different probability of x belonging to each class. (Sum of this vector is 1)

(b) Estimate the parameters of the GNB model from the dataset. Report the best fit accuracy achieved.

With this method, the accuracy archived is 80.58%

```
Accuracy: 80.58%
```

Q 2

2. Use k-NN to learn a classifier. Repeat the experiment for different values of k and report the performance for each value.

(a) Are there particular values of k which perform better? Why does this happen ? Use validation accuracy for model selection.

I look like the more k is growing, the

```
Accuracy with k=1: 96.2%
Accuracy with k=10: 96.0%
Accuracy with k=12: 96.0%
Accuracy with k=15: 96.0%
Accuracy with k=21: 95.8%
Accuracy with k=35: 94.6%
```

(b) Report the best fit accuracy achieved by this classifier.

```
Accuracy with k=1: 96.2%
```

Q 3

3. Compare the performance of GNB and k-NN for this MNIST classification. If one performs better over the other, explain why.

> The k-NN model perform better than the GND. That is because each different number lookalike even if they are write by different people. Furthermore, the data have been preprocessed to all look the same and have the same color and the data has been normalized to be on the same scale. That's why the k-NN model works well. The GND does not give us bad result, they have a good accuracy, but not enough for real use.