# CMPE 544 Pattern Recognition

# Assignment 1

By Arda Melikşah Demir

Codes: https://github.com/Ardameliksah/ImageRecognitionML

Table of Contents

[Expectation Maximization 2](#_Toc195473578)

[Scatter Plot of The Data 2](#_Toc195473579)

[Mean and Covariance Matrix Values 3](#_Toc195473580)

[Mean Values 3](#_Toc195473581)

[Covariance Matrix values 3](#_Toc195473582)

[Gaussian Distributions 4](#_Toc195473583)

[Classification 4](#_Toc195473584)

[Preprocessing 4](#_Toc195473585)

[k-Nearest Neighbor Implementation 5](#_Toc195473586)

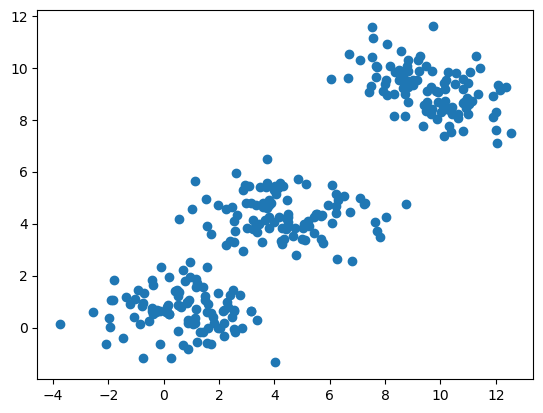
[Naïve Bayes Implementation 6](#_Toc195473587)

[Logistic Regression Implementation 6](#_Toc195473588)

[Comparison of The Models 8](#_Toc195473589)

# Expectation Maximization

## Scatter Plot of The Data



In the scatterplot of the data, we catch the glimpse of three different Gaussian distribution.

## Mean and Covariance Matrix Values

### Mean Values

|  |  |  |
| --- | --- | --- |
|  | Feature 1 | Feature 2 |
| Red | 0.702 | 0.661 |
| Green | 9.605 | 9.168 |
| Blue | 4.379 | 4.352 |

### Covariance Matrix values

|  |  |
| --- | --- |
| **Red Cluster** | |
| 2.118 | -0.0997 |
| -0.0997 | 0.641 |

|  |  |
| --- | --- |
| **Green Cluster** | |
| 2.012 | -0.642 |
| -0.642 | 0.822 |

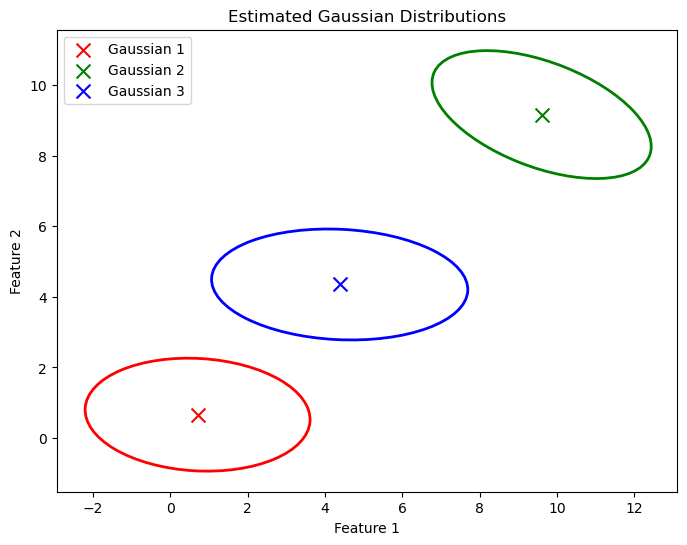
|  |  |
| --- | --- |
| **Blue Cluster** | |
| 2.748 | -0.119 |
| -0.119 | 0.618 |

metin, ekran görüntüsü, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

## Gaussian Distributions

Gaussian distributions with standard deviation until 2 are given in the following graph:



# Classification

## Preprocessing

This task consists of a dataset that has 20000 different 28x28 hand drawn images. The first thing that was done is flattening the 28x28 into 784 length vectors. Then, as each cell has a value that varies between 0 and 255, they were normalized. After normalization two different dimensionality reduction techniques were used. Principal Component Analysis and Linear Discriminant Analysis. Dimensionality reduction techniques were used throughout the problem, distance learning methods were used.

## k-Nearest Neighbor Implementation

During the implementation of k-Nearest Neighbor algorithm, Euclidean distance was used. The accelerated version of kNN algorithm was not used thus the code has a very high time complexity, resulting in a longer running time.

To find the best k value, grid search method was used. In this method 3,5,7,9 neighbor numbers were tested. Also, different preprocessing algorithms were also used with differing neighbor amounts.

Only normalized results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Neighbor (k)** | **3** | **5** | **7** | **9** |
| **Accuracy** | 0.84 | 0.84 | 0.85 | 0.84 |

Linear Discriminant Analysis preprocessing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Neighbor (k)** | **3** | **5** | **7** | **9** |
| **Accuracy** | 0.72 | 0.73 | 0.74 | 0.75 |

Principal Component Analysis preprocessing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dimension Reduction/Neighbor(k)** | **k = 3** | **k = 5** | **k = 7** | **k = 9** |
| **16** | 0.81 | 0.82 | 0.83 | 0.83 |
| **32** | 0.84 | 0.85 | 0.85 | 0.85 |
| **64** | 0.85 | 0.85 | 0.86 | 0.86 |
| **128** | 0.86 | 0.86 | 0.87 | 0.86 |
| **256** | 0.85 | 0.85 | 0.85 | 0.85 |

After the results are observed, it is possible to see that PCA preprocessing works best with kNN algorithm. 7 neighbor and from 784 to 128-dimension reduction using PCA gives the best result for kNN algorithm.

## Naïve Bayes Implementation

Naive Bayes algorithm was implemented with Normal class conditional density assumption. The same preprocessing steps used in KNN were also repeated for Naive Bayes.

|  |  |
| --- | --- |
| Naive Bayes Preprocessing | Accuracy |
| With Just Normalization | 0.57 |
| Linear Discriminant Analysis | 0.74 |

|  |  |
| --- | --- |
| PCA Component Amount | Accuracy |
| 16 | 0.73 |
| 32 | 0.74 |
| 64 | 0.74 |
| 128 | 0.74 |
| 256 | 0.72 |

There isn’t a huge difference between LDA and PCA analysis but for each case it is possible to see preprocessing greatly increases the classification accuracy.

## Logistic Regression Implementation

In this project, two different Logistic Regression in multi-class classification were used. One vs All method and Multi-class with SoftMax method. Multi-class with SoftMax is the most used logistic regression method. In this project, however, both were used to make a comparison between each other. Also, hyperparameters lambda for regularization and learning rate were compared to choosing the most optimal hyperparameters.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression With LDA** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.612 | 0.628 | 0.658 | 0.612 | 0.626 | 0.647 |
| **λ=0.01** | 0.612 | 0.622 | 0.626 | 0.612 | 0.626 | 0.647 |
| **λ=0.1** | 0.612 | 0.612 | 0.612 | 0.612 | 0.626 | 0.647 |
| **λ=1** | 0.612 | 0.612 | 0.612 | 0.612 | 0.626 | 0.647 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression with PCA = 16** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.699 | 0.711 | 0.667 | 0.696 | 0.709 | 0.701 |
| **λ=0.01** | 0.698 | 0.710 | 0.644 | 0.696 | 0.709 | 0.701 |
| **λ=0.1** | 0.692 | 0.693 | 0.517 | 0.696 | 0.709 | 0.701 |
| **λ=1** | 0.651 | 0.651 | 0.422 | 0.696 | 0.709 | 0.701 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression with PCA=32** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.718 | 0.730 | 0.701 | 0.710 | 0.727 | 0.728 |
| **λ=0.01** | 0.717 | 0.726 | 0.665 | 0.710 | 0.727 | 0.728 |
| **λ=0.1** | 0.706 | 0.706 | 0.577 | 0.710 | 0.727 | 0.728 |
| **λ=1** | 0.663 | 0.663 | 0.423 | 0.710 | 0.727 | 0.728 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression with PCA=64** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.724 | 0.740 | 0.722 | 0.719 | 0.737 | 0.740 |
| **λ=0.01** | 0.724 | 0.737 | 0.680 | 0.719 | 0.737 | 0.740 |
| **λ=0.1** | 0.715 | 0.717 | 0.526 | 0.719 | 0.737 | 0.740 |
| **λ=1** | 0.669 | 0.669 | 0.424 | 0.719 | 0.737 | 0.740 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression with PCA=128** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.730 | 0.752 | 0.727 | 0.724 | 0.749 | 0.750 |
| **λ=0.01** | 0.730 | 0.744 | 0.688 | 0.724 | 0.749 | 0.750 |
| **λ=0.1** | 0.718 | 0.719 | 0.550 | 0.724 | 0.749 | 0.750 |
| **λ=1** | 0.675 | 0.675 | 0.424 | 0.724 | 0.749 | 0.750 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression With PCA= 256** | | | | | | |
| **Type** | **Multi-Class** | | | **One vs All** | | |
| **λ/learning rate(lr)** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** | **lr = 0.01** | **lr = 0.1** | **lr = 0.5** |
| **λ=0** | 0.732 | 0.752 | 0.728 | 0.724 | 0.748 | 0.757 |
| **λ=0.01** | 0.730 | 0.745 | 0.687 | 0.724 | 0.748 | 0.757 |
| **λ=0.1** | 0.718 | 0.719 | 0.561 | 0.724 | 0.748 | 0.757 |
| **λ=1** | 0.676 | 0.676 | 0.424 | 0.724 | 0.748 | 0.757 |

As can be seen from the graphs, there is no clear distinction between One vs All and Multi Class on better accuracy. However, the change in regularization term, lambda, has very miniscule effect on One vs All logistic models while increase in lambda decreases the model accuracy significantly in Multi Class model.

## Comparison of The Models

**Non-parametric:** k-Nearest Neighbor. Highest performance: k= 7, PCA = 128 Accuracy: 0.87. Its advantages are that it doesn’t make any assumption about the distribution of the data and learns everything directly from data. However, it comes with high computational costs for this reason. kNN is also very sensitive to irrelevant features however by using LDA or PCA, this problem was overcome.

**Parametric:** Naive-Bayes Highest performance: LDA or PCA = 128, Accuracy: 0.74. Naive Bayes is by far the fastest model out of all the models in this task. Its implementation is also very simple, and the model itself works well with high dimensional data. However, it makes strong assumptions about the distribution of the data.

**Linear Discriminant Function Classifier:** Logistic Regression. Highest Performance: PCA = 256, Learning Rate = 0.5, Accuracy: 0.76. Logistic regression supports regularization which helps prevent overfitting. However, it assumes linear decision boundaries which in this case clearly didn’t work well.

These findings indicate that while linear decision boundaries give fine accuracy ratings, the features are not linearly separable as the best results are given by the non-linear, non-parametric model (kNN). In each case using PCA or LDA greatly reduced computation time and increased accuracy.