**Digit Recognizer – LDA Classifier**

**Author:**

Kyler Cheong

**Table of Contents**

1. **Abstract**
2. **Introduction**
3. **Data**

**3.1. About the Data**

**3.2. Possible Problems**

**3.2.1. Curse of Dimensionality**

**3.2.2. Computationally Intensive**

**3.3. Dimensionality Reduction Technique – Autoencoders**

**3.4. Train and Test Sets**

1. **Methodology**

**4.1. Overview**

**4.2. Linear Discriminant Analysis (LDA) classifier**

**4.2.1. Motivations**

**4.2.2. Assumptions**

**4.2.3. How it works**

**4.2.4. Mathematical Representation**

**4.3. Naïve Bayes Classifier**

**4.4. Quadratic Discriminant Analysis (QDA) classifier**

1. **Results**

**5.1. Assumption Checks**

**5.1.1. Multivariate Normality**

**5.1.2. Equal Covariance Matrices**

**5.1.3. Conclusion**

**5.2. LDA Classifier**

**5.2.1. Scores**

**5.2.2. Evaluation**

**5.3. Naïve Bayes Classifier**

**5.3.1. Scores**

**5.3.2. Evaluation**

**5.4. QDA Classifier**

**5.4.1. Scores**

**5.4.2. Evaluation**

1. **Conclusion**

**6.1. Effectiveness of LDA**

**6.2. Further Improvements**

1. **Abstract**

This project report I will attempt to use Linear Discriminant Analysis (LDA) Classifier to do Image Recognition for handwritten digits. I will use an Autoencoder (Dimensionality Reduction Technique) to compress and shrink the image down. Using the compressed data, I fit it to a LDA classifier and analyze the results, give a summary of the advantages and disadvantages of the LDA classifier, and compare the effectiveness of LDA classifier with other classification methods, namely the Quadratic Discriminant Analysis (QDA) Classifier and Naïve Bayes Classifier. From the results, the LDA classifier performs relatively well with a good accuracy score, serves as a good balance between runtime and accuracy.

1. **Introduction**

In the past 20 years, the amount of data on the internet had exploded exponentially. In the past decade there was a rise in technologies aimed to extract, process, and analyze this data. Image Recognition is one technology that quickly gained popularity among data scientists. Its applications are endless – facial recognition systems, medical x-rays imaging, security systems and autonomous vehicles, just to name a few.

Image Recognition helps save and reduce human effort. If there is a system that can track and recognize suspicious personnel from a Close-Circuit-Television (CCTV), then we could save the need to deploy human manpower to look through CCTV footages.

There had been targeted efforts aimed at improving image recognition systems. Many deep learning models have been created to improve the accuracy of the image recognition. Convolutionary Neural Networks (CNN) are one of the most popular neural networks around for image recognition. However, there is slightly less popular method Linear Discriminant Analysis (LDA) classifiers that works as both a predictive model and a dimensionality reduction technique.

Till this day, handwriting recognition remains an extremely complicated and difficult task for many data scientists. The handwriting recognition technologies are still in its early stages of development. This is evident from our daily lives. Recall the time when you used an iPad or any electronic device that have a stylus, how often does the iPad correctly recognizes the words you wrote? Personally, I find that the number of times it incorrectly predicts my words are uncountably many!

In this report, I want to find out whether LDA classifiers could be used for Image Recognition, specifically, handwritten number recognition. Here, I hope to provide an analytical and statistical study on the advantages and disadvantages of using LDA classifiers and provide some insight to using LDA classifiers for this seemingly complicated handwriting recognition task.

1. **Data**

**3.1. About the Data**

The MNIST dataset is a collection of handwritten digits, numbers from 0, 1, …, 9. It is often used as a benchmark for image recognition tasks. This labelled dataset consists of 42,000 images, each image contains a handwritten digit. The image is in grayscale. Each image is a square image, with height and width of 28 pixels each. In each pixel, it contains a value from 0 to 255. This represents the color brightness of that pixel. A pixel with a value of 0 means that it is completely black, while a pixel with value of 255 means it is completely white. (See Figure 1.)

A picture containing text, clipart

Description automatically generated

Figure 1. Example of an observation in the dataset.

In the MNIST dataset, each image consists of 28 x 28 = 784 pixels, hence, there are a total of 42,000 rows/ images and 784 columns/ pixels and one ‘Label’ column. (See Figure 2.)

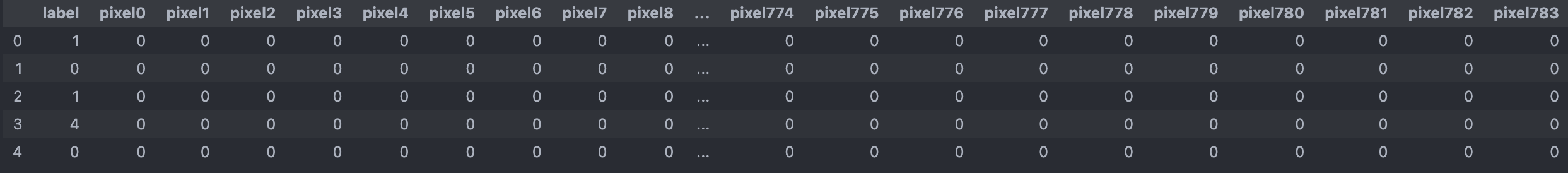


Figure 2. First 5 rows of the dataset.

**3.2. Possible Problems**

The MNIST dataset have a very high dimension, this creates a few problems that make it difficult to work with.

**3.2.1. Curse of Dimensionality**

First, there will be problems with checking if the columns are multivariate normal. This is because most of the values are the same value. Take the number 1 for an example, almost 80% of the image contains the same value. This leads to a singular matrix (shown in Python code file), which means it is not invertible and has a determinant of 0. In a multivariate normal analysis, the assumption is that the covariance matrix of the variables is positive definite, meaning that the determinant of the matrix is non-zero, however, a singular matrix has a determinant of 0. Therefore, this prevents us from checking for multivariate normality between the variables which is detrimental because LDA classifiers capitalizes on the fact that the data is multivariate normal to do its analysis.

**3.2.2. Computationally Intensive**

Second, it will be computationally expensive in terms of time and space because of the large number of dimensions and the large amount data that the model is going to have to process and train on. For LDA classifiers specifically, the runtime of the linear discriminant function to be maximized is linearly related to the number of variables in the dataset. (to be explained in section 4) The smaller the dimension, the faster the runtime. Similar to the argument for the singularity of the covariance matrix of the data, most of the original data (before any dimensionality reduction) would contain many repeated mathematical operations, which results in wasted computational time.

**3.3. Dimensionality Reduction Technique - Autoencoders**

Dimensionality Reduction Techniques are needed to make the covariance matrix non-singular and speed up the overall computation process. The covariance matrix is singular not because the variables are truly singular, but it is because of the way the data is created. The data is represented in 784 pixels for easy interpretation and visualization. Therefore, I used an Autoencoder to shrink the dimensionality down from 784 columns to just 32 columns.

Autoencoders are a dimensionality reduction technique, it is a type of artificial neural network used to learn data encodings in an unsupervised manner. ([link](https://www.v7labs.com/blog/autoencoders-guide#h1)) An autoencoder consists of 3 main parts – input layer, encoded layer and a decoded layer. To explain in the context of the MNIST dataset, this means that the autoencoder takes in the original image in the input layer, pass it through the encoded layer and outputs a reconstructed image in the decoded layer. The model tries to make the reconstructed image as similar as possible to the original image (See Figure 4.). The model tries to minimize the reconstructed image loss. The idea is that it forces the encoded layer to retain the essential information of the input layer, therefore shrinking the dimension of the data.

A picture containing text, clipart

Description automatically generatedA picture containing text, clipart, night sky

Description automatically generated

Figure 4. Left (Original Image- Input layer), Right (Reconstructed Image-Output layer).

Specifically, the autoencoder takes in as input the data for one image, with a shape of (1, 784), i.e. 1 row, 784 columns. Thus, there are 784 nodes (variables) in the first input layer, denoted by X1, … , X783. And the decoded layer consists of 784 nodes as well, denoted by Y0, …, Y783. The number of nodes in the encoded layer is determined by the user. I chose 32 nodes for this project, denoted by E0, …, E31. 32 nodes are used because this is an ideal number of nodes that balances between overfitting and underfitting of the autoencoder. (See Figure 5.)

Diagram

Description automatically generated

Input

Encoded

Decoded

X0

X1

X782­­­

X783­

E0

E31

Y0

Y1

Y782­­­

Y783­

Figure 5. Autoencoder Architecture used for MNIST Dataset.

**3.4. Train and Test Sets**

Using the encoded layer as the new variables, I split the dataset into training and testing set. The training set consists of 80% of the original data while the testing set consist of the remaining 20%. The splitting of the data is done completely by random using the random function in Python. Considering I have 32 variables and 33,600 observations, the ratio of number of observations and number of variables is more than sufficient to draw meaningful conclusion. Furthermore, the frequency count of each digit is roughly the same (see Figure 5.), thus each class is fairly and adequately represented.

1. **Methods**

**4.1. Overview**

I chose three different statistical methods for this task – Linear Discriminant Analysis (LDA) classifier, Naïve Bayes Classifier and Quadratic Discriminant Analysis Classifier (QDA). I scored these methods based on three metrics – Hit rate, Maximum Chance Criterion and Proportional Chance Criterion.

**4.2. LDA Classifier**

**4.2.1. Motivation**

LDA Classifiers are a popular classification technique that is robust and easy to interpret. It is often used as both a dimensionality reduction tool and classification method. LDA classifiers clearly separates the decision space linearly, making it easy to interpret and it is a supervised learning method, which makes it suitable for large datasets like the MNIST dataset.

**4.2.2. Assumptions**

Assumption 1. LDA assumes that the variables are multivariate normal. The joint probability distribution function (PDF) of the 32 variables is normally distributed, following a multivariate Gaussian distribution.

Assumption 2. LDA assumes that the covariance matrices of classes are the same.

Assumption 3. LDA classifier for this dataset assumes a uniform discrete distribution as the prior where each digit has an equal probability of occurring. (See Figure 5).

Chart, bar chart, histogram

Description automatically generated

Figure 5. Frequency count justifies a uniform discrete distribution as choice of prior.

**4.2.3. How it works**

LDA classifiers work by classifying an observation into one of the several pre-defined classes. This is done by maximizing the linear discriminant function and allocating an observation to the class that maximizes this function. The assumption that the covariance matrices are the same allows the discriminant function to be linear in nature and therefore, the decision boundary in the decision space, will separate the regions linearly. Thus, the LDA classifier should split the decision space into 10 different regions, each region corresponding to one of the ten different digits, separated by linear lines.

**4.2.4. Mathematical Representation**

Linear Discriminant Function works by maximizing the log-posterior probability of the classes and will assign the observation **x** to the class *k* that maximizes the probability. In the MNIST dataset, the prior probabilities are assumed to be constant (See Figure 5.).

**Text

Description automatically generated**

Figure 6. Mathematical Representation of LDA Function

For full calculations, see R-code in the R-Markdown file submitted, Section 5.

**4.3. Naïve Bayes Classifier**

Naïve Bayes Classifiers is a popular classification technique which is simple to implement and has a faster runtime than LDA. For this task, I assumed that the variables of the data are normally distributed to provide a fair comparison to the LDA classifier. This assumption was chosen because it provides a known representation of the conditional probabilities and because it is similar to the LDA assumption. Both classifiers assumes that the variables of the dataset are multivariate normal. There are three key differences between the Naïve Bayes and the LDA classifiers.

First, LDA classifiers allows the variables in the dataset to correlate while the Naïve Bayes classifiers assumes independence between the variables. Second, the LDA classifiers assumes that the covariance matrices between class are the same while the Naïve Bayes classifiers assumes that the covariance matrices between classes are different. Thirdly, due to the assumption of independence of the variables for Naïve Bayes Classifiers, it gives rise to the difference in structures of the covariance matrices between the two classifiers. Naïve Bayes classifiers’ covariance matrices are diagonal matrices while LDA classifiers’ covariance matrices are not.

A Bayes Classifier would not be possible because if we were to use the Bayesian approach strictly without the assumptions that the Gaussian Naïve Bayes classifier have, then we must treat the mean vectors and covariance matrices of each class as unknowns. This complicates calculations because we assume no knowledge of the probability distributions of the unknowns. This leaves us with three unknown parameters in the prior distribution– mean, covariance and frequency of the digits. Therefore, I made the above assumption that the Naïve Bayes Classifier to be Gaussian in nature for simplification and ease of understanding.

**4.4. QDA Classifier**

QDA is another frequently used classification technique which is similar to LDA. But unlike LDA, the QDA classifier does not assume Assumption 2 of LDA. QDA assumes that the covariance matrices between the classes are different. Assumption 1 of LDA still holds.

This forces the decision boundary to be quadratic in nature and might be a better option than an LDA if the true decision boundary is non-linear. I wanted to explore if this difference in assumption will make a significant difference in the accuracy of the model.

1. **Results**

**5.1. Assumption Checks**

The MVN test is used to check for multivariate normality while the boxM test is used to check if the covariance matrices are the same. The prior probabilities have already been checked by observing the frequency count of each digit (See Figure 5.)

**5.1.1. Multivariate Normality**

MVN test is set-up with the following hypotheses. Null hypothesis (*H0*) is that the data comes from a multivariate normal distribution and the alternative hypothesis (*HA*) is that the data does not come from a multivariate normal distribution. Low p-values provide evidence against the null hypothesis while high p-values does not.

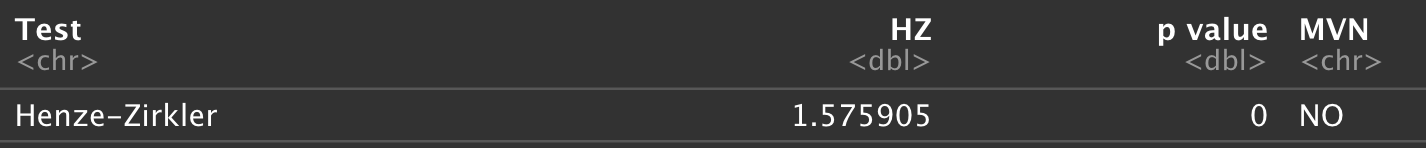


Figure 7. P-value is 0 for the MVN test.

The p-value is 0. There is sufficient evidence to reject the null hypothesis, meaning that the data is not multivariate normal. (See Figure 7.)

**5.1.2. Equal Covariance Matrices**

BoxM test is set-up with the following hypotheses. Null hypothesis (*H0*) is that all the *k* covariance matrices are the same and the alternative hypothesis (*HA*) is that at least one of the *k* covariance matrices is different. Low p-values provide evidence against the null hypothesis while high p-values does not.

Graphical user interface, text

Description automatically generated

Figure 8. P-value is NA for boxM test.

The p-value is NA. This means that there is no conclusion to the test. One possible reason I could think of is that the covariance matrices are not possible to be inverted, due to singularity of the matrices. This seems contradicting to the MVN test because the MVN test arrived at the conclusion, which means that the covariance matrix should be invertible (further supported in the Python file where I checked the determinant of all the covariance matrices). However, considering I do not have full understanding of the nitty-gritty of the underlying code of the 2 tests, there might be a difference in the computation of the covariance matrices or some estimation difference between the 2 tests, which could explain the difference in the results.

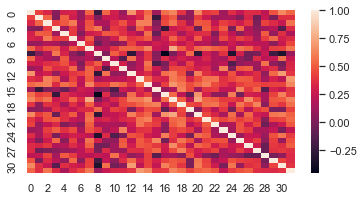
Despite the boxM test being invalid, there are other ways to validify this assumption. The most obvious way is to use intuitive reasoning. Take the number 1 and number 8 for example. I would expect their pixel correlations to be significantly different because if we were to hypothetically overlap these 2 numbers, we would see a large portion of the pixels to be different. Thus, these non-overlapping regions would be different across class 1 and class 8. Thus, favoring the alternate hypothesis.

In addition to an intuitive check, heatmaps of the covariance/ correlation matrices between classes 1, 6 and 8 could be plotted. True enough, there seems to be a significant difference by observation. (See Figure 9 and 10). Notice that the areas of darker colors are similar for number 6 and number 8 (see light green boxes), while the corresponding areas of for that of number 1 is vastly different. Thus the equal covariance matrix assumption is invalid because at least one covariance matrix is differing from the others.

A screenshot of a computer

Description automatically generated with low confidence

Figure 9. Above – Heatmap of correation matrix for number 1.

A screenshot of a computer

Description automatically generated with low confidence

Figure 10. Left – Heatmap of correlation matrix for number 6, Right – Number 8, classes 6 and 8 seems to be more similar to each other than class 1..

**5.1.3. Conclusion**

Therefore, I conclude that the 2 assumptions that LDA classifiers use are both invalid. However, LDA classifiers are robust in a way that even though the assumptions were not met, the classifier could possibly still work well.

**5.2. LDA Classifier**

**5.2.1. Scores**

Text

Description automatically generatedText

Description automatically generated

Figure 11. Left – Training Results, Right – Test Results

A general rule of thumb is that if the LDA classifier is significantly better than a null classifier (25% better), this implies that the LDA classifier could be considered useful. From the results, the hit rate of the LDA classifier on the test set is 86.62%. It performs significantly better than the Maximum Chance Criterion and the Proportional Chance Criterion. (See Figure 11.) This suggests that the LDA classifier is a good classifier for this dataset.

**5.2.2. Evaluation**

The Train and Test set hit rates are close to each other. This means generally means that the model is fitted well and no overfitting has occurred.

The classifier still predicts well despite both the assumptions not fulfilled. One possible reason could be due to Assumption 2 being almost fulfilled. Intuitively, classes 3,4,5,6,8 and 9 are similar and classes 1 and 7 as well. This means that although the covariance matrices between classes are different, but the values of the covariance matrices are mostly close to each other. Thus, almost fulfilling Assumption 2. (See Figure 10. For example of similar covariance matrices)

**5.3. Naïve Bayes Classifier**

**5.3.1. Scores**

**Graphical user interface, text

Description automatically generated with medium confidence**

Figure 12. Results of the Naïve Bayes Classifier

The hit rate of the Naïve Bayes Classifier is 72.68%. Naïve Bayes classifier performs worse than the LDA classifier but still better than a null classifier.

**5.3.2. Evaluation**

The LDA classifier performs better than the Naïve Bayes classifier. It highlights the strength of the LDA classifier in being able to capture the correlations between the variables and use this to generate better accuracy.

However, it should be noted that Naïve Bayes classifier is computationally much more efficient that the LDA classifier because Naïve Bayes classifier assumptions simplifies computation of the covariance matrices, does simple frequency counting and basic numeric operations, while LDA classifiers need to calculate the means and performs vector operations.

However, it is to be noted that should the Gaussian assumption of the Naïve Bayes be replaced by Multinomial Naïve Bayes (because the pixel values could be interpreted as both continuous or discrete in nature), the Multinomial Naïve Bayes classifier will perform significantly better than a Gaussian Naïve Bayes classifier. This shows the importance of the gaussian versus a discrete assumption.

**5.4. QDA Classifier**

**5.4.1. Scores**

Graphical user interface, text

Description automatically generated

Figure 13. Results of the QDA Classifier

The hit rate for the QDA classifier is 96.36%. This is significantly better than the LDA classifier. This accentuates the importance of fulfilling Assumption 2.

**5.4.2. Evaluation**

The difference in QDA classifiers is that the covariance matrices between classes are assumed to be different. This change in assumption caused a huge jump in accuracy in the classification.

However, the QDA classifier is computationally more expensive than LDA classifier. The number of parameters to be estimated for the LDA classifier is linear in nature while QDA classifier is quadratic in nature. In the MNIST dataset, the number of parameters to be estimated in the LDA classifier is 297 parameters, while in QDA, it is 5049 parameters.

Number of parameters to be calculated for LDA is linear while QDA is quadratic. Refer to Table 1. below for comparison. Here, K = 10 and p = 32. Here, I assumed that the priors need to be estimated. The covariance matrix in LDA is the same across all the classes, thus, no need for calculations. The number of terms in the covariance matrix is the sum of the number of elements in the Upper or Lower triangle.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **LDA** | **QDA** |
| K classes | K-1 | K-1 |
| p predictors (mean vector) | p | p |
| Covariance matrix | 0 | (p(p+1))/2 |
| Priors | 1 | 1 |
| Total (Algebraic) | (K-1)\*(p+1) | (K-1)\*(p + p(p+1)/2 + 1) |
| **Total (MNIST)** | **297** | **5049** |

Table 1. Calculation for Number of parameters to be estimated.

1. **Conclusion**

In conclusion, I have explored the possibility of using LDA classifier as a method to do Image Recognition for handwritten digits. I conclude that the LDA classifier could be used as a good Image Recognition Model for handwritten digits. It is a robust classifier because even though the two main assumptions of the LDA classifier is not satisfied, it is still a reasonably strong prediction model.

**6.1. Effectiveness of LDA**

However, the validity of the assumptions could drastically change the accuracy of the model, as seen in the QDA model, highlighting the importance of the assumptions. In addition to the theoretical aspect of the LDA classifier, the practicality of using LDA classifier for real-world applications are also being evaluated. Depending on the classification task at hand, one has to consider the trade-off between accuracy and the computational runtime of the model. If the task requires a very high accuracy, then QDA or LDA classifier might be favored. However, if the runtime is a limiting factor, or a high dimensional dataset is used, then one can consider sacrificing a higher accuracy model for a computationally faster model, like the Naïve Bayes or the LDA classifier. This can be seen when comparing the Naïve Bayes, LDA and QDA classifiers together in terms of runtime and accuracy.

Therefore, I think that LDA classifiers provides a decent balance between accuracy and runtime, which makes it a good classifier for this image recognition task.

**6.2. Further Improvements**

More could be done about the assumptions of LDA. LDA have 2 main assumptions that are restrictive by nature. This means that most datasets out there do not satisfy these assumptions. Many variables in the datasets out there are categorical (discrete) or a mix of both discrete and continuous variables. Thus, it is difficult to use LDA as a classifier. In the context of this handwritten digit recognition, the values could be interpreted as being discrete as well (because it contains values from a fixed interval).

Therefore, other classification methods should be explored. A direct extension of LDA and QDA classifiers are Mixture Discriminant Analysis (MDA) classifiers or Flexible Discriminant Analysis (FDA) classifiers, are able to capture relationships that are otherwise not possible with LDA or QDA. These mixture models usually includes regularization terms which limits and combines the effect of LDA and QDA ([link](http://www.sthda.com/english/articles/36-classification-methods-essentials/146-discriminant-analysis-essentials-in-r/)). (See Figure 14.)

Chart, scatter chart

Description automatically generated

Figure 14. Screenshot from this [website](http://www.sthda.com/english/articles/36-classification-methods-essentials/146-discriminant-analysis-essentials-in-r/). Decision boundary of the classifiers.

Neural Networks like Convolutionary Neural Networks (CNN) are a popular type of Neural Network for Image Classification that could still be explored. Neural Networks have shown to boost accuracy up to 99% and above. They can work well with high dimensional data, even when the dataset is small (through Data Augmentation techniques).