Linear Discriminant Analysis

Nimil Shah
Roll number: 1741048

Rutvi Tilala
Roll number: 1741091

Rajvi Patel
Roll number: 1741078

Kalagee Anjaria
Roll number: 1741052

Hetvi Suthar
Roll number: 1741089

***Abstract*—Information explosion has occurred in most of the sciences and researches due to advances in data collection and storage capacity in last few decades.Traditional statistical method breaks down partly because of the increase in the number of variables. Much of the data is highly redundant which can be ignored,the process of mapping of high dimensional data to lower dimensional space in such a way to discard uninformative variance which can be achieve through PCA and LDA.The objective of LDA is to perform dimensionality reduction , preserving as much of the class discriminatory information as possible.LDA aims to maximize the between class distance and minimize the within class distantce in the dimensionality-reduced space. We do this by transforming the data to a different space that is optimal for distinguishing between the classes. To achieve this goal, LDA utilizes class information to find the best subspace where the ratio of between-class scatter to within-class scatter is maximized. Where the scatter matrices are proportional to the covariance matrices, remains to choose which eigenvalue and eigenvector corresponds to the desired solution.**

Keywords : 1. Dimensionality Reduction 2. Eigenvalues and Eigenvectors 3. Variance between class 4. Variance within class 5. Scatter matrices 6. Classification

I. Introduction About The Problem

Discriminant analysis methods can be good candidates to address such problems. These methods are supervised, so they include label information. The goal is to find directions on which the data is best separable. One of the very well-known discriminant analysis method is the Linear Discriminant Analysis. Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications.

The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (curse of dimensionality) and also reduce computational costs. Pertaining to our problem, we are given a [32 X 32] binary image as input and the goal is to apply LDA technique to transform the features into a lower dimensional space, which maximizes the ratio of the between-class variance to the within-class variance, thereby guaranteeing maximum class separability between two classes in our case with the minimal loss.

II. Literature Review

Dimensionality reduction techniques are important in many applications related to machine learning, data mining, Bioinformatics, biometric and information retrieval.There are two major approaches of the dimensionality reduction techniques and two types of LDA techniques.

- Unsupervised and Supervised reduction technique: There are two major approaches of the dimensionality reduction techniques, namely unsupervised and supervised approaches. In the unsupervised approach, there is no need for labeling classes of the data. While in the supervised approach, the dimensionality reduction techniques take the class labels into consideration. There are many unsupervised dimensionality like ICA and NFM but the most famous technique of the unsupervised approach is the Principal Component Analysis (PCA).This type of data reduction is suitable for many applications such as visualization and noise removal. On the other hand,the supervised approach has many technique Neural Networks (NN) , but the most famous technique of this approach is the Linear Discriminant Analysis(LDA) . This category of dimensionality reduction techniques are used in biometrics Bioinformatics , and chemistry
- Class-Dependent vs. Class-Independent: There are two types of LDA technique to deal with classes: class-dependent and class-independent. In the class-dependent LDA, one separate lower dimensional space is calculated for each class to project its data on it whereas, in the class-independent LDA, each class will be considered as a separate class against the other classes . In this type, there is just one lower dimensional space for all classes to project their data on it .Class-dependent method calculates separate lower dimensional spaces for each class which has two main limitations: (1) it needs more CPU time and calculations more than class-independent method; (2) it may lead to SSS problem because the number of samples in each class affects the singularity of SWi. These reveals that the standard LDA technique used the class-independent method rather than using the class-dependent method.

  Although the LDA technique is considered the most well used data reduction techniques, it suffers from a number of problems. In the rst problem, LDA fails to nd the lower dimensional space if the dimensions are much higher than the number of samples in the data matrix. Thus, the within-class matrix becomes singular, which is known as the small sample problem(SSS). To find LDA , we need to calculate the between-class variance (SB), the separation

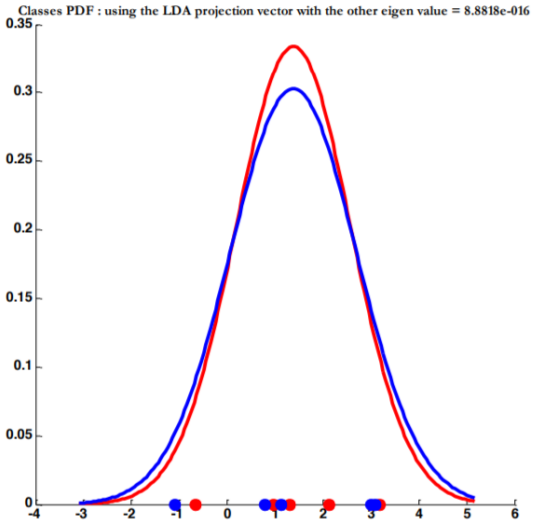
distance between different classes which is denoted by (mi m) where mi represents the projection of the mean of the ith class , mi = WT i, where m is the projection of the total mean of all classes , m = WT , W represents the transformation matrix of LDA, 1 i(1 M) represents the mean of the ith class.

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

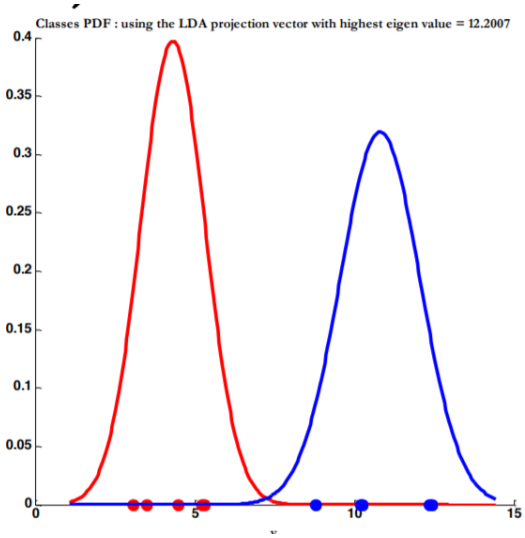
The within-class variance of the ith class (SWi) represents the difference between the mean and the samples of that class. LDA technique searches for a lower dimensional space, which is used to minimize the difference between the projected mean (mi) and the projected samples of each class (WT xi), or simply minimizes the within-class variance. The within-class variance of each class (SWj) is calculated as in the below equation.

$$S_W = \sum_{j=1}^{c} \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T$$

- The graph showing bad seperability. Using eigen vectors of lowest eigen vlues.



- The graph showing good seperability. Using eigen vectors of highest eigen vlues.



III. Results of simulation through scripting and inbuilt Matlab functions

After obtaining two classes of dataset, our goal of finding LDA can be achieved by selecting one line from all possible lines that maximizes the separability of scalars. Our approach comprises of following steps.

Step1: Computing the d-dimensional mean vectors.

Step2: Computing the Scatter Matrices :

2.1 Within-class scatter matrix Sw.

2.2 Between-class scatter matrix Sb.

Step3: Solving the generalized eigenvalue problem for the matrix. Checking the eigenvector and eigenvalue calculation

Step4: Selecting linear discriminants for the new feature subspace by choosing w eigenvector with largest eigenvalues.

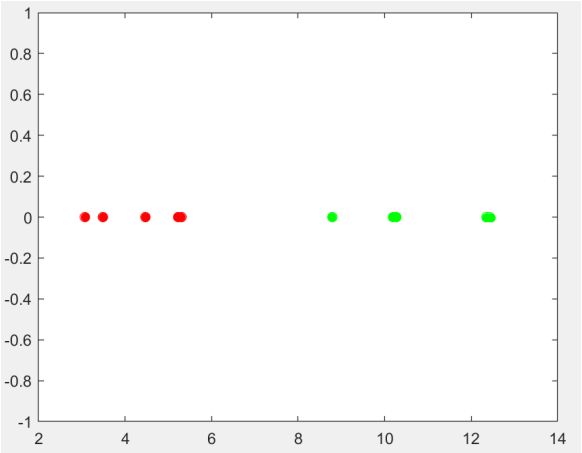
1. $mi = 1/N^{i} \sum_{x \in Di} x^{k}$

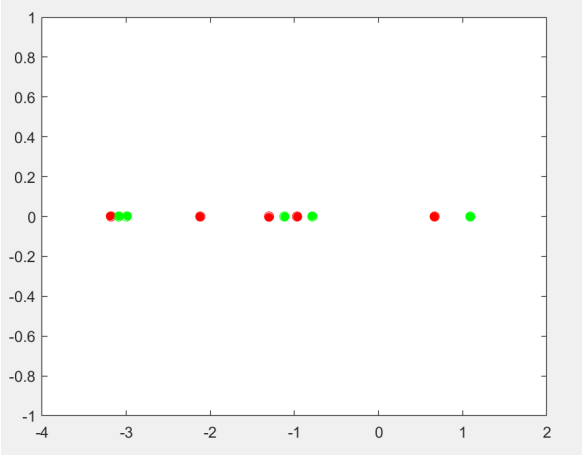
2.1 $S_W = S_1 + S_2$ where $S_i = \sum_{x \in Di} (x - mi)(x - mi)^T$

2.2 $S_B = (m_1 - m_2)(m_1 - m_2)^T$

3. $(Sw^{-1}Sb)\ w = \lambda w$.

When working with the Matlab,built-in functions provides flexibility to direct calculate functions like inverse , transpose , multiplication etc. of the matrix. Thus, we can easily obtain eigenvalues and eigenvectors by using built-in function [V,D]=eig(invSw*Sb). While using our approach, LDA projection is obtained as solution of the generalized eigenvalue problem as mentioned in Step3. There is almost no variance in between the result obtained by using matlab inbuilt function and through our scripting. The eigenvectors obtained by both the approach are different in sign ,but the ratio of eigenvector is always maintained same. Thus, there is no difference between the result obtined by both approaches. The output of our approach provides two eigenvectors, one of which will be optimal projection to the problem. Here are the projection corresponding maximum eigenvalue(optimal) and minimum eigenvalue.



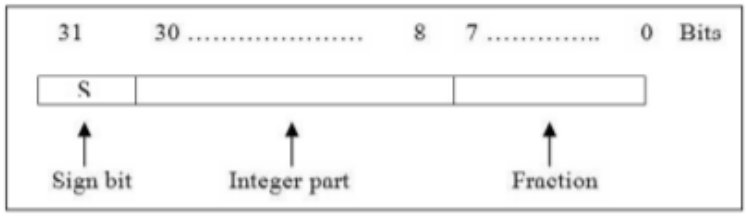


IV. Verilog VHDL Code

Firstly, we tried to convert the Matlab code into verilog using the HDL converter in Matlab. But, there were many errors which we were not able to solve. Thus, we then switched to Xilinx verilog coding. We converted the matlab code into verilog by dividing into different modules based on their functions. But the main problem which was faced here was that the code was such that it only accepted integer values and thus gave integer values as the output. And this was affecting the output as accurate answer was not obtained. So, in order to overcome this problem, integer values must be changed to floating point values. For that, we came across an IEEE format that converted the integer values to floating point. In it, there were two methods given :

Fixed point representation

Normalized floating point reprsentation 1) Fixed point : Divide the 32 bits into 2 parts, one part to represent an integer part of the number and the other the fractional part.



In this figure, we have arbitrarily fixed the (virtual) binary point between bits 7 and 8. With this representation, called fixed point representation, the largest and the smallest positive binary numbers that may be represented using 32 bits are...

Largest: + 111 ..... 1.11111111 = 1677215.998046875
23 bits 8 bits

Smallest: + 000 ... 0.00000001 = 0.00390625
23 bits 8 bits

2) Floating point : This range of real numbers, when fixed point representation is used, is not sufficient in many practical problems. Therefore, another representation called normalized floating point representation is used for real numbers in computers. In this representation, 32 bits are divided into two parts: a part called the mantissa with its sign and the other called the exponent with its sign. The mantissa represents fractions with a non-zero leading bit and the exponent the power of 2 by which the mantissa is multiplied. This method increases the range of numbers that may be represented using 32 bits. In this method, a binary floating point number is represented by (sign) mantissa 2 exponent where the sign is one bit, the mantissa is a binary fraction with a non-zero leading bit, and the exponent is a binary integer. If 32 bits are available to store floating point numbers, we have to decide the following:

1. How many bits are to be used to represent the mantissa (1 bit is reserved for the sign of the number).
2. How many bits are to be used for the exponent.
3. How to represent the sign of the exponent.

IEEE floating point standard : IEEE floating point representation for binary real numbers consists of three parts. For a 32-bit (called single precision) number, they are:

1. Sign, for which 1 bit is allocated.
2. Mantissa (called significant in the standard) is allocated 23 bits.
3. Exponent is allocated 8 bits. As both positive and negative numbers are required for the exponent, instead of using a separate sign bit for the exponent, the standard uses a biased representation. The value of the bias is 127. Thus an exponent 0 means that 127 is stored in the exponent field. A stored value 198 means that the exponent value is (198 127) = 71.

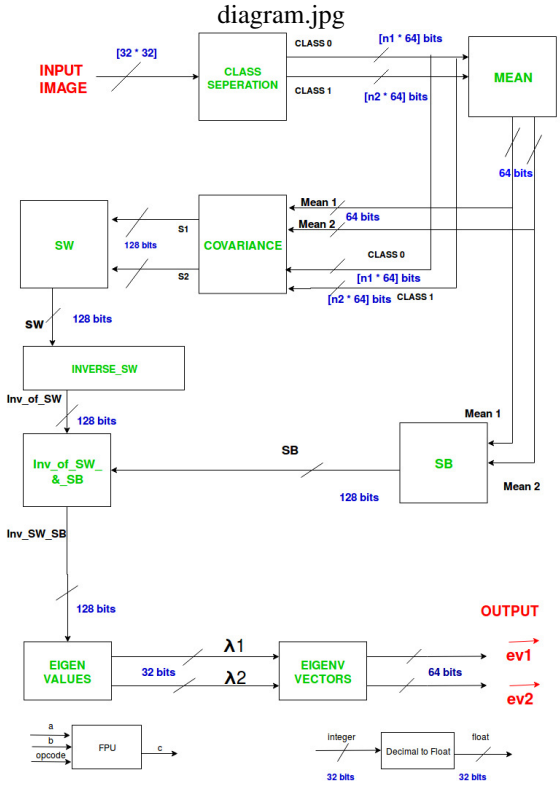
The exponents 127 (all 0s) and + 128 (all 1s) are reserved for representing special numbers.

Newton Raphson division algorithm : Newton-Raphson computational division algorithm computes the multiplicative inverse, which is calculated by the iterative process. Then the calculated multiplicative inverse is multiplied to the dividend to compute the final quotient (Q). In this proposed design, Newton-Raphson computational division algorithm designed by using a 32-floating point multiplier module and subtractor module. In this division algorithm minimal of maximal relative error can be achieved by scaling the divisor (D) in the interval (0.5, 1). Scaling of the divisor is done by shifting operation. To produce a precise result more iterations are required. For this purpose, fast division algorithms are developed.

The calculation of multiplicative inverse of the divisor is performed by equation (1), where Zi is multiplicative inverse at iteration i, as given below. Zi+1 = Zi + Zi (1 DZi) = Zi (2 DZi) (1) Newton-Raphson division algorithm uses a complex initialization for computing continual iterations. To minimize the maximum of the relative error in the interval (0.5, 1), Zi should be initialized. The initialization is represented by Z0 and is initialized as follows. Z0= (48/17) (32/17) D (2)

The following figure is the block diagram of the different modules that we have followed in our verilog code.

diagram.jpg



The adder-subtractor functions are created by us, which takes floating points as input and also gives floating points as the output. We were facing some problems in the division method, therefore we used IP core for implementing that.

IP core : We have used the IP core generator available in Xilnx for the multiplication, division and square root operations.

V. Results Comparision

Matlab

eigen-value 1 : 0.000

eigen value 2 : 12.007

Verilog

eigen value 1 : 0.000

eigen value 2 : 12.001

Time : 414 ns

Matlab

eigen vector 1 : (0.9088, 0.4173)

eigen vector 2 : (0.5755, 0.8178)

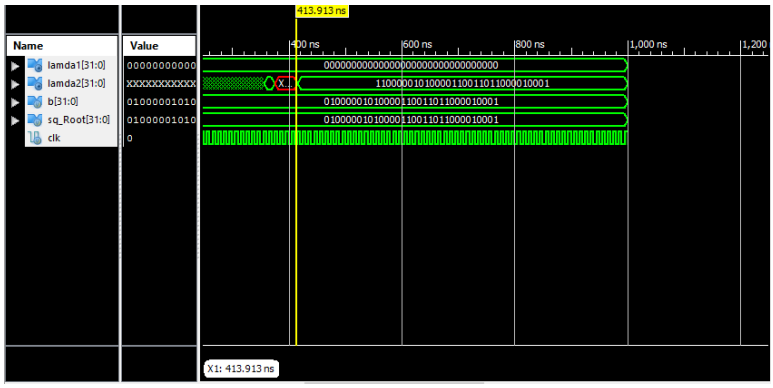
Verilog

eigen vector 1 : (0.9076, 0.4120)

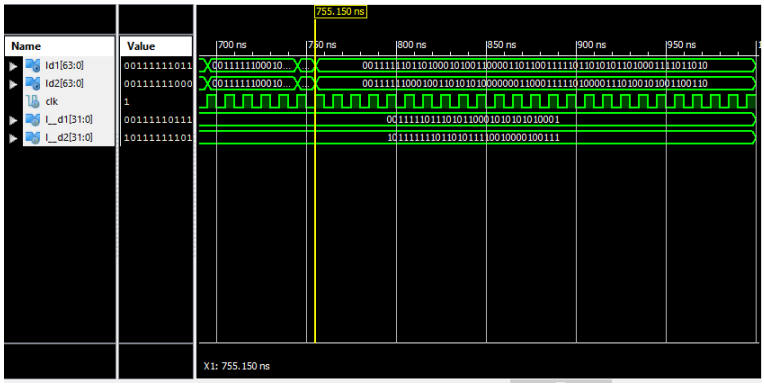
eigen vector 2 : (0.5101, 0.8014)

Time : 755 ns

The verilog behavioral simulation for eigen values.



The verilog behavioral simulation for eigen vectors.



VI. Conclusion

To conclude, LDA works better than PCA with large dataset having multiple classes; class separability is an important factor while reducing dimensionality.Among two LDA methods i.e. class-independent is better than class-dependent method. In this report, result and approaches of implementation of LDA by simulation through our scripting and Matlab inbuilt function were presented and compared. It aimed to give low-level details on how the LDA technique can address the dimensional reduction problem by extracting discriminative features based on maximizing the ratio between the between-class variance(SB),and within-class variance (SW), thus discriminating between different classes. To achieve this aim, we followed the approach of not only simulation through scripting but also through matlab built-in functions. In all, if we compare our experience of working on LDA in matlab with that in Xilinx verilog, then it was found that we faced more problems in Xilinx than in matlab. We found that matlab was more feasible for the matrix operations. It took less time to process all the functions and the output was accurate and precise.

References

[1] https://www.lsv.uni-saarland.de/fileadmin/teaching/dsp/ss15/DSP$_1$5$_L$ec6.pdfb2https : //www.ics.uci.edu/ welling/teaching/273ASpring09/Fisher − LDA.pdf

[2] b3 https://www.andrew.cmu.edu/user/skolouri/Presentations/DiscAnalysis.pdf

[3] b4 http://erepository.uonbi.ac.ke/bitstream/handle/11295/60021/Principal

[4] b5 http://shodhganga.inflibnet.ac.in/bitstream/10603/87031/11/11$_c$hapterb6https : //en.wikipedia.org/wiki/Linear$_d$iscriminant$_a$nalysisLDA$_f$or$_t$wo$_c$lasses

[5] b7 https://www.researchgate.net/publication/302977172$_D$esign$_a$nd$_S$ynthesis$_o$f$_S$ingle$_P$recision$_F$loating$_P$oint$_D$ivision$_b$ased$_o$n$_N$ewton− Raphson$_A$lgorithm$_o$n$_F$PGA

[6] b8 https://www.h-schmidt.net/FloatConverter/IEEE754.html