Linear Discriminant Analysis as a dimensionality reduction technique

Yuanhang Wang, Songyang Cheng, Yifu Zhou
CS 532 Group 9

May 1, 2020

1 Abstract

Dimensionality reduction techniques are important in many applications related to machine learning. Lecture 4.2 covered Principal Component Analysis (PCA), a popular dimensionality reduction technique. This activity will introduce another common dimensionality reduction technique known as Linear Discriminant Analysis (LDA). Both techniques project a high-dimensional dataset onto a lower-dimensional space. However, while PCA preserves the low-dimensional component with the most variance of the data, LDA preserves that with the best class-separability. Moreover, we will also explore LDA as a classification technique.

2 Background

2.1 Basic probability

2.1.1 Normal distribution

The normal distribution is one of the most common distribution of data. It is also known as the Gaussian distribution or the bell curve.
Definition: A continuous random variable X has a (univariate) normal distribution with mean μ and standard variance σ^2 if it has a probability density function of the form

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

This means if $X \sim N(\mu, \sigma^2)$, then the probability of X taking some value x is $p(X = x) = f_X(x)$ defined above.
Figure 1 is a graph of the probability density function of normal distribution:

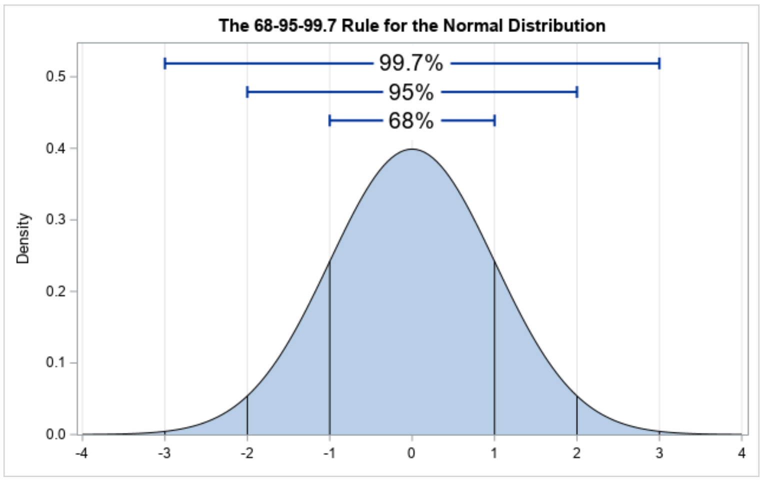


Figure 1: Normal distribution

3 Problem Setting

Suppose we have n data points:
$X = \{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(n)}\}$ with labels $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}$.
Each data point has N features: X can thus be written in a matrix form:

$$X = \begin{bmatrix} x_1^{(1)} & \dots & x_N^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_N^{(n)} \end{bmatrix} \in \mathbb{R}^{n*N}$$

We denote the number of classes by k. I.e. we have k classes and $y_i \in \{0, 1, \dots, k-1\}$ $\forall i \in 1, \dots, n$.
Assumptions:
LDA assumes that the data in each class follows a multi-variate Gaussian distribution in order for its formula to work.

4 Linear Discriminant Analysis with Binary Classes

4.1 Intuitive explanation

Recall that intuitively, LDA wants to project data points into a lower dimensional subspace (we concentrate on 1-dimensional subspace in this section) with best separability of different classes. How do we represent separability?
Intuitively, if multiple classes are better "separated", then the different classes will have their means far away from each other as possible (maximize the difference between means), and the clusters of each class are as tight as possible (minimize the variance within each class).

In figure 2, The projection on the right is better than the projection on the left because the means of the two classes after projection are farther away from each other and the variances are more controlled.

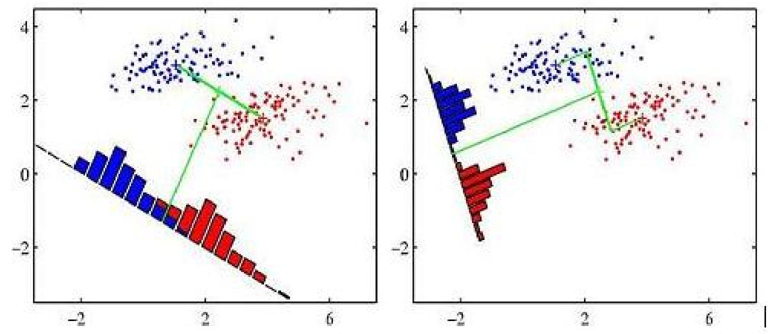


Figure 2: LDA 2-D example

4.2 Mathematical derivation

In this section, We will illustrate the mathematics behind LDA with $k = 2$, i.e. $y \in \{0, 1\}$. Suppose we have n_0 data points of class 0 ($y = 0$), and n_1 data points of class 1 ($y = 1$). Let C_0 be the collection of data points $\vec{x}^{(i)}$ such that $y_i = 0$ and C_1 be the collection of data points $\vec{x}^{(i)}$ such that $y_i = 1$.
Let μ_0 be the mean of the data points in C_0, and μ_1 be the mean of data points in C_1:

$$\vec{\mu_0} = \frac{1}{n_0}\sum_{i=1}^{n_0} \vec{x}^{(i)}$$

$$\vec{\mu_1} = \frac{1}{n_1}\sum_{i=1}^{n_1} \vec{x}^{(i)}$$

Let Σ_0 be the covariance matrix of C_0, and Σ_1 be the covariance matrix of C_1.

$$\Sigma_0 = \frac{1}{n_0 - 1}\sum_{i=1}^{n_0} (\vec{x}^{(i)} - \vec{\mu_0})(\vec{x}^{(i)} - \vec{\mu_0})^T$$

$$\Sigma_1 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1} (\vec{x}^{(i)} - \vec{\mu_1})(\vec{x}^{(i)} - \vec{\mu_1})^T$$

We want to find a vector $\vec{w}$ such that when projecting our data on w, the difference between the two means are maximized and each in-class variance is minimized. We do this by maximize the following formula:

$$\max_{\vec{w}} J(\vec{w}) = \frac{(m_0 - m_1)^2}{(n_0 - 1)s_0 + (n_1 - 1)s_1}$$

where m_i is the mean of data in C_i after projection on $\vec{w}$ and s_i is the variance of data in C_i after projection, $i = 0, 1$.
Note that each data point $x^(i)$ is transformed into $\vec{w}^T x^{(i)}, i = 1, ..., n$. As this transformation is linear, we have

$$m_0 = \vec{w}^T \vec{\mu_0} \quad m_1 = \vec{w}^T \vec{\mu_1}$$

Similarly,

$$s_0 = \frac{1}{n_0 - 1} \sum_{i=1}^{n_0} (\vec{w}^T \vec{x}^{(i)} - \vec{w}^T \vec{\mu_0})(\vec{w}^T \vec{x}^{(i)} - \vec{w}^T \vec{\mu_0})^T = \vec{w}^T \Sigma_0 \vec{w}$$

$$s_1 = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (\vec{w}^T \vec{x}^{(i)} - \vec{w}^T \vec{\mu_1})(\vec{w}^T \vec{x}^{(i)} - \vec{w}^T \vec{\mu_1})^T = \vec{w}^T \Sigma_1 \vec{w}$$

Warm-up exercise 1: Use the fact that the covariance matrix is always positive semi-definite to convince yourself that 1) s_i is a scalar; and 2) s_i is non-negative.

Therefore, we transform the above problem to the following form:

$$\begin{aligned}
\vec{w} &= \arg\max_{\vec{w}} \frac{(\vec{w}^T \vec{\mu_0} - \vec{w}^T \vec{\mu_1})^2}{(n_0 - 1)\vec{w}^T \Sigma_0 \vec{w} + (n_1 - 1)\vec{w}^T \Sigma_1 \vec{w}} \\
&= \arg\max_{\vec{w}} \frac{((\vec{\mu_0} - \vec{\mu_1})^T \vec{w})^2}{\vec{w}^T((n_0 - 1)\Sigma_0 + (n_1 - 1)\Sigma_1)\vec{w}} \\
&= \arg\max_{\vec{w}} \frac{((\vec{\mu_0} - \vec{\mu_1})^T \vec{w})^2}{\vec{w}^T S \vec{w}}
\end{aligned} \tag{1}$$

We write $(n_0 - 1)\Sigma_0 + (n_1 - 1)\Sigma_1$ as a matrix S. The solution of this problem is

$$\vec{w} = \alpha S^{-1}(\vec{\mu_0} - \vec{\mu_1})$$

where α is a non-zero scalar. As we are only interested in a "direction", the value of α does not matter. This optimization problem can be solved using some tricks or Lagrange multiplier, but we'll not give a detail proof here, but you can find an intuitive proof in Appendix C (optional). Also note that we assume that there are more data than features and the features are linearly independent. This guarantees that S is non-singular and so the solution is well-defined.

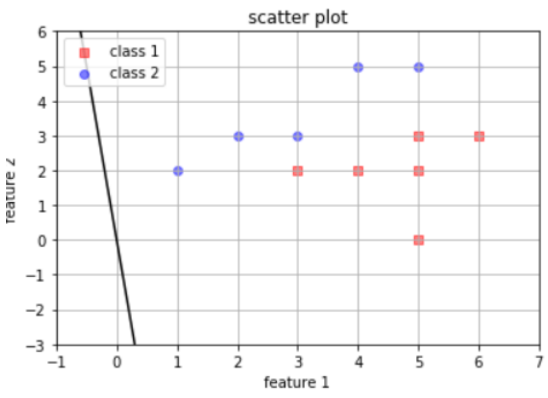
4.3 Activity 1

In this activity, you will do LDA on 2 classes. Here's the data:

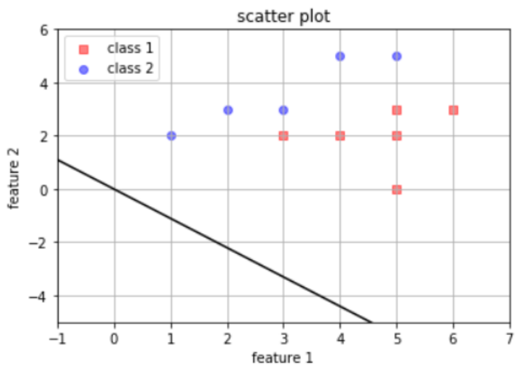
	$(x_1^{(i)}, x_2^{(i)})$	y
$x^{(1)}$	(1,2)	0
$x^{(2)}$	(2,3)	0
$x^{(3)}$	(3,3)	0
$x^{(4)}$	(4,5)	0
$x^{(5)}$	(5,5)	0
$x^{(6)}$	(4,2)	1
$x^{(7)}$	(5,0)	1
$x^{(8)}$	(5,2)	1
$x^{(9)}$	(3,2)	1
$x^{(10)}$	(5,3)	1
$x^{(11)}$	(6,3)	1

Activity question 1:

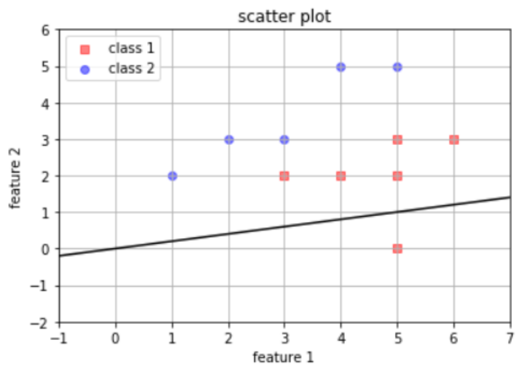
(a) For the following 4 subspaces, which one is more likely to be a the result of LDA on the 11 data points given? *Hint*: No calculations needed. Which line would best separate the 2 classes if the points were projected onto the line?



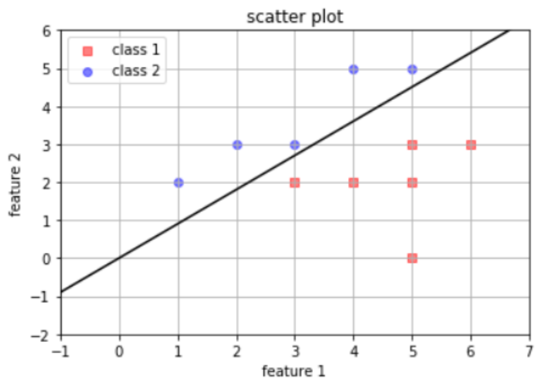
(a) $\vec{w} = (-10, 1)$



(b) $\vec{w} = (-1.1, 1)$



(c) $\vec{w} = (0.2, 1)$



(d) $\vec{w} = (0.9, 1)$

Table 1: Question 1 (a)

(b) Calculate the mean of each class $(\vec{\mu_0}, \vec{\mu_1})$
(c) Verify your choice in (a) by calculating and comparing the value of $J(\vec{w})$ for each of the 4 options. We have calculated the matrix S for you:

$$S = \begin{bmatrix} 15.33 & 9 \\ 9 & 13.2 \end{bmatrix}$$

You may find the hand calculation hard. We have provided several lines of code for you to fill in to calculate these quantities. To access this code, go to ????????????(CAE address).

5 Linear Discriminant Analysis with Multiple Classes

5.1 LDA with multiple classes

In this section, we will go over the general LDA for multiple classes (of course it also works for binary classes). We will introduce several new terms, namely Between-Class Variance (S_B), Within-Class Variance (S_W) and Fisher's criteria.
Suppose we have k classes. Let C_j be the collection of data points $\vec{x}^{(i)}$ such that $y_i = j$, $j = 1, \ldots, k$. Let n_j be the number of data points of class j.
Let μ_j be the mean of the data points in C_j, and μ be the mean of all data points:

$$\vec{\mu_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} \vec{x}^{(i)}$$

$$\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}^{(i)}$$

Definition: The **Between-Class Variance** matrix of the j^{th} class (S_{B_j}) is defined as the distance between the mean of j^{th} class ($\vec{\mu_j}$) and the total mean ($\vec{\mu}$):

$$S_{B_j} = (\vec{\mu_j} - \vec{\mu})(\vec{\mu_j} - \vec{\mu})^T$$

The total Between-Class Variance S_B is the sum of the Between-Class Variance of all classes:

$$S_B = \sum_{j=0}^{k-1} n_j S_{B_j}$$

Definition: The **Within-Class Variance** matrix of the j^{th} class (S_{W_j}) is defined as the sum of the distances between each data point in C_j and the mean of the j^{th} class ($\vec{\mu_j}$).

$$S_{W_j} = \sum_{\vec{x}^{(i)} \in C_j} (\vec{x}^{(i)} - \vec{\mu_j})(\vec{x}^{(i)} - \vec{\mu_j})^T$$

The total Within-Class Variance S_B is the sum of the Within-Class Variance of all classes:

$$S_W = \sum_{j=0}^{k-1} S_{W_j}$$

Definition: The **Fisher's criterion** is defined as

$$\arg\max_{\vec{w}} \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$$

where $\vec{w}$ is the 1-dimensional subspace we are interested in. To obtain a r-dimensional subspace for best projection, we take the best r $\vec{w}$'s and they form a r-dimensional space.

Intuitively, LDA wants to find the low-dimensional space that maximize the Between-Class Variance and minimize the Within-Class Variance after projection. Figure 3 gives you a sense of what the above definitions are. It also shows the best 2 $\vec{w}$s. It turns out that they are eigenvectors of $S_W^{-1}S_B$, which we will explain later.

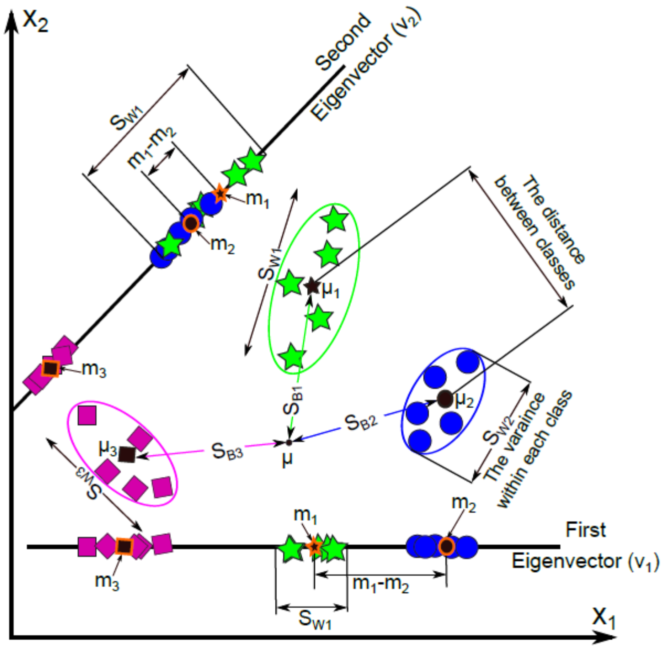


Figure 3: LDA with multi-class

5.2 Mathematic derivation

5.2.1 understanding the Fisher's criterion

Let's take a closer look at the Fisher's criterion:
Let m_j denote the mean of C_j after projection and m be the mean of all data points after projection.
(1) The numerator:

$$\begin{aligned}\vec{w}^T S_B \vec{w} &= \sum_{j=0}^{k-1} \vec{w}^T(\vec{\mu_j}-\vec{\mu})(\vec{\mu_j}-\vec{\mu})^T\vec{w} \\ &= \sum_{j=0}^{k-1}(\vec{w}^T\vec{\mu_j}-\vec{w}^T\vec{\mu})^2 = \sum_{j=0}^{k-1}(m_j-m)^2\end{aligned} \tag{2}$$

As we are trying to maximize the numerator in order to maximize the Fisher's criterion, this is essentially the same as maximizing the difference of the mean of each class after projection.

(2) The denominator:

$$\begin{aligned}
\vec{w}^T S_W \vec{w} &= \vec{w}^T \sum_{j=0}^{k-1} S_{W_j} \vec{w} \\
&= \sum_{j=0}^{k-1} \sum_{\vec{x}^{(i)} \in C_j} \vec{w}^T (\vec{x}^{(i)} - \vec{\mu_j})(\vec{x}^{(i)} - \vec{\mu_j})^T \vec{w} \\
&= \sum_{j=0}^{k-1} \sum_{\vec{x}^{(i)} \in C_j} (\vec{w}^T \vec{x}^{(i)} - \vec{w}^T \vec{\mu_j})^2 \\
&= \sum_{j=0}^{k-1} \sum_{\vec{x}^{(i)} \in C_j} (\vec{w}^T \vec{x}^{(i)} - m_j)^2
\end{aligned} \tag{3}$$

As we are trying to minimize the denominator in order to maximize the Fisher's criterion, this is essentially the same as minimizing the variance of each class after projection.

5.2.2 Solution to the Fisher's criterion (optional)

An key property of the Fisher's criterion is that the objective function $J = \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$ does not change if we scale the vector $\vec{w}$ by a scalar. Therefore, we can always pick $\vec{w}$ such that $\vec{w}^T S_W \vec{w} = 1$. We can thus reformulate the problem to the following form:

$$\begin{aligned}
\min_{\vec{w}} \quad & -\frac{1}{2} \vec{w}^T S_B \vec{w} \\
\text{s.t.} \quad & \vec{w}^T S_W \vec{w} = 1
\end{aligned} \tag{4}$$

This optimization problem can be solved using Lagrange multiplier. We will not go over this in detail here, but just provide you with the solution. The solution of this problem is the eigenvector of matrix $S_W^{-1} S_B$ that corresponds to the greatest eigenvalue. To construct the best r-dimensional subspace, take the r eigenvectors that correspond to the largest r eigenvalues.

5.2.3 Activity 2 Multi-class LDA implementation

We will walk you through the steps of LDA for multiple classes. We have provided a code sample. In Activity 2, you are provided a generated (fake) data set of 100 data points with 3 classes and 3 features. We have provided the generator code so you can generate and play around with new data after you complete this activity.

Activity question 2

Complete the script provided to finish the following steps.
Step 1: Calculate the mean of each class (μ_j) and the mean of all data μ
Step 2: Calculate the Between-Class Variance matrix S_B using the formula

$$S_B = \sum_{j=0}^{k-1} n_j(\vec{\mu_j} - \vec{\mu})(\vec{\mu_j} - \vec{\mu})^T$$

Step 3: Calculate the Within-Class Variance matrix S_W using the formula

$$S_W = \sum_{j=0}^{k-1} \sum_{\vec{x}^{(i)} \in C_j} (\vec{x}^{(i)} - \vec{\mu_j})(\vec{x}^{(i)} - \vec{\mu_j})^T$$

Step 4: Find the eigenvectors and eigenvalues of matrix $S_W^{-1} S_B$.
Step 5: Sort the eigenvectors by decreasing eigenvalues, and select top r eigenvectors to construct a r-dimensional subspace
Step 6: Transform the data points onto the new subspace.

This completes our discussion on LDA. In the next section, we will give a very simple comparison between LDA and PCA as dimensionality reduction techniques.

6 LDA vs PCA

In this section, we will give a simple comparison between 2 commonly used dimensionality reduction method: Linear Discriminant Analysis (LDA) and Principle Component Analysis(PCA).

We have covered PCA in lecture 4.2. PCA aims to find a lower dimensional subspace such that the variance is maximized when the data is projected onto the subspace. PCA does not care about the labels of the data (y), so it can be thought of a ”unsupervised” dimensionality reduction technique.

LDA also aims to find a lower dimensional subspace, but it emphasize on the separability of different classes. Therefore, LDA takes the labels of the data into account, and can be thought of a ”supervised” dimensionality reduction technique.

Both techniques are commonly used as data preprocessing steps of a machine learning task, aiming to eliminate redundant features. Normally, when you encounter a classification problem, most of the time LDA will be a better way to do feature selection. In Activity 3, you will see the performance of LDA and PCA on a classification task on a real data set. However, there are sometimes when PCA is better than LDA even in a classification problem, especially when sample sizes are small.

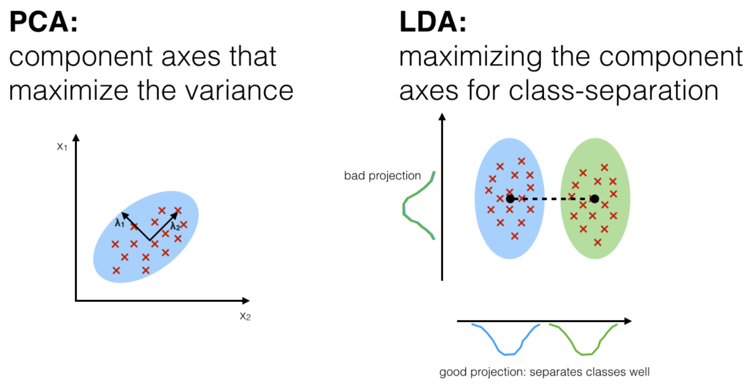

PCA:
component axes that maximize the variance
x₁
x₂
λ₁
λ₂
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Figure 4: PCA vs LDA

6.1 Activity 3

In this activity, we want to compare the performance of different dimensionality techniques on classification tasks. The data we are using is the Iris dataset, we projected the data to the best 2-dimensional space indicated by different dimensionality reduction techniques, and used Logistic Regression as the classification algorithm.

Activity question 3

Run the given script. In this example, which method works better?

7 Further Remarks

7,1 Problems of LDA

Linearity problem

One huge problem of LDA is that it does not work well on data sets that are not linear separable. An extreme example will be as follows: Consider a data set with 2 classes and 3 features. class 0 has data of form $x_1^2+x_2^2+x_3^2 = 1$, and class 1 has data of form $x_1^2+x_2^2+x_3^2 = 2$. In this example LDA will not generate good linear discriminants. However, this problem can be solved using kernel methods just like what we learned in lecture 6.3. We will not cover details of Kernel Fisher discriminant analysis. Here is a link if you are interested: Kernel Fisher discriminant analysis on Wikipedia.

Small sample size problem

Another huge problem for LDA is Small Sample Size (SSS) problem. This problem results from a low number of training samples available for each class compared to the dimensionality of the sample space. In such situation, the matrix S_W may be non-invertible. This often occurs in image classification problems, where each pixel represents a dimension. A common solution to this problem is to add regularization term. Another common solution is to first use PCA to reduce the high-dimensional features to relatively low-dimensional features, and then use LDA to further reduce the dimension of the features.

7.2 Notes on LDA classifier

Note that LDA can also be used as a classifier, but we are only talking about LDA as a dimensionality reduction tool in this tutorial. There are some differences between LDA and LDA classifier, but as they are both developed by Sir Ronald Fisher in 1936, the two terms are used interchangeably.

Appendix

Appendix A. Solution to warm-up exercise

Solution to Warm-up exercise 1:

$$s_0 = \frac{1}{n_0 - 1}\sum_{i=1}^{n_0}(\vec{w}^T\vec{x}^{(i)} - \vec{w}^T\vec{\mu_0})(\vec{w}^T\vec{x}^{(i)} - \vec{w}^T\vec{\mu_0})^T = \vec{w}^T\Sigma_0\vec{w}$$

As $\vec{w}$ is a $N * 1$ vector, Σ_0 is a $N * N$ matrix, so $\vec{w}^T\Sigma_0\vec{w}$ is well defined and is a scalar. Moreover, as Σ_0 is positive semi-definite, s_0 is always non-negative. In fact, s_0 is an analogous to "variance".

Appendix B. Solution to activity questions

Activity 1:

(a) B
(b) $\vec{\mu_0} = [3, 3.6]; \vec{\mu_1} = [4.67, 2]$
(c) $J_A = 0.247, J_B = 1.029, J_C = 0.103, J_D = 0.001$

Activity 2:

See the attached code solution.

Activity 3:

The LDA classifier works better than the others. See the attached code solution.

Appendix C. Solution to the optimization problem in section 4.2

In this section, we will show you the solution to the optimization problem in section 4.2 using some tricks.

$$\vec{w} = \arg\max_{\vec{w}}\frac{((\vec{\mu_0} - \vec{\mu_1})^T\vec{w})^2}{\vec{w}^TS\vec{w}}$$

As S is positive definite, we can find a invertible (full-rank) square R such that $S = R^T R$. Also, denote $\vec{m} = \vec{m_0} - \vec{m_1}$

$$\begin{aligned}\vec{w} &= \arg\max_{\vec{w}} \frac{((\vec{\mu_0} - \vec{\mu_1})^T \vec{w})^2}{\vec{w}^T S \vec{w}} \\ &= \arg\max_{\vec{w}} \frac{(\vec{m}^T \vec{w})^2}{\vec{w}^T R^T R \vec{w}}\end{aligned} \tag{5}$$

Write $\vec{v} = R\vec{w}$. Since R is invertible, $\vec{w} = R^{-1}\vec{v}$. Then we have:

$$\begin{aligned}\frac{(\vec{m}^T \vec{w})^2}{\vec{w}^T R^T R \vec{w}} = \frac{(\vec{m}^T R^{-1} \vec{v})^2}{\vec{v}^T \vec{v}} &= \frac{(\vec{m}^T R^{-1} \vec{v})^2}{|\vec{v}|^2} \\ &= (\vec{m}^T R^{-1} \frac{\vec{v}}{|v|})^2 \\ &= (((R^{-1})^T \vec{m})^T \frac{\vec{v}}{|v|})^2\end{aligned} \tag{6}$$

We want to maximize the quantity above, which is the squared inner product of a vector $((R^{-1})^T\vec{m})^T$ and a unit vector $\frac{\vec{v}}{|v|}$. Basically, we want to find the unit vector $\frac{\vec{v}}{|v|}$ that maximize the above quantity since $((R^{-1})^T\vec{m})^T$ is independent of $\vec{w}$. The maximum is reached when the vector $\frac{\vec{v}}{|v|}$ has the same direction with $(R^{-1})^T\vec{m}$, i.e.

$$\vec{v} = \alpha(R^{-1})^T \vec{m} = \alpha(R^{-1})^T(\vec{\mu}_0 - \vec{\mu}_1)$$

for some scalar α. Then, we have

$$\vec{w} = R^{-1}\vec{v} = \alpha R^{-1}(R^{-1})^T(\vec{\mu}_0 - \vec{\mu}_1) = \alpha S^{-1}(\vec{\mu}_0 - \vec{\mu}_1)$$

which is the same formula as what we give in section 4.2.

References

[1] Tharwat, Alaa & Gaber, Tarek & Ibrahim, Abdelhameed & Hassanien, Aboul Ella. (2017). *Linear discriminant analysis: A detailed tutorial.* Ai Communications. 30. 169-190,. 10.3233/AIC-170729.

[2] Welling, Max. (2009). *Fisher Linear Discriminant Analysis.*
https://www.ics.uci.edu/w̃elling/teaching/273ASpring09/Fisher-LDA.pdf.

[3] Shadmehr, Reza. (2014). JHU Learning Theory Lecture 22.
https://www.youtube.com/watch?v=T26-kN54gJY

[4] Raschka, Sebastian. (2014). Linear Discriminant Analysis.
https://sebastianraschka.com/Articles/2014_python_lda.html#step-3-solving-the-generalized-eigenvalue-problem-for-the-matrix-s_w-1s_b.

[5] Comparison of LDA and PCA 2D projection of Iris dataset.
https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html