Fisher's Linear Discriminant

Jianxin Wu

LAMDA Group
National Key Lab for Novel Software Technology
Nanjing University, China
wujx2001@gmail.com

April 6, 2017

Contents

1 Introduction & Motivation

The topic of this note is Fisher's Linear Discriminant (FLD), which is also a linear dimensionality reduction method. FLD extracts lower dimensional features utilizing linear relationships among the dimensions of the original input.

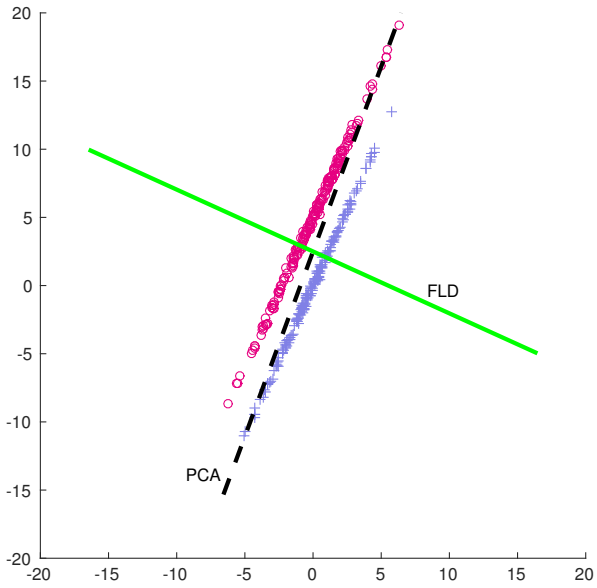


Figure 1: FLD vs. PCA.

A natural question is: what makes FLD different from PCA, and why do we still need FLD when we have PCA handy?

A short answer is: FLD is supervised, but PCA is unsupervised. We illustrate their differences using a somehow extreme example in Figure 1.

Suppose we are to deal with a binary classification problem. We are given a set of input vectors $\boldsymbol{x} \in \mathbb{R}^D$ (which is two dimensional in Figure 1), and need to classify them into two classes. The class label y can take value in the set $\{1, 2\}$. In Figure 1, we are given 200 examples with $y = 1$ and 200 examples with $y = 2$. These 400 examples form our training set for this binary classification problem. The positive examples (with $y = 1$) are denoted by the $\circ$ sign, and negative examples ($y = 2$) are denoted by the $+$ sign in Figure 1. Examples in different classes are also shown in different colors.

In this example, the two classes have special properties: the inherent dimensionality of both seem to be 1. In fact, these data points are generated using the following Matlab command:

```
n1 = 200;
n2 = 200;
rot = [1 2; 2 4.5];
data1 = randn(n1,2) * rot;
data2 = randn(n2,2) * rot + repmat([0 5],n2,1);
```

In other words, we generate two very thin Gaussians, and shift one of them to

separate them apart.

Visual inspection tells us that it is easy to classify examples into two classes if we project any $\boldsymbol{x}$ into the projection direction labeled "FLD" (the green solid line). After the projection, the positive class examples will clump into a small range of values, so do the negative examples. However, the range of projected values in different classes almost never overlap with each other. That is, a proper linear dimensionality reduction makes our binary classification problem trivial to solve.

However, if we compute the PCA solution using all 400 training examples, the first eigenvector generates the black dashed line (with a label "PCA" in Figure 1). The PCA projection direction is surprisingly bad for the classification. The ranges of projection values are almost identical for the positive and the negative class, that is, this direction is almost useless for classification.

What makes PCA so unsuitable in this example? PCA is an unsupervised method, which does not consider any label information. It tries to eliminate the projection direction that corresponds to small variance (which is the FLD projection direction). However, in this example, the labels tell us that it is this specific direction that separates the two classes!

Hence, when we are given labels of examples in a classification task, it is essential to take these labels into consideration when we perform linear dimensionality reduction. FLD is such a supervised dimensionality reduction method, and we will start the introduction of FLD from the binary classification setup.[1]

2 FLD for binary classification

The example in Figure 1 also gives us some hint on how to incorporate the labels. Given any projection direction $\boldsymbol{w}$, we can compute the projected values as $\boldsymbol{w}^T\boldsymbol{x}$, and require that projected values from the positive and the negative classes are *far apart*.

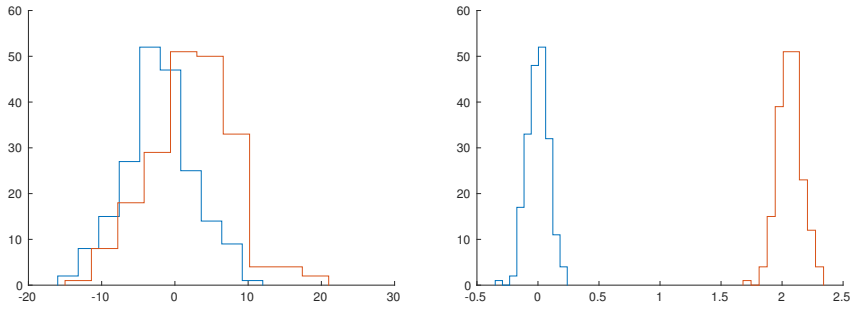
2.1 Idea: what is far apart?

But, how do we translate the phrase *two sets of values are far apart* into precise mathematical expressions? That is, we need a way to measure the level of separation between two sets of values. There are many different ways to compare the distance between two sets of values, and we will introduce only the FLD way in this note.[2]

Figure 2 shows the histogram (with 10 bins) of the projection values for the data in Figure 1, when the PCA or FLD projection direction is used. A plot that shows all the projection values might be too cluttered to reflect any valuable

[1] The example in Figure 1 is a little bit contrived and extremal. In real applications, both PCA and FLD are useful and they can be combined to produce better linear dimensional reduction. PCA+FLD has been a very successful method in face recognition.

[2] I believe the readers of this note can think of different ways on their own.



(a) PCA (b) FLD

Figure 2: Histogram of projected values of the dataset in Figure 1 along the PCA (2a) or FLD (2b) direction. Please note that the scales in the x-axis are *different* in these figures.

information. The histogram succinctly captures the distribution properties of the projection values, and hence is a useful tool in various studies.

Let us first look at the FLD figure (2b). The histograms of both classes are sharp, meaning that the projection values of the positive examples are in a small range, so do those of the negative examples. In this case, we can use the mean of each class to represent that class. And, the level of separation between these two classes can be measured by the distance between the two mean values. In this example, the two mean values are 0.0016 and 2.0654, respectively. The distance between them is 2.0638.

But, if we apply this measure of separation, we will encounter difficulty when we move to the PCA projection direction, whose histograms are shown in Figure 2a. The two mean values are -2.2916 and 2.2916, respectively. The distance, 4.5832, is even larger than that in the FLD case (2.0638). Note that the scale of the x-axis is different in Figure 2b and 2a.

It does not mean the PCA projection direction is better, though. As shown in Figure 2a, the two histograms overlap seriously, meaning that the level of separation is low. This paradox (i.e., PCA has larger distance between projected mean values, but worse separation) appears because the *shape* of the histograms are not taken into account. The FLD histograms is sharp and has small variances (or standard deviations). The distance between mean values is very large compared to *both* standard deviations. Hence, the two histograms do not overlap with each other.

In the PCA case, the distance between mean values is smaller than both standard deviations. Hence, the two histograms significantly overlap with each other, which means low separation ability.

Putting these facts together, it is the ratio between 1) distance between two mean values, and 2) the two standard deviations that decides separability, not the distance alone. We want to maximize this ratio.

2.2 Translation to mathematics

Let us define our notations. One training example for FLD will be a pair $(\boldsymbol{x}_i, y_i)$ instead of a single vector $\boldsymbol{x}_i$, in which $\boldsymbol{x}_i \in \mathbb{R}^D$ is the original input vector that describes the i-th example in the training set, and $y_i \in \{1, 2\}$ is a label that specifies to which class $\boldsymbol{x}_i$ belongs. Hence, a training set X with N such pairs can be written as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, which is an abbreviation for the set $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$.

Because there are two categories, we can collect all positive examples to form a subset of X, as $X_1 = \{\boldsymbol{x}_i | 1 \leq i \leq N, y_i = 1\}$, and $N_1 = |X_1|$ is the size of this subset, which is also the number of positive examples in X. Similarly, we define $X_2 = \{\boldsymbol{x}_i | 1 \leq i \leq N, y_i = 2\}$, and $N_2 = |X_2|$. Note that we do not include the labels y_i in X_1 or X_2, because labels of examples in them can be easily deduced.

The mean of vectors in these subsets are also easy to define, as

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{\boldsymbol{x} \in X_1} \boldsymbol{x}\,, \tag{1}$$

$$\boldsymbol{m}_2 = \frac{1}{N_2} \sum_{\boldsymbol{x} \in X_2} \boldsymbol{x}\,, \tag{2}$$

respectively. Later, we will find these two notations very useful, and it is convenient to define them early. Similarly, the covariance matrices are

$$C_1 = \frac{1}{N_1} \sum_{\boldsymbol{x} \in X_1} (\boldsymbol{x} - \boldsymbol{m}_1)\,(\boldsymbol{x} - \boldsymbol{m}_1)^T\,, \tag{3}$$

$$C_2 = \frac{1}{N_2} \sum_{\boldsymbol{x} \in X_2} (\boldsymbol{x} - \boldsymbol{m}_2)\,(\boldsymbol{x} - \boldsymbol{m}_2)^T\,, \tag{4}$$

respectively.

Given any projection direction $\boldsymbol{w}$, as is in the PCA case, we can safely assume $\|\boldsymbol{w}\| = 1$. Hence, the projected value for $\boldsymbol{x}_i$ is $\boldsymbol{x}_i^T \boldsymbol{w}$. The mean of projected values for the two classes are

$$m_1 = \boldsymbol{m}_1^T \boldsymbol{w}, \quad \text{and} \quad m_2 = \boldsymbol{m}_2^T \boldsymbol{w}\,. \tag{5}$$

The variance for projected positive examples is

$$\frac{1}{N_1} \sum_{\boldsymbol{x} \in X_1} \left(\boldsymbol{x}^T \boldsymbol{w} - \boldsymbol{m}_1^T \boldsymbol{w}\right)^2 \tag{6}$$

$$= \boldsymbol{w}^T \left(\frac{1}{N_1} \sum_{\boldsymbol{x} \in X_1} (\boldsymbol{x} - \boldsymbol{m}_1)(\boldsymbol{x} - \boldsymbol{m}_1)^T \right) \boldsymbol{w} \tag{7}$$

$$= \boldsymbol{w}^T C_1 \boldsymbol{w}\,, \tag{8}$$

and the standard deviation is

$$\sigma_1 = \sqrt{\boldsymbol{w}^T C_1 \boldsymbol{w}}\,. \tag{9}$$

Similarly, the variance and standard deviation for the negative examples are

$$C_2 = \boldsymbol{w}^T C_2 \boldsymbol{w}\,, \qquad \sigma_2 = \sqrt{\boldsymbol{w}^T C_2 \boldsymbol{w}}\,, \tag{10}$$

respectively.

The ratio we want to maximize can be translated into different forms, e.g.,

$$\frac{|m_1 - m_2|}{\sigma_1 + \sigma_2}\,, \tag{11}$$

or

$$\frac{|m_1 - m_2|}{\sqrt{\sigma_1^2 + \sigma_2^2}}\,. \tag{12}$$

Both equations are sometimes referred to as the Fisher's Linear Discriminant by some authors. However, the classic FLD has a slightly different form, which we will introduce soon. We want to, however, point out that the solution of these two equations should also give reasonable projection directions for binary problems, because they share the same intuition with FLD.

2.3 Scatter vs. covariance matrix

Maximizing Equation 12 does not seem convenient. The absolute value and the square root both complicates the optimization. However, we can maximize the alternative objective function

$$\frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}\,, \tag{13}$$

because it is obviously equivalent to Equation 12.

The projection $\boldsymbol{w}$ is only implicitly embedded in Equation 13. Plugging in the expressions for m_1, m_2, σ_1^2 and σ_2^2, we get the objective which is explicit in $\boldsymbol{w}$:

$$\frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T\boldsymbol{w}}{\boldsymbol{w}^T(C_1 + C_2)\boldsymbol{w}}\,. \tag{14}$$

The term $C_1 + C_2$ uses the covariance matrix statistics of X_1 and X_2 to measure how *scattered* they are. The *scatter matrix* is another interesting statistics to measure how scattered a set of points are. The scatter matrix for a set of points $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$ is defined as

$$S = \sum_{i=1}^{n} (\boldsymbol{z}_i - \bar{\boldsymbol{z}})(\boldsymbol{z}_i - \bar{\boldsymbol{z}})^T\,, \tag{15}$$

where $\bar{\boldsymbol{z}}$ is the mean of these points, defined as $\bar{\boldsymbol{z}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{z}_i$.

It is obvious that these two statistics are related to each other. Let us denote the scatter matrix for X_1 and X_2 as S_1 and S_2, respectively. Then,

$$S_1 = N_1 C_1\,, \tag{16}$$

$$S_2 = N_2 C_2\,. \tag{17}$$

They are equivalent to each other, modulo the slight difference in multiplying the number of points or not.

Traditionally, the scatter matrix is used in FLD, not the covariance matrix. That is, the FLD objective is

$$\frac{\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T\boldsymbol{w}}{\boldsymbol{w}^T(S_1 + S_2)\boldsymbol{w}}\,. \tag{18}$$

However, we want to point out the solutions for both forms should be similar to each other qualitatively. When the number of examples in all classes are the same, these two should give the same answer.

2.4 The two scatter matrices and the FLD objective

In FLD (for binary problems), two additional symbols are introduced,

$$S_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T\,, \tag{19}$$
$$S_W = S_1 + S_2\,, \tag{20}$$

in which S_B is called the between-class scatter matrix, and S_W is the within-class scatter matrix (both are $D \times D$ is size). S_W measures how *scattered* is the original input dataset *within* each class. S_B measures the scatter caused by the two class means, which measures how scattered it is *between* different classes.

Now, we can formally define the FLD objective as

$$J = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}\,. \tag{21}$$

In other words, this optimization aims at finding a projection direction that makes the *between-class scatter much larger than the within-class scatter.*

A side note will be useful soon when we deal with multiclass problems. In Equation 19, one class is approximately represented by its mean (as $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, respectively). This is a reasonable simplification, especially when the scatter within each class is not large. However, if one mean vector acts as a surrogate for all examples in its respective class, we would expect that the *scatter matrix of the mean vectors* to be the between-class scatter, i.e., we should expect that $\sum_{i=1}^{2}(\boldsymbol{m}_i - \bar{\boldsymbol{m}})(\boldsymbol{m}_i - \bar{\boldsymbol{m}})^T$ is S_B, in which $\bar{\boldsymbol{m}} = \frac{\boldsymbol{m}_1 + \boldsymbol{m}_2}{2}$.

These two terms are not exactly the same, however, it is easy to verify that

$$\frac{1}{2}S_B = \sum_{i=1}^{2}(\boldsymbol{m}_i - \bar{\boldsymbol{m}})(\boldsymbol{m}_i - \bar{\boldsymbol{m}})^T\,. \tag{22}$$

Hence, it is reasonable to define S_B in the form of Equation 19, because a constant factor $\frac{1}{2}$ will not change the optimal parameter for Equation 21.

2.5 The optimization

The right hand side of Equation 21 is called the generalized Rayleigh quotient, or the Rayleigh-Ritz ratio, whose maximization is not only useful in computer science (e.g., in FLD), but also in other subjects (such as physics).

Finding the derivative of J with respect to $\boldsymbol{w}$ and set that to 0, we get

$$\frac{\partial J}{\partial \boldsymbol{w}} = \frac{2\left((\boldsymbol{w}^T S_W \boldsymbol{w}) S_B \boldsymbol{w} - (\boldsymbol{w}^T S_B \boldsymbol{w}) S_W \boldsymbol{w}\right)}{(\boldsymbol{w}^T S_W \boldsymbol{w})^2} = 0\,. \tag{23}$$

Hence, one necessary condition for optimality is

$$S_B \boldsymbol{w} = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} S_W \boldsymbol{w}\,. \tag{24}$$

Noticing that $\frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$ is a scalar value, this condition is stating that $\boldsymbol{w}$ should be a *generalized eigenvector of* S_B *and* S_W, with $\frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$ being its corresponding generalized eigenvalue!

Since $J = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$ is both the generalized eigenvalue and the optimization objective, we should use the generalized eigenvector of S_B and S_W whose corresponding generalized eigenvalue is the largest, which gives the optimal parameter $\boldsymbol{w}^\star$ that maximizes J.

2.6 Wait, we have a shortcut!

Although there is efficient routines to find the generalized eigenvalues and eigenvectors, to solve FLD for a binary problem we have an easier way—if we observe carefully we will find it.

Some small tweaking to Equation 24 tell us that

$$\begin{aligned} S_W \boldsymbol{w} &= \frac{\boldsymbol{w}^T S_W \boldsymbol{w}}{\boldsymbol{w}^T S_B \boldsymbol{w}} S_B \boldsymbol{w} && (25)\\ &= \frac{\boldsymbol{w}^T S_W \boldsymbol{w}}{\boldsymbol{w}^T S_B \boldsymbol{w}} (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} && (26)\\ &= \frac{\boldsymbol{w}^T S_W \boldsymbol{w}}{\boldsymbol{w}^T S_B \boldsymbol{w}} (\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} (\boldsymbol{m}_1 - \boldsymbol{m}_2) && (27)\\ &= c(\boldsymbol{m}_1 - \boldsymbol{m}_2)\,, && (28) \end{aligned}$$

in which we define $c = \frac{\boldsymbol{w}^T S_W \boldsymbol{w}}{\boldsymbol{w}^T S_B \boldsymbol{w}} (\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w}$ and c is a scalar value. Hence, this optimality condition directly leads to the optimal projection direction:

$$S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)\,. \tag{29}$$

Then, by normalizing this direction to make its ℓ_2 norm equal 1, we get the optimal $\boldsymbol{w}$. In the above equation, we have omitted the factor c, because it will disappear anyway in the normalization process.

2.7 The FLD method for binary problems

Algorithm 1 describes the steps to find the FLD projection direction for a binary classification problem.

Algorithm 1 The FLD algorithm for a binary problem

1: **Input**: a D-dimensional binary training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$
2: Compute $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and S_W following the equations in this note
3: Compute $\boldsymbol{w} \leftarrow S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$
4: Normalize: $\boldsymbol{w} \leftarrow \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$

The idea of FLD was originally described by Ronald Fisher, a famous statistician and biologist. This method is named after him, as *Fisher's Linear Discriminant*, abbreviated as FLD. This name is sometimes used interchangeably with LDA (which stands for Linear Discriminant Analysis).[3]

2.8 A caveat: what if S_W is not invertible?

There is an obvious shortcoming of Algorithm 1: it is not well-defined if S_W is not invertible.

If there are more positive (or negative) examples than the number of dimensions, i.e., if $N_+ > D$ or $N_- > D$, S_W will be invertible with high probability. However, if S_W^{-1} does not exist in some cases, we can use the Moore-Penrose (or MP for short) pseudoinverse to replace the inverse. A detailed coverage of the Moore-Penrose pseudoinverse is beyond the scope of this short note, we will only briefly explain how to compute it for S_W.

Instead of a $^{-1}$ superscript, the Moore-Penrose pseudoinverse is usually denoted by a $^+$ superscript. It is easy to compute the MP pseudoinverse for an 1×1 matrix, i.e., a scalar $x \in \mathbb{R}$, following the rule

$$x^+ = \begin{cases} 0 & \text{if } x = 0 \\ \dfrac{1}{x} & \text{otherwise} \end{cases} . \tag{30}$$

Then, a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$ has its MP pseudoinverse as

$$\Lambda^+ = \mathrm{diag}(\lambda_1^+, \lambda_2^+, \ldots, \lambda_D^+) . \tag{31}$$

Note that when Λ is invertible (i.e., when $\lambda_i \neq 0$ for $1 \leq i \leq D$), the MP pseudoinverse is the same as the usual matrix inverse.

Since S_W is the sum of two covariance matrices, it is positive semi-definite. Hence, we can find its spectral decomposition as $S_W = E\Lambda E^T$, where the diagonal matrix Λ contains the eigenvalues of S_W and the columns of the orthogonal

[3] LDA, however, is often used to refer to any linear discriminant function, including FLD but not limited to it.

matrix E contain the eigenvectors of S_W. The Moore-Penrose pseudoinverse of S_W is then

$$S_W^+ = E\Lambda^+ E^T \,. \tag{32}$$

Note that when S_W is invertible, $S_W^+ = S_W^{-1}$.

3 FLD for more classes

When our data is multiclass, i.e., when there are more than two possible labels, we need to extend Algorithm 1 to handle this situation.

We still have a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. However, y_i now has a different domain. Let $\mathcal{Y} = \{1, 2, \ldots, K\}$ denotes the K possible labels in a K-class problem, and $y_i \in \mathcal{Y}$ for all $1 \leq i \leq N$.

The idea to let the projection values of examples in these K classes to be far apart from each other still makes sense in this more complex setting. It is more complex to translate it to precise mathematics though.

In the binary case, we maximize $\frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$ (cf. Equation 21), whose meaning is to make the between-class variance much larger than the within-class variance. This intuition is still valid in multiclass problems. However, the key issue becomes: how to define these two variances (or scatter) when we have $K > 2$ classes (i.e., how do we define S_B and S_W?)

3.1 A slightly modified notation and S_W

The within-class scatter matrix S_W can be easily extended to multiclass, and keep its semantic meaning. We just need to modify the notation a bit.

Let X_k, N_k, $\boldsymbol{m}_k$, C_k and S_k be the subset of X that belongs to the k-th class, its size, mean, covariance matrix and scatter matrix, respectively, $1 \leq k \leq K$, i.e.,

$$X_k = \{\boldsymbol{x}_i | y_i = k\} \,, \tag{33}$$

$$N_k = |X_k| \,, \tag{34}$$

$$\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x} \in X_k} \boldsymbol{x} \,, \tag{35}$$

$$C_k = \frac{1}{N_k} \sum_{\boldsymbol{x} \in X_k} (\boldsymbol{x} - \boldsymbol{m}_k)(\boldsymbol{x} - \boldsymbol{m}_k)^T \,, \tag{36}$$

$$S_k = N_k C_k = \sum_{\boldsymbol{x} \in X_k} (\boldsymbol{x} - \boldsymbol{m}_k)(\boldsymbol{x} - \boldsymbol{m}_k)^T \,. \tag{37}$$

The within-class scatter is then simply the sum of scatter matrices of the K subsets, which measures the overall within-class scatter of the multiclass training set,

$$S_W = \sum_{k=1}^K S_k \,. \tag{38}$$

3.2 Candidates for S_B

It is, however, not trivial to extend the between-class scatter matrix S_B to multiclass in a natural way. A straightforward extension might be

$$\sum_{i=1}^{K}\sum_{j=1}^{K}(\boldsymbol{m}_i - \boldsymbol{m}_j)(\boldsymbol{m}_i - \boldsymbol{m}_j)^T\,, \tag{39}$$

which mimics Equation 19. But, this equation requires the creation of K^2 $D \times D$ matrices, and also need to sum them up. Hence, it is not very attractive in terms of computational efficiency.

Equation 22 may lead to another possible extension

$$\sum_{k=1}^{K}(\boldsymbol{m}_k - \bar{\boldsymbol{m}})(\boldsymbol{m}_k - \bar{\boldsymbol{m}})^T\,, \tag{40}$$

where

$$\bar{\boldsymbol{m}} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{m}_k\,. \tag{41}$$

Equation 40 is the scatter of the K mean vectors, which fits our intuition of the between-class scatter and is also fast to compute.

However, researchers have found an another definition, which makes even more sense.

3.3 A tale of three scatter matrices

Note we can compute the scatter matrix for the entire training set regardless of the class labels. We denote it as S_T,

$$S_T = \sum_{i=1}^{N}(\boldsymbol{x} - \boldsymbol{m})(\boldsymbol{x} - \boldsymbol{m})^T\,, \tag{42}$$

where

$$\boldsymbol{m} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \tag{43}$$

is the mean of all training points. Note that Equation 43 ($\boldsymbol{m}$) is different from Equation 41 ($\bar{\boldsymbol{m}}$).

S_T is called the *total scatter* matrix. It is curious to see what is the difference between the total and within-class scatter, and some algebraic manipulation gives

$$S_T - S_W = \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T\,. \tag{44}$$

The right hand side of Equation 44 closely resembles the scatter matrix, with two slight exceptions. First, $\boldsymbol{m}$ replaces $\bar{\boldsymbol{m}}$; second, N_k is multiplied before each term in the summation.

In FLD, the between-class scatter is defined as

$$S_B = \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T . \tag{45}$$

We want to point out that when the classes have equal sizes, i.e., when $N_i = N_j$ for any i and j, the above two deviations are trivial. In that case, $\boldsymbol{m} = \bar{\boldsymbol{m}}$ and Equation 45 differs from Equation 40 only by a constant multiplier $\frac{K}{N}$.

Furthermore, these changes are effective in imbalanced problems. Consider an imbalanced problem where $N_i \gg N_j$ for some i and j. We expect $\boldsymbol{m}_i$ to be more important than $\boldsymbol{m}_j$. Both Equations 43 and 45 put more weights on $\boldsymbol{m}_i$ than $\boldsymbol{m}_j$, while they are treated equally in Equations 41 and 40.

Finally, it makes sense if the sum of between- and within-class scatter matrices equals the total scatter matrix, since now we have $S_T = S_B + S_W$.

We also point out that: when $K = 2$, Equation 45 is different from Equation 19. When we are dealing with an imbalanced binary problem (e.g., when the negative class has much more examples than the positive class), it might help if we define S_B following Equation 45 even in the binary setup.

3.4 The solution

Now that S_B is not a rank 1 matrix in multiclass problems, Algorithm 1 is not applicable. However, we can still find the optimal projection direction by solving a generalized eigenvalue problem $S_B\boldsymbol{w} = \lambda S_W\boldsymbol{w}$.

When S_W is invertible, the generalized eigenvalue problem is equivalent to an eigenvalue problem $S_W^{-1}S_B\boldsymbol{w} = \lambda\boldsymbol{w}$. However, directly solving the generalized eigenvalue problem is more efficient. For example, in Matlab we can obtain the solution using a simple command `eig(A,B)`, where $A = S_B$ and $B = S_W$, and retrieve the generalized eigenvector associated with the largest generalized eigenvalue.

3.5 Finding more projection directions

As in PCA, we can find more projection direction for multiclass FLD: just use the generalized eigenvector associated with the top $K - 1$ largest generalized eigenvalues.

For a K class problem, we can extract at most $K - 1$ meaningful features (i.e., projected values). We will leave the proof of this statement to the readers of this note.

The generalized eigenvectors are not necessarily ℓ_2 normalized or perpendicular to each other.

Exercises

1. (Matrix rank) Let A be a real matrix of size $m \times n$. Its rank, denoted as $\text{rank}(A)$, is defined as the maximal number of linearly independent rows in A, which is also called the row rank. Similarly, the column rank is the number of linearly independent columns in A.

   (a) Prove that the row rank is equal to the column rank. Hence, $\text{rank}(X) = \text{rank}(X^T)$.

   (b) Prove that $\text{rank}(A) \leq \min(m, n)$.

   (c) Let X and Y be two matrices of the same size. Show that $\text{rank}(X + Y) \leq \text{rank}(X) + \text{rank}(Y)$.

   (d) Show that $\text{rank}(XY) \leq \min(\text{rank}(X), \text{rank}(Y))$ so long as the matrix multiplication is well-defined.

   (e) A matrix X of size $m \times n$ is called *full rank* if $\text{rank}(X) = \min(m, n)$. Prove that if X is full rank, then $\text{rank}(X) = \text{rank}(XX^T) = \text{rank}(X^TX)$.

   (f) Let $\boldsymbol{x}$ be a vector, what is the rank of $\boldsymbol{x}\boldsymbol{x}^T$? Show that the rank of a real symmetric matrix X equals its number of non-zero eigenvalues.

2. (Matrix norm) Similar to vectors norms, various matrix norms can be defined. $f : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$ is a matrix norm if it satisfies the following conditions:

   (1) $f(X) \geq 0$ for any $X \in \mathbb{R}^{m\times n}$;

   (2) $f(X) = 0$ if and only if X is a zero matrix;

   (3) $f(X + Y) \leq f(X) + f(Y)$ for $X, Y \in \mathbb{R}^{m\times n}$;

   (4) $f(cX) = |c|f(X)$ for $c \in \mathbb{R}$ and $X \in \mathbb{R}^{m\times n}$.

   (a) Prove that $\|X\|_2 = \max_{\|\boldsymbol{v}\|=1} \|X\boldsymbol{v}\|$ is a valid matrix norm (which is called the 2-norm). The vector norm considered here is the vector 2-norm $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$.

   (b) Let $\sigma_{\max}(X)$ denote the largest singular value of a matrix X. Prove that if X and Y are of the same size, then $\sigma_{\max}(X)+\sigma_{\max}(Y) \geq \sigma_{\max}(X+Y)$. (Hint: what is the relationship between $\sigma_{\max}(X)$ and $\|X\|_2$?)

3. (Condition number) Given any matrix norm $\|\cdot\|$, a corresponding *condition number* can be defined for a non-singular real square matrix. The condition number of a matrix X is defined as $\kappa(X) = \|X\|\|X^{-1}\|$. One frequently used condition number is the 2-norm condition, as $\kappa_2(X) = \|X\|_2\|X^{-1}\|_2$. A matrix is called *ill-conditioned* if its condition number is large.

   (a) If you already know the singular values of X as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$, what is the condition number $\kappa_2(X)$?

   (b) Let f be a function from a set $\mathbb{X}$ to $\mathbb{Y}$. Suppose $f(\boldsymbol{x}) = \boldsymbol{y}$ and $f(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{y} + \Delta\boldsymbol{y}$. We call f as *ill-conditioned* if a small $\Delta\boldsymbol{x}$ leads to

a large $\Delta \boldsymbol{y}$. Let A be a square full-rank matrix (i.e., invertible matrix) in $\mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, and we are interested in solving $A\boldsymbol{x} = \boldsymbol{b}$. Show that this linear system is ill-conditioned if $\kappa_2(A)$ is large. (You are not required to prove this conclusion. Just give some intuitions why an ill-conditioned matrix A is bad.)

(c) Show that an orthogonal matrix is well-conditioned (i.e., have small condition number).

4. (LU, LDL and Cholesky factorizations) In this problem, we study various factorizations that are useful in computing the generalized eigenvalue problem.

(a) Show that $\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \\ 0 & -2 & -3 \end{bmatrix}$.

(b) (Gauss transformation) In a more general form, let $\boldsymbol{v}$ be a vector in $\mathbb{R}^n$ and $v_k \neq 0$. Then, the *Gauss transformation* M_k is an $n \times n$ lower triangular matrix of the form:

$$M_k = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -\frac{v_{k+1}}{v_k} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -\frac{v_n}{v_k} & 0 & \cdots & 1 \end{bmatrix}. \tag{46}$$

That is, entries in M_k are all zeros except those in the diagonals and those in the k-th column. The diagonals are all ones. In the k-th column, the first $k-1$ entries are zeros, the k-th entry is 1, and the i-th entry ($i > k$) is $\frac{v_i}{v_k}$. What is the result of left-multiplying M_k to $\boldsymbol{v}$?

(c) For an $n \times n$ matrix A, what is the effect of running the following algorithm?

1: $A^{(1)} = A$
2: **for** $k = 1, 2, \ldots, n-1$ **do**
3: Let $\boldsymbol{v} = A^{(k)}_{:k}$ (i.e., the k-th column of $A^{(k)}$), and compute the Gauss transformation M_k based on $\boldsymbol{v}$
4: $A^{(k+1)} \leftarrow M_k A^{(k)}$
5: **end for**

In the above algorithm, we assume $a^{(k)}_{kk} \neq 0$ for $1 \leq k \leq n-1$.

(d) Show that $\det(M_k) = 1$ for all $1 \leq k < n$. If $L = M_{n-1}M_{n-2}\ldots M_1$, prove that L is lower triangular and its determinant is 1.

(e) (LU Factorization) Let $A \in \mathbb{R}^{n\times n}$. And, for $1 \leq k \leq n-1$, the leading principal submatrix $A_{1:k,1:k}$ is non-singular. Prove that there exists a decomposition $A = LR$, where $L \in \mathbb{R}^{n\times n}$ is lower triangular with its

diagonal entries all being 1, and $U \in \mathbb{R}^{n \times n}$ is upper triangular. If A is furthermore non-singular, prove that the LU decomposition is *unique*.

(f) (LDL Factorization) We pay our attention to the LDL factorization of a real symmetric and positive definite matrix. If A is such a matrix, prove that there exists a *unique* lower triangular real matrix L (with its diagonal entries all being 1) and a diagonal matrix D such that $A = LDL^T$.

(g) (Cholesky Factorization) Let A be a real symmetric positive definite matrix. Then, prove that there exists a *unique* real lower triangular matrix G such that $A = GG^T$, and the diagonal entries in G are all positive. This matrix decomposition is called the *Cholesky* factorization or Cholesky decomposition, named after the French mathematician Andre-Louis Cholesky.

5. (Solving FLD) Consider a C ($C > 2$) class FLD problem, in which there are N examples of D dimensions.

   (a) If $N < D$, will S_W be invertible?

   (b) Prove that at most $C - 1$ generalized eigenvectors can be obtained. (Hint: What is the rank of S_B?)

   (c) Explain why S_W is very likely invertible if $N > D$, especially if $N \gg D$?

   (d) The generalized eigenvalue problem $S_B\boldsymbol{w} = \lambda S_W\boldsymbol{w}$ is equivalent to $S_W^{-1}S_B\boldsymbol{w} = \lambda\boldsymbol{w}$ when S_W is invertible. However, in practice we do not solve $S_W^{-1}S_B\boldsymbol{w} = \lambda\boldsymbol{w}$. One algorithm for computing the FLD solution is as follows. First, assuming S_W is positive definite, the Cholesky factorization finds a G such that $S_W = GG^T$. Second, compute $C = G^{-1}S_BG^{-T}$. Third, diagonalize C such that there is an orthogonal matrix Q and Q^TCQ is a diagonal matrix. Finally, let $X = G^{-T}Q$. Prove that X^TS_BX is diagonal and X^TS_WX is the identity matrix. Show that the generalized eigenvectors are in the columns of X and the generalized eigenvalues are in the diagonal entries of X^TS_BX. Are the generalized eigenvectors computed this way unit norm or orthogonal?

6. (PCA+FLD face recognition) Both PCA and FLD are useful in the face recognition application.

   (a) The ORL dataset is an early dataset for face recognition. Download the ORL dataset from http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. Read the instructions in that page to understand the format of this dataset.

   (b) OpenCV is an open source computer vision library, which provides many functions useful for various computer vision applications. Download the library from http://opencv.org/. Learn OpenCV basics from http://docs.opencv.org/.

   (c) There is an OpenCV tutorial on face recognition, which is available at http://docs.opencv.org/2.4/modules/contrib/doc/facerec/

facerec_tutorial.html. Try to understand every line of code in the tutorial, especially those on Eigenfaces (PCA) and Fisherfaces (FLD). Run experiments on the ORL dataset, analyze the differences between recognition results of these methods.

(d) In the Eigenface experiment, you can reconstruct an approximate face image using the eigenfaces (i.e., eigenvectors). Modify the source code in the OpenCV tutorial, and observe the visualization using different amount of eigenfaces. How many eigenfaces are required if you want the face reconstructed from eigenfaces are visually indistinguishable from the original input face image?