Kernel Fisher Discriminant Analysis (KFDA)

Soham Chhapre, (0801CS171081)
1899sohamchhapre@gmail.com

Ashutosh Mehra, (0801CE171023)
mehraashu1017@gmail.com

December 13, 2020

Contents

Chapter 1

Introduction

1.1 Linear discriminant Analysis

Linear Discriminant Analysis (LDA) was proposed by R. Fischer in 1936. It consists in finding the projection hyperplane that minimizes the inter-class variance and maximizes the distance between the projected means of the classes. Similarly to PCA, these two objectives can be solved by solving an eigenvalue problem with the corresponding eigenvector defining the hyperplane of interest. This hyperplane can be used for classification, dimensionality reduction and for interpretation of the importance of the given features.

1.2 Kernel Fisher discriminant Analysis

kernel Fisher discriminant analysis (KFDA)[2] also known as kernel discriminant analysis, is a kernelized version of linear discriminant analysis (LDA). It is named after Ronald Fisher. Using the kernel trick, LDA is implicitly performed in a new feature space, which allows non-linear mappings to be learned.
The principle that underlies KFDA is that input data are mapped into a high-dimensional feature space by using a nonlinear function, after which FDA is used for recognition or classification in feature space.

Chapter 2

Mathematical Formulation

2.1 Kernel trick with LDA

To extend LDA to non-linear mappings, the data, given as the ℓ points x_i, can be mapped to a new feature space, $\mathbf{F}$, via some function $\boldsymbol{\phi}$. In this new feature space, the function that needs to be maximized is [2]

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_B^{\phi}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_W^{\phi}\mathbf{w}} \tag{2.1}$$

where,

$$\begin{aligned} \mathbf{S}_B^{\phi} &= \left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right)\left(\mathbf{m}_2^{\phi} - \mathbf{m}_1^{\phi}\right)^{\mathrm{T}} \\ \mathbf{S}_W^{\phi} &= \sum_{i=1,2}^{4}\sum_{n=1}\left(\phi\left(\mathbf{x}_n^i\right) - \mathbf{m}_i^{\phi}\right)\left(\phi\left(\mathbf{x}_n^i\right) - \mathbf{m}_i^{\phi}\right)^{\mathrm{T}} \end{aligned} \tag{2.2}$$

and

$$\mathbf{m}_i^{\phi} = \frac{1}{l_i}\sum_{j=1}^{l_i}\phi\left(\mathbf{x}_j^i\right) \tag{2.3}$$

Further, note that $\mathbf{w} \in F$.
LDA can be reformulated in terms of dot products by first noting that $\mathbf{w}$ will have an expansion of the form

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \phi\left(\mathbf{x}_i\right) \tag{2.4}$$

Then note that

$$\mathbf{w}^{\mathrm{T}} \mathbf{m}_i^{\phi} = \alpha^{\mathrm{T}} \mathbf{M}_i \tag{2.5}$$

where

$$\left(\mathbf{M}_i\right)_j = \frac{1}{l_i} \sum_{k=1}^{l_i} k\left(\mathbf{x}_j, \mathbf{x}_k^i\right) \tag{2.6}$$

The numerator of $J(\mathbf{w})$ can then be written as:

$$\mathbf{w}^{\mathrm{T}} \mathbf{S}_B^{\phi} \mathbf{w} = \alpha^{\mathrm{T}} \mathbf{M} \alpha, \quad \text{where} \quad \mathbf{M} = \left(\mathbf{M}_2 - \mathbf{M}_1\right)\left(\mathbf{M}_2 - \mathbf{M}_1\right)^{\mathrm{T}} \tag{2.7}$$

Similarly, the denominator can be written as

$$\mathbf{w}^{\mathrm{T}} \mathbf{S}_W^{\phi} \mathbf{w} = \alpha^{\mathrm{T}} \mathbf{N} \alpha, \quad \text{where} \quad \mathbf{N} = \sum_{j=1,2} \mathbf{K}_j \left(\mathbf{I} - \mathbf{1}_{l_j}\right) \mathbf{K}_j^{\mathrm{T}} \tag{2.8}$$

The equation for J can be rewritten as

$$J(\alpha) = \frac{\alpha^{\mathrm{T}} \mathbf{M} \alpha}{\alpha^{\mathrm{T}} \mathbf{N} \alpha} \tag{2.9}$$

2.2 Multi-class KFDA

Let c be the number of classes. Then multi-class KFD[1] involves projecting the data into a $(c-1)$ -dimensional space using $(c-1)$ discriminant functions

$$y_i = \mathbf{w}_i^{\mathrm{T}} \phi(\mathbf{x}) \quad i = 1, \ldots, c-1 \tag{2.10}$$

In matrix notation

$$y = \mathbf{W}^{\mathrm{T}} \phi(\mathbf{x}) \tag{2.11}$$

The multi-class KFD written as:

$$\mathbf{A}^* = \underset{\mathbf{A}}{\operatorname{argmax}} = \frac{\left|\mathbf{A}^{\mathrm{T}} \mathbf{M} \mathbf{A}\right|}{\left|\mathbf{A}^{\mathrm{T}} \mathbf{N} \mathbf{A}\right|} \tag{2.12}$$

where $A = [\alpha_1, \ldots, \alpha_{c-1}]$ and

$$\begin{aligned} M &= \sum_{j=1} l_j \left(\mathbf{M}_j - \mathbf{M}_*\right) \left(\mathbf{M}_j - \mathbf{M}_*\right)^{\mathrm{T}} \\ N &= \sum_{j=1}^{c} \mathbf{K}_j \left(\mathbf{I} - \mathbf{1}_{l_j}\right) \mathbf{K}_j^{\mathrm{T}} \end{aligned} \tag{2.13}$$

$\mathbf{A}^*$ can then be computed by finding the $(c-1)$ leading eigen vectors of $N^{-1}M$ [3]

Chapter 3

Algorithm

In this section, we give an overview of the KFDA algorithm where we formulate the optimization problem as a standard eigen problem.

3.1 KFDA algorithm

Algorithm 1: KFDA algorithm

input : $X \epsilon R^{nxd}$, $Y \epsilon R^{n}$
where d is dimensionality of X

output: w

1. Calculate kernel matrix K
2. Calculate M and N matrix
   - 2.1 $M = \sum_{j=1}^{c} l_j \left(\mathbf{M}_j - \mathbf{M}_*\right) \left(\mathbf{M}_j - \mathbf{M}_*\right)^{\mathrm{T}}$
   - 2.2 $N = \sum_{j=1}^{c} \mathbf{K}_j \left(\mathbf{I} - \mathbf{1}_{l_j}\right) \mathbf{K}_j^{\mathrm{T}}$
3. Calculate $N^{-1}M$
4. Find eigen values of $(N^{-1}M)$
5. Return top $(c-1)$ eigen values

Chapter 4

Documentation of API

4.1 Package organization

Kernel Fisher Discriminant Analysis (KFDA) Discriminant Analysis in high dimension using the kernel trick.

Parameters:

- n_components : int, the amount of Fisher directions to use.
  default=2
  This is limited by the amount of classes minus one.
  Number of components (lower than number of classes -1) for dimension reduction.
- kernel : str,["linear" |"poly" |"rbf" |"sigmoid" |"cosine" |"precomputed"] default="linear".
  The kernel to use. Use **kwds to pass arguments to these functions. See https://scikit-learn.org/stable/modules/metrics.html#polynomial-kernel for more details.
- alpha : float, default=1e-3
  Regularization term for singular within-class matrix.
- tol : float, default=1e-4
  Singularity toleration level.
- kprms : mapping of string to any, default=None
  parameters to pass to the kernel function.

Attributes:

- X_: Training vector after applying input validation
- y_: label vector after applying input validation
- W_: array of shape (n_components)
  contains weights of eigen vectors
- unique_classes : array of shape (n_classes,)
  The unique class labels

4.2 Methods

- **fit**
  Fit the model from the data in X and the labels in y.
  Parameters
  - X : array-like, shape (N x d)
    Training vector, where N is the number of samples, and d is the number of features.
  - y : array-like, shape (N)
    Labels vector, where N is the number of samples.

  Returns
  - self : object
    Returns the instance itself.
- **transform**
  Applies the kernel transformation.
  Parameters
  - X : (N x d) matrix, optional
    Data to transform. If not supplied, the training data will be used.

  Returns
  - transformed: (N x d') matrix.
    Input data transformed by the learned mapping.

- **get_kernel_**
  Get the kernel matrix.
  Parameters
  - X : (N x d) matrix,
    Data to get kernel.
  - y : (N x d) matrix, optional

  Returns
  - Kernel Matrix: (N x d') matrix.
    returns A kernel matrix K such that $K_{i,j}$ is the kernel between the *ith* and jth vectors of the given matrix X, if Y is None. If Y is not None, then $K_{i,j}$ is the kernel between the ith array from X and the jth array from Y

Chapter 5

Example

5.1 Example 1

```
from KFDA import KFDA
y=np.array([1,1,1,1,2,2,2])
X=np.array([[2,3],[3,3],[4,5],[5,5],[1,0],[2,1],[3,1]])
kfda = KFDA(n_components=1,kernel='linear')
kfda.fit(X,y)
```

5.2 Example 2

```
from KFDA import KFDA
y=np.array([1,1,1,1,2,2,2])
X=np.array([[2,3],[3,3],[4,5],[5,5],[1,0],[2,1],[3,1]])
kfda = KFDA(n_components=1,kernel='poly')
kfda.fit(X,y)
kfda.tranform()
```

Chapter 6

Learning Outcome

- We came to know about LDA which is a dimensionality reduction technique which is commonly used for the supervised classification problems.
- We got to learn how the LDA works
- We learned the kernel trick and uses to map the date into higher dimensions
- We learned how to implement LDA on non seperalable data using the kernel trick i.e Kernel LDA

Bibliography

[1] G. Baudat and F. Anouar. "Generalized Discriminant Analysis Using a Kernel Approach". In: *Neural Computation* 12.10 (2000), pp. 2385–2404. DOI: 10.1162/089976600300014980. eprint: https://doi.org/10.1162/089976600300014980. URL: https://doi.org/10.1162/089976600300014980.

[2] S. Mika et al. "Fisher discriminant analysis with kernels". In: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*. 1999, pp. 41–48. DOI: 10.1109/NNSP.1999.788121.

[3] Jianguo Zhang, Kai-kuang Ma, et al. "Kernel fisher discriminant for texture classification". In: (2004).