Shri G.S. Institute of Technology and Science, Indore

BTech-CSE-IV Year

Data Science - Project 2

Multiview Uncorrelated Linear Discriminant Analysis (MULDA)

Guided By:

Mr. Surendra Gupta

Submitted By:

Aadeesh Jain (0801CS171001)
Harsh Pastaria (0801CS171027)
Kanishk Gupta (0801CS171031)

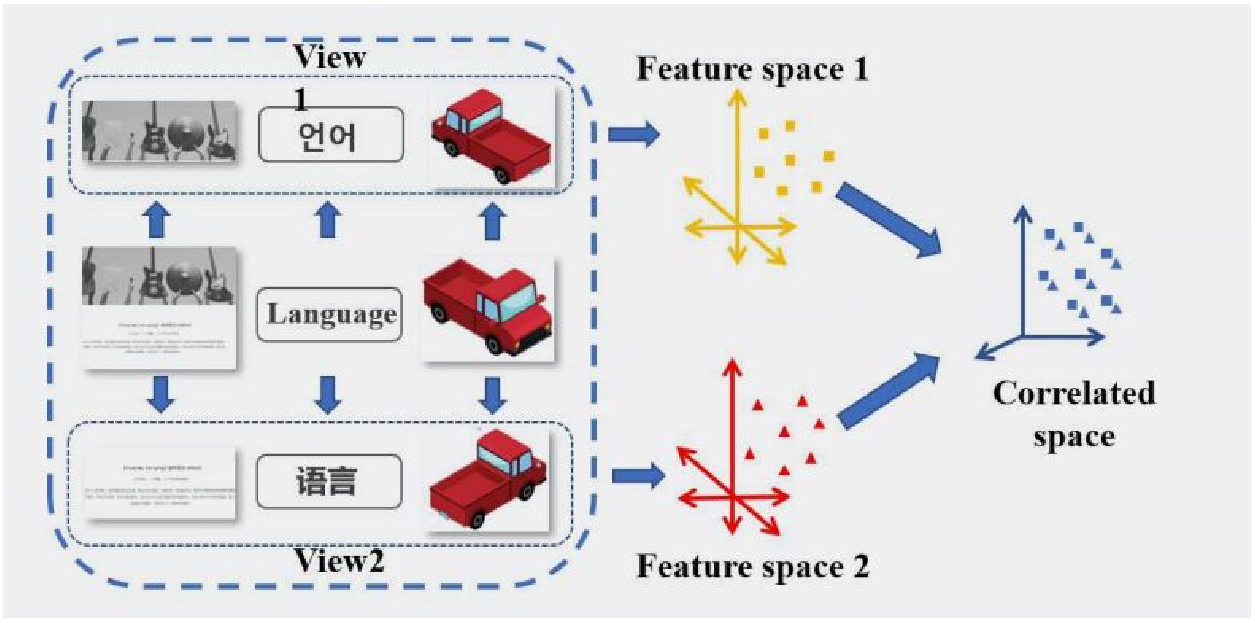
MultiView Uncorrelated Linear Discriminant Analysis(MULDA)

Introduction

Before understanding MLDA lets understand first meaning of multiview learning:

Multiview Learning

Multi-view learning is also known as data fusion or data integration from multiple feature sets.Multi-view learning is an emerging direction in machine learning which considers learning with multiple views to improve the generalization performance.Many real-world datasets can be described from multiple "viewpoints" such as pictures taken from different angles of the same object, different language expressions of the same semantic, texts and images on the same web page, etc. The representations from different perspectives can be treated as different views.



The above image depicts a perfect example of multiview learning with multiple views of the car from different angles and the final goal is to project feature space on a smaller size subspace.

LDA:

LDA stands for linear discriminant analysis. It is the most used **Dimensionality Reduction Technique.** It basically finds the axes that maximises the separation between multiple classes.

Uncorrelated LDA (ULDA) is an extension of LDA by adding some constraints into the optimization objective of LDA, so that the feature vectors extracted by ULDA could contain minimum redundancy.

MULDA Uncorrelated LDA (ULDA) is an extension of LDA by adding some constraints into the optimization objective of LDA, so that the feature vectors extracted by ULDA could contain minimum redundancy.
It extracts uncorrelated features in each view and computes transformations of each view to project data into a common subspace.

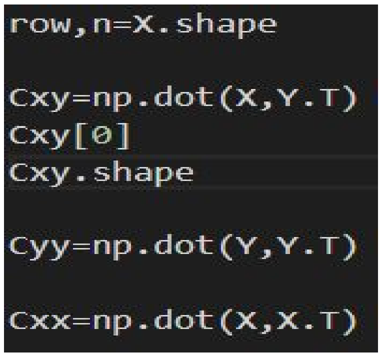
Mathematical Formulation

Let X and Y be two normalized feature matrices whose mean values are 0, respectively. X = [x1, x2, . . . , xn] = [X1, X2, . . . , Xk], X ∈ Rp×n, where xj ∈ Rp(1 ≤ j ≤ n) represents an example, n is the number of examples, m is the number of classes, and Xi ∈ Rp×ni denotes the subset of all the examples in class i with ni being the number of examples in this subset. Similarly,
Y = [y1, y2, . . . , yn] = [Y1, Y2, . . . , Yk], Y ∈ Rq×n. Then we have a two-view dataset {(x1, y1), . . . , (xn, yn)}

1. Calculation of covariance matrix

$$C_{xy} = \frac{1}{n}XY^T, \; C_{xx} = \frac{1}{n}XX^T, \; C_{yy} = \frac{1}{n}YY^T.$$

   Cxy is the covariance matrix between two dataset X and Y, whereas Cxx and Cyy are the covariance matrix between same dataset i.e., X and Y.



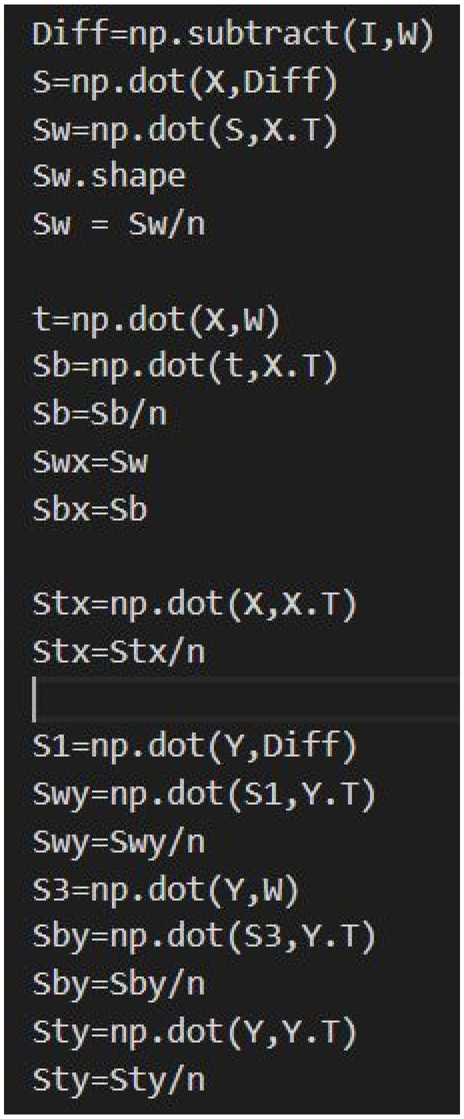
```
row,n=X.shape

Cxy=np.dot(X,Y.T)
Cxy[0]
Cxy.shape

Cyy=np.dot(Y,Y.T)

Cxx=np.dot(X,X.T)
```

2. Calculation of Scatter Matrices Sb, Sw, St where Sb, Sw, and St denote the between-class, within-class, and total scatter matrix, respectively. These scatter matrices are calculated as

$$S_w = \frac{1}{n}X(I - W)X^T, \; S_b = \frac{1}{n}XWX^T, \; S_t = \frac{1}{n}XX^T$$

   where $W = \mathrm{diag}(W_1, W_2, \ldots, W_k)$, and W_i is an $(n_i \times n_i)$ matrix with all elements equal to $(1/n_i)$.

   here n represents number of features in dataset.



```
Diff=np.subtract(I,W)
S=np.dot(X,Diff)
Sw=np.dot(S,X.T)
Sw.shape
Sw = Sw/n

t=np.dot(X,W)
Sb=np.dot(t,X.T)
Sb=Sb/n
Swx=Sw
Sbx=Sb

Stx=np.dot(X,X.T)
Stx=Stx/n

S1=np.dot(Y,Diff)
Swy=np.dot(S1,Y.T)
Swy=Swy/n
S3=np.dot(Y,W)
Sby=np.dot(S3,Y.T)
Sby=Sby/n
Sty=np.dot(Y,Y.T)
Sty=Sty/n
```

3. The optimization problem of MULDA can be formulated as

$$\begin{aligned}\max_{w_{xr},w_{yr}} \quad & w_{xr}^T S_{b_x} w_{xr} + w_{yr}^T S_{b_y} w_{yr} + 2\gamma w_{xr}^T C_{xy} w_{yr} \\ \text{s.t.} \quad & w_{xr}^T S_{t_x} w_{xr} + \sigma w_{yr}^T S_{t_y} w_{yr} = 1 \\ & w_{xr}^T S_{t_x} w_{xj} = w_{yr}^T S_{t_y} w_{yj} = 0 \\ & (j = 1, 2, \ldots, r-1)\end{aligned}$$

   where wxr and wyr represent the rth discriminant vectors of matrices X and Y, respectively

4. The rth discriminant vector pair (wxr,wyr) of matrices X and Y is the eigenvector corresponding to the maximum eigenvalue of the following generalized eigenequation:

$$\begin{bmatrix} P_x & \mathbf{0} \\ \mathbf{0} & P_y \end{bmatrix} \begin{bmatrix} S_{b_x} & \gamma C_{xy} \\ \gamma C_{yx} & S_{b_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix} = \lambda \begin{bmatrix} S_{t_x} & \mathbf{0} \\ \mathbf{0} & \sigma S_{t_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix}$$

where

$$\begin{aligned} P_x &= I - S_{t_x} D_x^T \left(D_x S_{t_x} D_x^T\right)^{-1} D_x \\ P_y &= I - S_{t_y} D_y^T \left(D_y S_{t_y} D_y^T\right)^{-1} D_y \\ D_x &= \left[w_{x1}, w_{x2}, \ldots, w_{x(r-1)}\right]^T \\ D_y &= \left[w_{y1}, w_{y2}, \ldots, w_{y(r-1)}\right]^T \\ I &= \mathrm{diag}(1, 1, \ldots, 1). \end{aligned}$$

5.

$$\begin{aligned} \alpha &= \left[\alpha_1, \alpha_2, \ldots, \alpha_{r-1}\right]^T \\ D_x &= \left[w_{x1}, w_{x2}, \ldots, w_{x(r-1)}\right]^T \end{aligned}$$

$$\begin{aligned} \beta &= \left[\beta_1, \beta_2, \ldots, \beta_{r-1}\right]^T \\ D_y &= \left[w_{y1}, w_{y2}, \ldots, w_{y(r-1)}\right]^T \end{aligned}$$

$$\begin{aligned} P_x &= I - S_{t_x} D_x^T \left(D_x S_{t_x} D_x^T\right)^{-1} D_x \\ P_y &= I - S_{t_y} D_y^T \left(D_y S_{t_y} D_y^T\right)^{-1} D_y. \end{aligned}$$

6. Final generated Eigenvalue solution

$$\begin{bmatrix} P_x & \mathbf{0} \\ \mathbf{0} & P_y \end{bmatrix} \begin{bmatrix} S_{b_x} & \gamma C_{xy} \\ \gamma C_{yx} & S_{b_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix} = \lambda \begin{bmatrix} S_{t_x} & \mathbf{0} \\ \mathbf{0} & \sigma S_{t_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix}.$$

7.

$$\text{I) } Z = \begin{bmatrix} W_x & \mathbf{0} \\ \mathbf{0} & W_y \end{bmatrix}^T \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$\text{II) } Z = \begin{bmatrix} W_x \\ W_y \end{bmatrix}^T \begin{bmatrix} X \\ Y \end{bmatrix}$$

Algorithm

```
Require :
        Training data X,Y;
        Dimension of the transformed feature space d;
        Parameter $\lambda$.
Ensure:
    Transformed data Z.
1:  Construct matrices $C_{xy}$, $S_{bx}$ , $S_{by}$ , $S_{tx}$ , $S_{ty}$ as in eqn(1),(2).
2:  $\sigma \leftarrow \frac{tr(S_{tx})}{tr(S_{ty})}$.
3:  Initialize $D_x$ and $D_y$ to be empty matrices.
4:  for r = 1 to d do
5:      Construct matrices $P_x$, $P_y$ as in (5);
6:      Obtain the $r^{th}$ vector pair ($w_{xr}$,$w_{yr}$) by solving (6);
7:      Set $D_x = [D_x,w_{xr}]$ (append $w_{xr}$ to $D_x$ as the last column),
        $D_y= [D_y,w_{yr}]$ (append $w_{yr}$ to $D_y$ as the last column).
8:  end for
9:  $W_x \leftarrow D_x$, $W_y \leftarrow D_y$.
10: Extract features according to (7).
11: return Z.
```

Documentation of API

- **Package Organization**

  mulda.py : This python file contain different method implementations like calculation of covariance matrix and scatter matrix and feature extraction by solving vector pair using eigenvalue decomposition

  DataScience_MULDA.ipynb : This is the jupyter notebook which contains the calculation of various parameters on sample dataset

  test.py : This python file contains the main method and code testing using sample data and it outputs the extracted features

  Dataset: mfeat-mor , mfeat-pix
  These are sample dataset for 2 views

- **Parameters in mulda.py**
  1. **X,Y :** 2 views of dataset
  2. **Cxx,Cyy,Cxy:** Covariance matrix between x-x,y-y and x-y
  3. **Swx,Sbx,Stx,Swy,Sby,Sty :** Within class, Between class and Total Scatter matrix along X and Y
  4. **wx,wy :** Final projections along X and Y

5. **Dx, Dy :** Dx =Transpose of [wx1,wx2,...,wx(r−1)]
   Dy = Transpose of[wy1,wy2,...,wy(r−1)]
6. **Px, Py :** Matrices used for calculation of Wx,Wy

$$P_x = I - S_{t_x} D_x^T \left(D_x S_{t_x} D_x^T \right)^{-1} D_x$$
$$P_y = I - S_{t_y} D_y^T \left(D_y S_{t_y} D_y^T \right)^{-1} D_y.$$

7. **d :** Dimension of transformed feature space
8. **wxr, wyr :** rth discriminant vector pair of matrices X and Y
9. **Wx, Wy :** Wx = [wx1,wx2,...,wxd] and Wy = [wy1,wy2,...,wyd]
10. **Z :** Final feature extraction vector

- **Methods in mulda.py**
  1. **selfCovariance :** returns the self covariance matrix as explained in eqn.(1)
  2. **diagMatrixW :** returns W=diag(W1, W2,..., Wk) ,which is an (ni * ni) matrix with all elements equal to (1/ni)
  3. **withinClassScatterMatrix :** returns the within class scatter matrix as explained in eqn.(2)
  4. **betweenClassScatterMatrix :** returns the between class scatter matrix as explained in eqn.(2)
  5. **totalScatterMatrix :** returns the total scatter matrix as explained in eqn.(2)
  6. **calculatingSigma :** return sigma value which is

$$\sigma \leftarrow \frac{tr(S_{tx})}{tr(S_{ty})}.$$

7. **fit :** It takes arguments as the first view,second view and dataset parameters and returns array of projections(Dx, Dy) calculated by find the values of rth vector pair using Lagrange duality equation :

$$\begin{bmatrix} P_x & \mathbf{0} \\ \mathbf{0} & P_y \end{bmatrix} \begin{bmatrix} S_{b_x} & \gamma C_{xy} \\ \gamma C_{yx} & S_{b_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix} = \lambda \begin{bmatrix} S_{t_x} & \mathbf{0} \\ \mathbf{0} & \sigma S_{t_y} \end{bmatrix} \begin{bmatrix} w_{xr} \\ w_{yr} \end{bmatrix}.$$

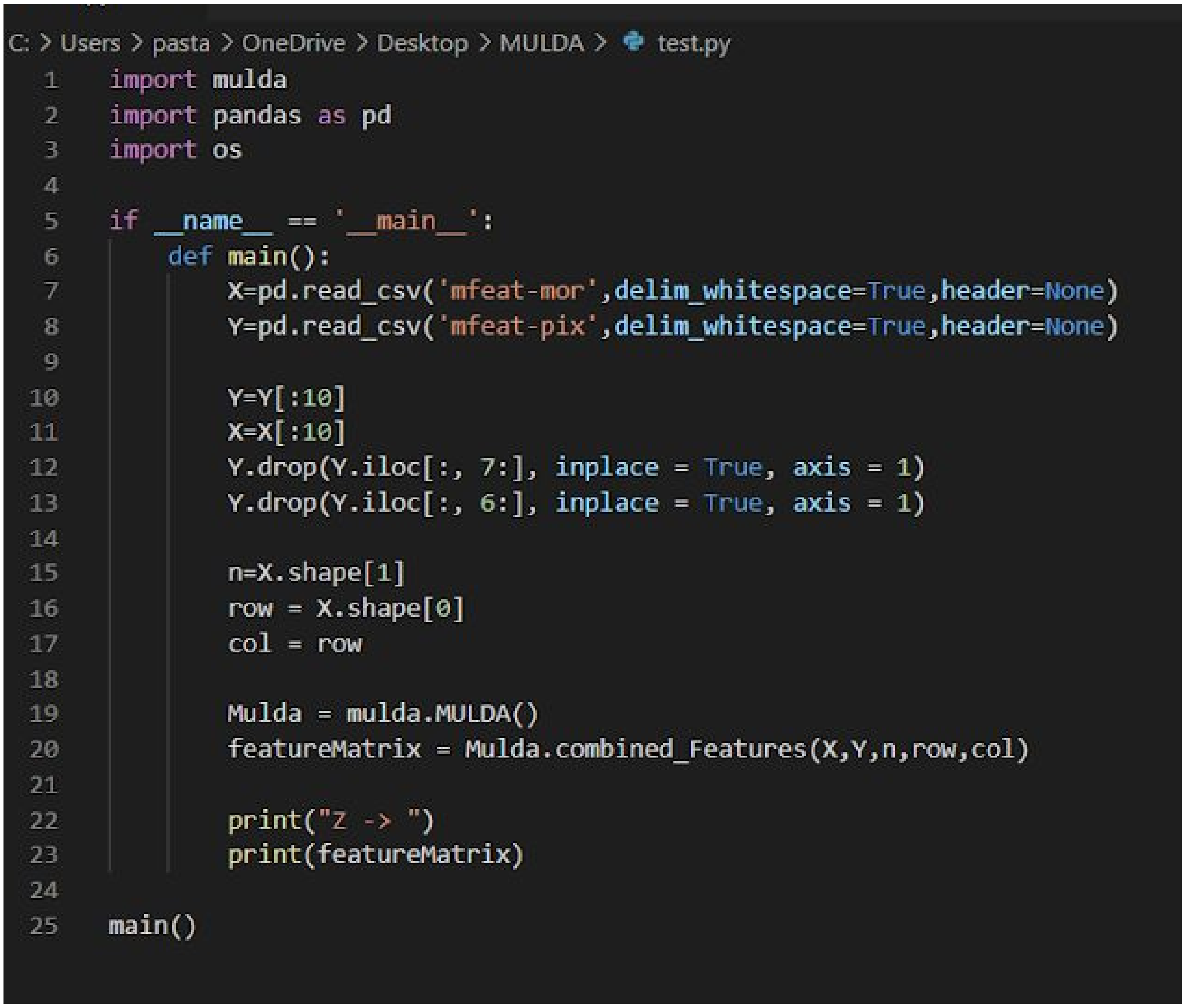
$$\frac{1}{\mu} \Sigma_x^{-1} \Sigma_{xy} \Sigma_y^{-1} \Sigma_{yx} a - \mu a = 0$$

8. **combined_features :** It takes the output of fit method and returns the extracted feature vector Z :-

$$\text{I) } Z = \begin{bmatrix} W_x & \mathbf{0} \\ \mathbf{0} & W_y \end{bmatrix}^T \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$\text{II) } Z = \begin{bmatrix} W_x \\ W_y \end{bmatrix}^T \begin{bmatrix} X \\ Y \end{bmatrix}$$

Example



C: > Users > pasta > OneDrive > Desktop > MULDA > test.py

```python
import mulda
import pandas as pd
import os

if __name__ == '__main__':
    def main():
        X=pd.read_csv('mfeat-mor',delim_whitespace=True,header=None)
        Y=pd.read_csv('mfeat-pix',delim_whitespace=True,header=None)

        Y=Y[:10]
        X=X[:10]
        Y.drop(Y.iloc[:, 7:], inplace = True, axis = 1)
        Y.drop(Y.iloc[:, 6:], inplace = True, axis = 1)

        n=X.shape[1]
        row = X.shape[0]
        col = row

        Mulda = mulda.MULDA()
        featureMatrix = Mulda.combined_Features(X,Y,n,row,col)

        print("Z -> ")
        print(featureMatrix)

main()
```

Learning Outcomes

1.Many views of data are needed as single-view data cannot comprehensively describe the information.

2.Understanding and Implementing multi-view learning.

3.MULDA utilizes the principle of CCA and Uncorrelated LDA.

4.. A well designed multi-view learning strategy may bring performance improvements.

5.We understand the use of many mathematical algorithms and equations in MULDA like **Lagrange Multiplier Technique, EigenValue Decomposition** etc..

References

1.Sun: Multiview uncorrelated discriminant analysis - Google Scholar (elsevier.com)

2.Canonical Correlation Analysis(CCA) Based Multi-View Learning: An Overview https://arxiv.org/pdf/1907.01693.pdf