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

BTech-CSE-IV Year

Data Science - Project 2

Multiview Linear Discriminant Analysis (MLDA)

Guided By:

Mr. Surendra Gupta

Submitted By:

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

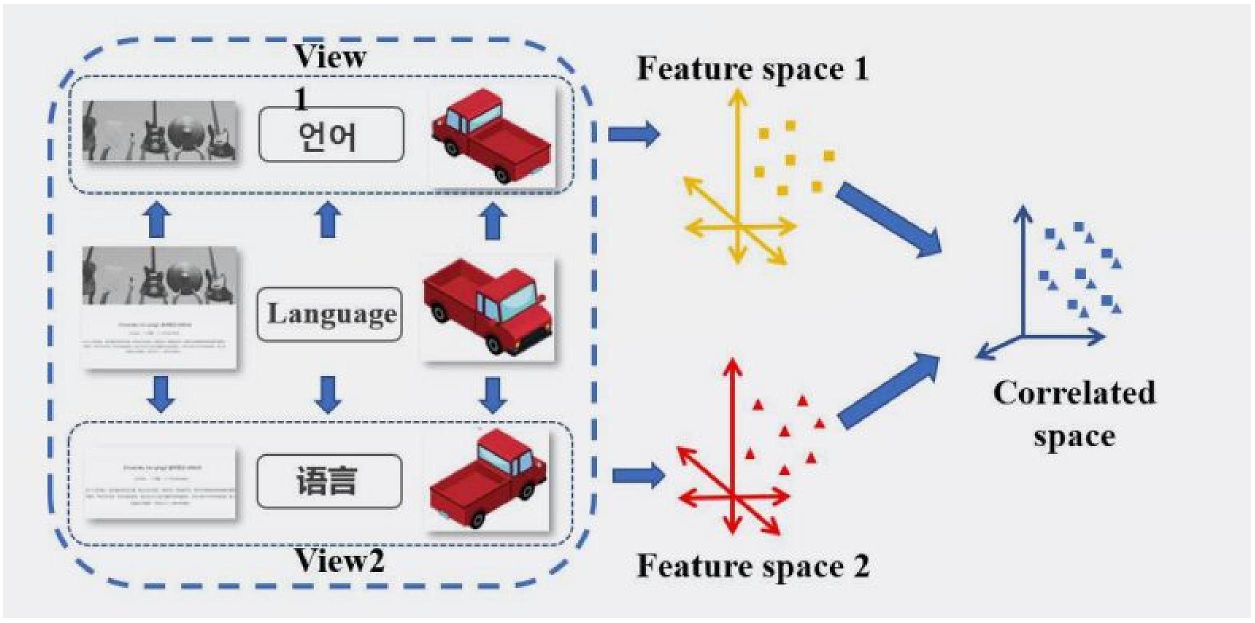
Multi View Linear Discriminant Analysis(MLDA)

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.

MLDA is described as combination of CCA and LDA

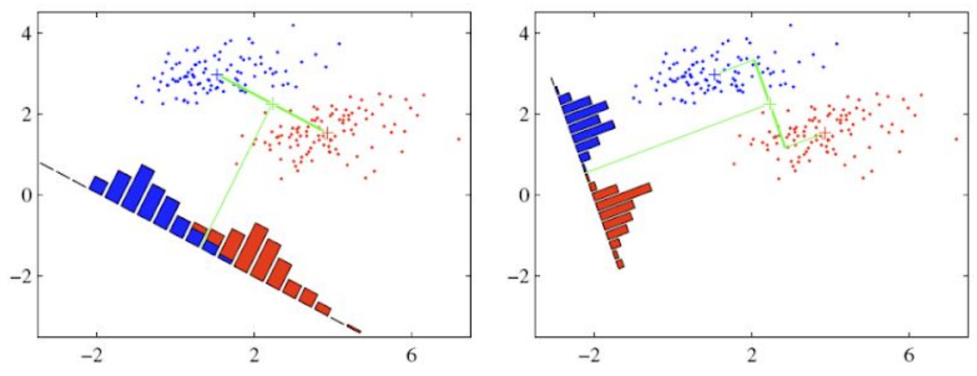
CCA

Canonical correlation analysis (CCA) is a popular technique to utilize information stemming from multiple feature sets. However, it does not exploit **label information** effectively. Later multiview linear discriminant analysis (MLDA) was proposed through combining CCA and linear discriminant analysis (LDA).
The traditional CCA has the following limitations: 1) It cannot handle more than two views. 2) It can only calculate the linear correlation between two views, whereas in many real-world applications the true relationship between the views may be nonlinear. 3) In supervised classification, labels are available; however, CCA, as an unsupervised algorithm, completely ignores the labels, and hence wastes information.

So here comes the LDA which will overcome the above limitations.

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. LDA is an effective supervised feature extraction method for single-view learning. It seeks an optimal linear transformation to map the data into a subspace so that the ratio between between-class distance and within-class distance is maximized.
The basic goal of Lda is to project the feature space(of N-dimension) on smaller subspace of size k(k<=N-1)



Through optimizing the corresponding objective the discrimination and correlation between two views can be maximized simultaneously.

MLDA:

LDA is a supervised algorithm for a single view that minimizes the within-class variance and maximizes the between-class variance. Multi-view linear discriminant analysis (MLDA) combines LDA and CCA, which not only ensures the discriminative ability within a single view, but also maximizes the correlation between different views. Through optimizing the corresponding objective, discrimination in each view and correlation between two views can be maximized simultaneously.

Mathematical Formulation

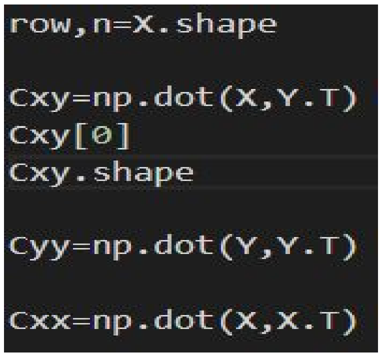
- **Formulation**

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 the 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)
```

The aim of CCA is to find two projection directions wx and wy, one for each View. Since wx and wy are scaler independent therefore,

$$\max_{w_x, w_y} \quad w_x^T C_{xy} w_y$$
$$\text{s.t.} \quad w_x^T C_{xx} w_x = 1,\ w_y^T C_{yy} w_y = 1.$$

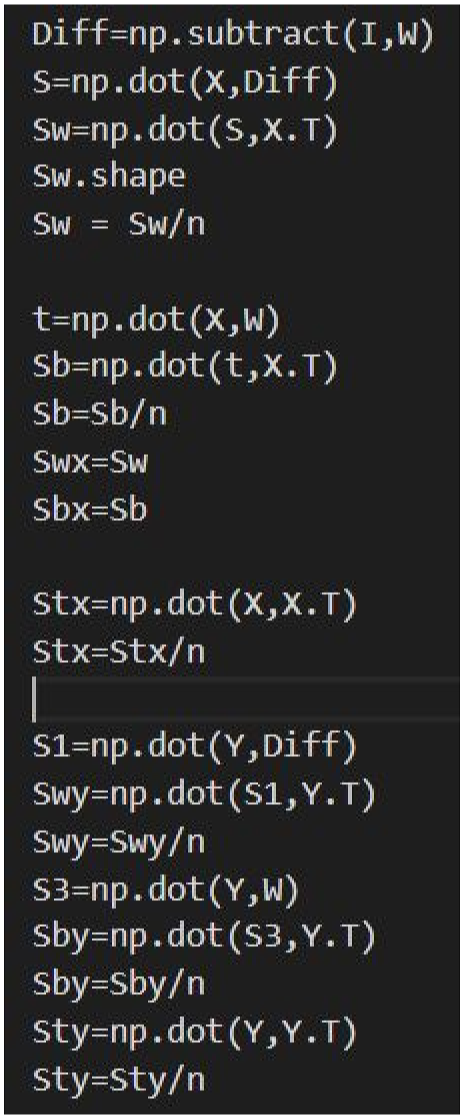
eq-1

2. Calculation of Scatter Matrices S_b, S_w, S_t where S_b, S_w, and S_t 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 the number of features in the 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
```

Similar to CCA, the optimization problem of criterion can be written as

$$\max_{w} \quad w^T S_b w$$
$$\text{s.t.} \quad w^T S_t w = 1.$$

eq-2

The optimal vector w is the eigenvector corresponding to the maximum eigenvalue of $S_t^{-1} S_b$.

3. Since MLDA is combination of LDA and CCA hence the final maximization equation of MLDA will be formed by a combination of eq-1 and eq-2.

$$\max_{w_x, w_y} \quad w_x^T S_{b_x} w_x + w_y^T S_{b_y} w_y + 2\gamma w_x^T C_{xy} w_y$$
$$\text{s.t.} \quad w_x^T S_{t_x} w_x = 1, \; w_y^T S_{t_y} w_y = 1$$

Eq-3

Using the Lagrangian multiplier technique, Eq-3 can be solved by a generalized multivariate eigenvalue problem in the following form:

$$\begin{bmatrix} S_{b_x} & \gamma C_{xy} \\ \gamma C_{yx} & S_{b_y} \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix} = \begin{bmatrix} S_{t_x} & 0 \\ 0 & S_{t_y} \end{bmatrix} \begin{bmatrix} \lambda_x w_x \\ \lambda_y w_y \end{bmatrix}$$

(Consideration $\gamma = 1$)

In order to obtain a closed-form solution, the constraints in the equation 3 can be coupled with $\sigma = (\mathrm{tr}(S_{tx})/\mathrm{tr}(S_{ty}))$, such that the constraints are transformed into a single constraint

$$\text{s.t. } W_X^T S_{tx} w_x + \sigma W_Y^T S_{ty} w_y = 1$$

Hence this equation constraints will be used further in our optimization problem and final vector w will be found out which then will be projected on a smaller subspace.

Documentation of API

- **Package Organization**

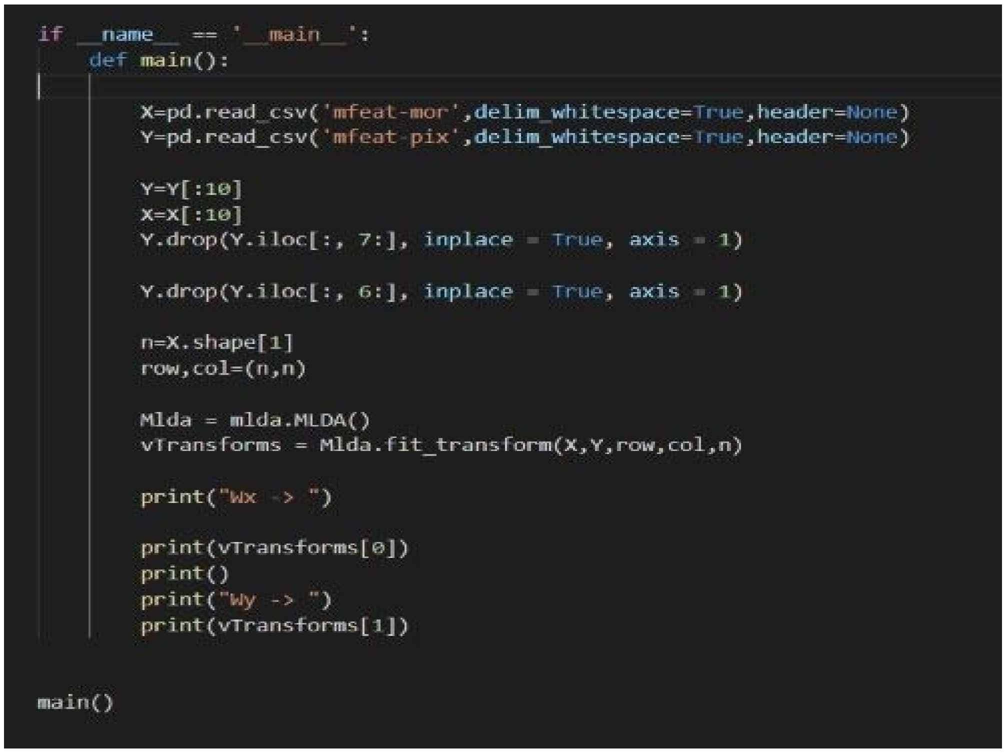
  mlda.py : This python file contain different method implementations like calculation of covariance matrix and scatter matrix
  DataScience_MLDA.ipynb : This is the jupyter notebook which contains the calculation of various parameters on sample dataset
  test.py : This python file contains the main method and code testing using sample data and it outputs the transformations
  Dataset: mfeat-mor , mfeat-pix
  These are sample dataset for 2 views

- **Parameters in mlda.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

- **Methods in mlda.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. **fit_transform :** It takes arguments as the first view,second view and dataset parameters and returns an array of projections(wx,wy) calculated by solving Lagrange duality equation :

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

- **Example**



```
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,col=(n,n)

        Mlda = mlda.MLDA()
        vTransforms = Mlda.fit_transform(X,Y,row,col,n)

        print("Wx -> ")

        print(vTransforms[0])
        print()
        print("Wy -> ")
        print(vTransforms[1])


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.MLDA utilizes the principle of CCA and LDA.

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

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

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