SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE

DEPARTMENT OF COMPUTER SCIENCE

DATA SCIENCE PROJECT 2

SPCCA - Sparsity Preserving Canonical Correlation Analysis
LPCCA - Locality Preserving Canonical Correlation Analysis

Submitted By:

Rithik Sachdev

Shreya Mishra

Shekhar Sharma

Submitted To:

Prof. S.Gupta

Indore, India

13th December 2020

Contents

1 Introduction 2
1.1 Multi View Learning 2
1.2 CCA 3
1.3 2D-LPCCA 4
1.4 2D-SPCCA.....4
2 Mathematical Formulation.....5
2.1 2D-LPCCA 5
2.2 2D-SPCCA 7
3 Algorithm.....8
4 API Documentation 10
5 References.....14

Introduction

Dimensionality reduction is an important procedure in the field of machine learning and pattern recognition along with data classification. The different algorithms such as K-nearest neighbours (KNN), Support vector machine, naive bayes, and etc. have been used for applications such as pattern recognition, image retrieval, text analysis and retrieval, and bioinformatics but the data used always have a high dimensionality which requires a lot of processing, memory and time. Moreover, a high dimension of data could lead to a curse of dimensionality. Therefore, dimensionality reduction strategy is used.

One of the major approaches involved in dimensionality reductions has been the computation of principal component analysis wherein higher dimensions can be reduced to lower dimensions using the concept of covariance matrix, eigenvalues and eigenvectors. It also gives details regarding the most important variable for clustering the graph and accuracy of the 2-dimensional graph. However, the use of principal components for reducing the dimensions in multi-view learning is limited.

PCA is a vector-based method, in which the image matrices need to be transformed into vectors before further computation. In order to use the image matrix directly, some two-dimensional based (2D-based) methods have been discovered such as 2D-PCA, 2-D LPCCA, 2-D SPCCA, etc. Here, the image vector is used directly rather than reshaping it prior to transformations.

1.1 Multi View learning

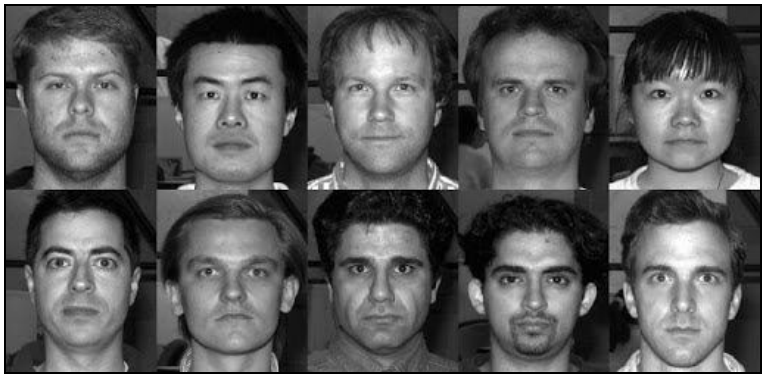
Nowadays, with the rapid development of the Internet, vast amounts of data with multiple views, i.e., sets of features are available.

Multi-view learning refers to fusion, analysis and exploitation of data from multiple data sources or simply from different views of the same data source, in order to achieve higher learning performance and accuracy.

Principal component analysis solely computes correlation for a single view, this is where Canonical Correlation Analysis comes into play.

1.2 CCA

CCA is being widely exploited for the purpose of image recognition, pattern recognition, image recovery, etc. Here, we have worked with the face database provided by Yale University.



Canonical Correlation Analysis was first proposed by Hotelling in 1936 and aimed at finding projection pairs within multiple views and thereby maximizing their correlation. Basically, it is used to compare a set of related variables with another set of related variables, that could be continuous or categorical. As we know, multiple regression can easily be used for finding correlation among different views but because of collinearity(Interaction between variables of the same set), it would not be able to handle the complexity of computations.

One of its advancement has been the General Canonical Correlation Analysis (GCCA) where we first extract two groups of low-dimensional feature vectors from two high-dimensional data sets, then fuse them using a feature fusion method which is either serial or parallel. However, the traditional CCA comes with certain drawbacks-

- It calculates only the linear correlation between two views whereas in many applications, there may be non-linear relationships between the views.

- It takes only 2 views into account.
- There is a wastage of information in CCA as it is an unsupervised algorithm, hence the labels provided in supervised classification are totally ignored by it.

Over the past few years, a large number of approaches have been proposed to deal with the complex nonlinear relationships between different views. Here, we aim to study the Locality preserving canonical correlation analysis (LPCCA) and Sparsity preserving canonical correlation analysis (SPCCA) frameworks.

1.3 2D-LPCCA:

Locality preserving canonical correlation analysis incorporates the information of the neighbours in a multi-view data into CCA. It keeps the neighborhood structure of the data while reducing the dimensionality, and reveals the intrinsic structure information of data. LPCCA assumes that the corresponding instances of different views should be as close as possible in the common latent space. It defines a laplacian matrix S to incorporate the relationship between samples, wherein the larger the entry in S , the larger sample correlation between samples.

1.4 2D-SPCCA:

As we know sparsity is an important way to improve the generalization capability of the model, it can also prevent the weight matrix from suffering from the difficulty of parameter selection as in the case of 2D-LPCCA, hence we can use sparse representation to design the weight matrix.

The reconstructive weight matrix S^x (or S^y) contains discriminant information naturally, and the entry S^x_{ij} represents the contribution of each X_j to reconstruct X_i. In other words, if the entry S^x_{ij} is bigger, the sample X_j is more important for reconstructing X_i.

Mathematical Formulation:

2.1 2D LPCCA:

Let $X = \{ X_1, \ldots, X_N \}$, $X_i \in R^{m \times p}$ and $Y = \{ Y_1, \ldots, Y_N \}$, $Y_i \in R^{m \times q}$ be two groups of images

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \sum_{i,j} G^x_{ij} (X_i - X_j)^T G^y_{ij} (Y_i - Y_j) \beta \\ s.t. \quad & \alpha^T \sum_{i,j} (G^x_{ij})^2 (X_i - X_j)^T (X_i - X_j) \alpha = 1, \\ & \beta^T \sum_{i,j} (G^y_{ij})^2 (Y_i - Y_j)^T (Y_i - Y_j) \beta = 1, \end{aligned}$$

where $G^x = \{ G^x_{ij} \}^N_{i,j=1}$ and $G^y = \{ G^y_{ij} \}^N_{i,j=1}$ are similarity matrices on graph G. There are many choices to define the similarity matrices, some of them are listed as follows:

(a) 0–1 method: $G_{ij} = 1$ if x_i and x_j are neighbors, otherwise $G_{ij} = 0$.
(b) Dot-product method: $G_{ij} = x_i^T x_j$ if x_i and x_j are neighbors, $G_{ij} = 0$ otherwise.
(c) Heat kernel method: $G_{ij} = e^{-||x_i - x_j||^2/t}$ if x_i and x_j are neighbors, $G_{ij} = 0$ otherwise.
(d) Cosine similarly method: $G_{ij} = \frac{x_i^T x_j}{||x_i|| \, ||x_j||}$ if x_i and x_j are neighbors, $G_{ij} = 0$ otherwise.

After some algebraic manipulations, we can get:

$$\begin{aligned} & \alpha^T \sum_{i,j} G^x_{ij} (X_i - X_j)^T G^y_{ij} (Y_i - Y_j) \beta \\ & = \alpha^T \sum_{i,j} G^x_{ij} G^y_{ij} (X_i^T Y_i + X_j^T Y_j - X_i^T Y_j - X_j^T Y_i) \beta \\ & = 2\alpha^T \sum_{i,j} G^x_{ij} G^y_{ij} X_i^T Y_i \beta - 2\alpha^T \sum_{i,j} G^x_{ij} G^y_{ij} X_i^T Y_j \beta \\ & = 2\alpha^T \sum_{i,j} G^{xy}_{ij} X_i^T Y_i \beta - 2\alpha^T \sum_{i,j} G^{xy}_{ij} X_i^T Y_j \beta \end{aligned}$$

where $G^{xy} = G^x \circ G^y$, the symbol $\circ$ denotes an operator such that $(G^x \circ G^y)_{ij} = G^x_{ij} G^y_{ij}$ for matrices G^x and G^y with the same size and G^x_{ij} denotes the ith row jth column entry of matrix G^x. $X = \{ X_1, \ldots, X_N \}$, $Y = \{ Y_1, \ldots, Y_N \}$, D^{xy} is a diagonal matrix with the ith diagonal entry being the ith row sum or column sum of G^{xy}, i.e., $D^{xy} = \text{diag}\left(\sum_{j=1}^{N} G^{xy}_{1j}, \ldots, \sum_{j=1}^{N} G^{xy}_{Nj} \right) = \text{diag}\left(\sum_{i=1}^{N} G^{xy}_{1i}, \ldots, \sum_{i=1}^{N} G^{xy}_{Ni} \right)$, and

$$G^{xy} \otimes I_m = \begin{bmatrix} G^{xy}_{11} I_m & \cdots & G^{xy}_{1N} I_m \\ \vdots & \cdots & \vdots \\ G^{xy}_{N1} I_m & \cdots & G^{xy}_{NN} I_m \end{bmatrix}$$

Then the maximization problem becomes

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \mathcal{X}^T (L^{xy} \otimes I_m) \mathcal{Y} \beta \\ s.t. \quad & \alpha^T \mathcal{X}^T (L^{xx} \otimes I_m) \mathcal{X} \alpha = 1, \\ & \beta^T \mathcal{Y}^T (L^{yy} \otimes I_m) \mathcal{Y} \beta = 1, \end{aligned}$$

where $L^{xx} = D^{xx} - G^{xx}$, $L^{yy} = D^{yy} - G^{yy}$, $G^{xx} = G^x \circ G^x$, $G^{yy} = G^y \circ G^y$, D^{xx} and D^{yy} are diagonal matrices whose definitions are similar to D^{xy}.

Since L^{xy},L^{xx} and L^{yy} are all symmetric and $L^{xy} = L^{yx}$,thus $L^{xy} \otimes I_m$,$L^{xx} \otimes I_m$ and $L^{yy} \otimes I_m$ are also symmetric matrices, and $L^{xy} \otimes I_m = L^{yx} \otimes I_m$. Using Lagrange multiplier method, problem can be solved by the following generalized eigenvalue problems

Once we obtained A = [$\alpha_1, \ldots, \alpha_d$] and B = [$\beta_1, \ldots, \beta_d$] which are eigenvectors corresponding to first d largest eigenvalues in Eq.(15)

$$\begin{aligned} &\hat{G_{xy}}(\hat{G_y})^{-1}\hat{G_{yx}}\alpha = \lambda^2\hat{G_x}\alpha, \\ &\hat{G_{yx}}(\hat{G_x})^{-1}\hat{G_{xy}}\beta = \lambda^2\hat{G_y}\beta, \end{aligned} \tag{15}$$

where $\hat{G_{xy}} = \mathcal{X}^T(L^{xy} \otimes I_m)\mathcal{Y}$, $\hat{G_x} = \mathcal{X}^T(L^{xx} \otimes I_m)\mathcal{X}$, $\hat{G_y} = \mathcal{Y}^T(L^{yy} \otimes I_m)\mathcal{Y}$ and $\hat{G_{yx}} = \hat{G_{xy}}^T$.

2.2 2D SPCCA:

we can get two sparse weight matrices S^x and S^y for the two sets of variables. Then the optimization

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \sum_{i,j} S_{ij}^x (X_i - X_j)^T S_{ij}^y (Y_i - Y_j)\beta \\ s.t. \quad & \alpha^T \sum_{i,j} (S_{ij}^x)^2 (X_i - X_j)^T (X_i - X_j)\alpha = 1, \\ & \beta^T \sum_{i,j} (S_{ij}^y)^2 (Y_i - Y_j)^T (Y_i - Y_j)\beta = 1, \end{aligned}$$

problem of 2D-SPCCA can be formulated as follows:

Using the similar method below, we can get the optimal solution.

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \sum_{i,j} G_{ij}^x (X_i - X_j)^T G_{ij}^y (Y_i - Y_j)\beta \\ s.t. \quad & \alpha^T \sum_{i,j} (G_{ij}^x)^2 (X_i - X_j)^T (X_i - X_j)\alpha = 1, \\ & \beta^T \sum_{i,j} (G_{ij}^y)^2 (Y_i - Y_j)^T (Y_i - Y_j)\beta = 1, \end{aligned}$$

Algorithm:

Algorithm for 2D-Locality Preserving Canonical Correlation Analysis (LPCCA)

Input : Training Data Set X,Y

Output : Projection Vectors A,B

Maximization Problem:

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \sum_{i,j} (X_i - X_j)^T (Y_i - Y_j)\beta \\ s.t. \quad & \alpha^T \sum_{i,j} (X_i - X_j)^T (X_i - X_j)\alpha = 1, \\ & \beta^T \sum_{i,j} (Y_i - Y_j)^T (Y_i - Y_j)\beta = 1. \end{aligned}$$

The algorithm that has been followed is:

1. Construct Similarity Matrix G^x and G^y using -

   a. Method 1 - Cosine Similarity : In this method, the G_{ij} is calculated as follows:

   $$G_{ij} = \frac{x_i^T x_j}{||x_i|| \cdot ||x_j||}$$ If x_i and x_j are neighbours,

   $$G_{ij} = 0$$ Otherwise

   b. Method 2 - Heat kernel method : In this method, the G_{ij} is calculated as follows:

   $$G_{ij} = e^{-||x_i - x_j||^2/t}$$ If x_i and x_j are neighbours,

   $$G_{ij} = 0$$ Otherwise

   c. Method 3 $*$ - Dot-product Method : In this method, the G_{ij} is calculated as follows:

   $$G_{ij} = x_i^T x_j$$ If x_i and x_j are neighbours,

   $$G_{ij} = 0$$ Otherwise

d. Method 4 ⁎ - 0-1 Method : In this method, the G_{ij} is calculated as follows:

$\mathbf{G_{ij} = 1,}$ If x_i and x_j are neighbours,

$\mathbf{G_{ij} = 0}$,Otherwise

2. After calculating G_{ij} calculate the value of D_{ij} according to the formula :

$$D^{xy} = \mathrm{diag}(\sum_{j=1}^{N} G_{1j}^{xy}, \ldots, \sum_{j=1}^{N} G_{Nj}^{xy}) = \mathrm{diag}(\sum_{i=1}^{N} G_{i1}^{xy}, \ldots, \sum_{j=1}^{N} G_{iN}^{xy})$$

3. After calculating the above values calculate the value of L_{ij} according to the formula :

$$L^{xx} = D^{xx} - G^{xx},\ L^{yy} = D^{yy} - G^{yy},\ G^{xx} = G^{x} \circ G^{x},\ G^{yy} = G^{y} \circ G^{y},$$

where the circle operator represents the element wise multiplication of the matrices.

4. Now the values of G cap can be calculated according to the formula :

$$\hat{G_{xy}} = \mathcal{X}^T (L^{xy} \otimes I_m)\mathcal{Y},\quad \hat{G_x} = \mathcal{X}^T (L^{xx} \otimes I_m)\mathcal{X},$$

5. Using the above calculated values, the following generalised eigenvalue equations need to be solved for α, β and λ^2 .

$$\hat{G_{xy}}(\hat{G_y})^{-1}\hat{G_{yx}}\alpha = \lambda^2 \hat{G_x}\alpha,$$
$$\hat{G_{yx}}(\hat{G_x})^{-1}\hat{G_{xy}}\beta = \lambda^2 \hat{G_y}\beta,$$

6. Now, the values of A and B can be calculated as follows, A=[$\alpha_1, \alpha_2 \ldots\ldots\alpha_N$] and B=[$\beta_1, \beta_2 \ldots\ldots\beta_N$].

⁎ : *the methods 3 and 4 are not implemented and kept for future work.*

API Documentation:

4.1 Package Organization

- class LPCCA2D

  Parameters:

  X: First View of the data

  Y: Second View of the data

  K: Number of K-nearest neighbours

- class LPCCA2DHeatKernel:

  Parameters:

  X: First View of the data

  Y: Second View of the data

  K: Number of K-nearest neighbours

4.2 Methods

__init__(X,Y,k)

To self-initialize the class. Here, self represents the class of the object itself. Parameters: X and Y are the multi views of the data. K represents the number of neighbor to be considered.(Default is always 5).

cosineSimilarity(self,X)

This function is used to define the similarity matrices. It takes one of the views as an input and returns its corresponding similarity matrix. This method is directly implemented using scipy.spatial.distance library (squareform imported).

The cosineSimilarity function works according to the formula:

$$G_{ij} = \frac{x_i^T x_j}{||x_i|| \cdot ||x_j||}$$ If x_i and x_j are neighbours,

$$G_{ij} = 0$$ Otherwise

meansquaredistance(self,X)（for class LPCCA2DHeatKernel ）

This function utilises mean square distance to find the neighbors of the input view（X in this case）. We have used the sqeuclidean library of Scipy. This library is used to compute the squared euclidean distance between 1-D Arrays.

The squared Euclidean distance between *u* and *v* is defined as

$$\left(\sum \left(w_i|(u_i - v_i)|^2\right)\right)^{||u-v||_2^2}$$

Parameters:
- **u** : *(N,) array_like*
  Input array.
- **v** : *(N,) array_like*
  Input array.
- **w** : *(N,) array_like, optional*
  The weights for each value in *u* and *v*. Default is None, which gives each value a weight of 1.0

Returns:
- **sqeuclidean** : *double*
  The squared Euclidean distance between vectors *u* and *v*.

similarity_matrix(X,k)

X：View of the data

K：Number of neighbors to be considered.

This function returns the similarity matrices that reflect the neighborhood information between the samples of X. It uses k-nearest neighbors algorithm to find the local neighbors. For example, if x1 is one of the k-nearest neighbors of x2 and vice-versa, then x1 and x2 are local neighbors.

This function also utilises the cosine_similarity function.

similarity_matrix(self,X,k)（for class LPCCA2DHeatKernel）

X：View of the data

K：Number of neighbors to be considered.

This function returns the similarity matrices that reflect the neighborhood information between the samples of X. It uses k-nearest neighbors algorithm to find the local neighbors. For example, if x1 is one of the k-nearest neighbors of x2 and vice-versa, then x1 and x2 are local neighbors. Here, we use the heat kernel method to find the similarity matrix of the given matrix.

Heat kernel method: $G_{ij} = e^{-||x_i - x_j||^2/t}$ if x_i and x_j are neighbors.
$G_{ij} = 0$ otherwise.

SimilarityMatricestoLMatrices(self)

This function is used to compute the 'L' matrices to make the further calculations easier (as defined in the research paper).

Then the maximization problem (12) becomes

$$\begin{aligned} \max_{\alpha,\beta} \quad & \alpha^T \mathcal{X}^T (L^{xy} \otimes I_m) \mathcal{Y} \beta \\ s.t. \quad & \alpha^T \mathcal{X}^T (L^{xx} \otimes I_m) \mathcal{X} \alpha = 1, \\ & \beta^T \mathcal{Y}^T (L^{yy} \otimes I_m) \mathcal{Y} \beta = 1, \end{aligned} \tag{14}$$

where $L^{xx} = D^{xx} - G^{xx}$, $L^{yy} = D^{yy} - G^{yy}$, $G^{xx} = G^x \circ G^x$, $G^{yy} = G^y \circ G^y$, D^{xx} and D^{yy} are diagonal matrices whose definitions are similar to D^{xy}

lpcca2d_Covariance_matrices (self)

This function is used to find the 'G cap' variables as explained in the research paper.

The formula simulated in this function is :

where $\hat{G_{xy}} = \mathcal{X}^T (L^{xy} \otimes I_m) \mathcal{Y}$, $\hat{G_x} = \mathcal{X}^T (L^{xx} \otimes I_m) \mathcal{X}$, $\hat{G_y} = \mathcal{Y}^T (L^{yy} \otimes I_m) \mathcal{Y}$ and $\hat{G_{yx}} = \hat{G_{xy}}^T$.

fit (self)

This function is used to solve the generalized eigenvalue problem aad described in the paper. The equation can be given as:

The formula simulated in this function is :

$$\begin{aligned} \hat{G_{xy}} (\hat{G_y})^{-1} \hat{G_{yx}} \alpha &= \lambda^2 \hat{G_x} \alpha, \\ \hat{G_{yx}} (\hat{G_x})^{-1} \hat{G_{xy}} \beta &= \lambda^2 \hat{G_y} \beta, \end{aligned} \tag{15}$$

The equation is of the form:

$$Aw = \lambda Bw$$

Where, A and B are similar matrices , w is the eigenvector and λ is the eigenvalue. The equation can be solved by the help of the eigh function from the scipy.linalg package.

fit_transform(self):

The dimension reduction technique of 2DLPCCA is applied. Eigen vectors are computed using the fit() function and are finally used to find the projection vectors by multiplying the respective vectors with the different views (X and Y).

4.3 Dependencies (Python libraries):

- **Pandas**

  The package can be found at - https://pandas.pydata.org/

- **Numpy**

  The package can be found at - https://numpy.org/

- **SciPy**

  The package can be found at - https://www.scipy.org/

- **Sklearn**

  The package can be found at - https://scikit-learn.org

References:

- *et. al.* Xizhan Gao, Quansen Sun, Haitao Xu, Yanmeng Li, 2D-LPCCA and 2D-SPCCA: Two new canonical correlation methods for feature extraction, fusion and recognition. Neurocomputing, *Journal homepage: www.elsevier.com/locate/neucom*
- *et. al. Chenfeng Guo,* Canonical Correlation Analysis Based Multi-View Learning An Overview, 2019,

  Hosted on: www.arxiv.org
- *et. al. Jing Zhao,* Multi-view learning overview Recent progress and new challenges, Information-Fusion, 2017,

  Journal homepage: www.elsevier.com/locate/inffus
- *et. al.* Chen Zu, Canonical sparse cross-view correlation analysis, 2016,

  Journal homepage: *www.elsevier.com/locate/neucom*