Least Squares Canonical Correlation Analysis (LS-CCA)

Sarvagna Shukla (0801CS171067)
sarvagna.shukla2107@gmail.com

December 10, 2020

Contents

Chapter 1
Introduction

1.1 Abstract

Canonical Correlation Analysis (CCA) is a well-known technique for finding the correlations between two sets of multi-dimensional variables. It projects both sets of variables into a lower-dimensional space in which they are maximally correlated. CCA is commonly applied for supervised dimensionality reduction, in which one of the multi-dimensional variables is derived from the class label. It has been shown that CCA can be formulated as a least squares problem in the binary class (two views) case.

1.2 LSCCA

Canonical correlation analysis (CCA) is a well-known technique in multivariate statistical analysis to find maximally correlated projections between two data sets. CCA only leverages two views of the data, extending it to multiple views is achieved with Least Squares based CCA. In practice, the multi-class and multi-label problems are more prevalent. The solution to least squares problems can be obtained by solving a linear system of equations. Furthermore, the least squares formulation can be readily extended using the regularization technique.

Chapter 2
Mathematical Formulation

2.1 Formulation

The LSCCA finds a set of M projections so that the correlation between paired data sets is maximized. Here, M = number of data sets (number of views). Let $\boldsymbol{X_k} \in C^{N\times m_k}$ for $k = 1, . . . , M$ be full-rank matrices. If we denote the successive canonical vectors and variables as $\mathbf{h}_k^{(i)}$ and $\mathbf{z}_k^{(i)} = \mathbf{X}_k\mathbf{h}_k^{(i)}$, respectively; and the estimated cross-correlation matrices as $\mathbf{R}_{kl} = \mathbf{X}_k^H \mathbf{X}_l$, then our CCA generalization can be formulated as the problem of sequentially maximizing the generalized canonical correlation

$$\rho^{(i)} = \frac{1}{M} \sum_{k=1}^{M} \rho_k^{(i)},$$

where $\rho_{kl}^{(i)} = \mathbf{h}_k^{(i)\mathrm{H}}\, \mathbf{R}_{kl}\, \mathbf{h}_l^{(i)}$ and

$$\rho_k^{(i)} = \frac{1}{M-1} \sum_{l=1}^{M} \rho_{kl}^{(i)}$$

In this case, the energy constraint to avoid trivial solutions is

$$\frac{1}{M} \sum_{k=1}^{M} \mathbf{h}_k^{(i)\mathrm{H}}\, \mathbf{R}_{kk}\, \mathbf{h}_k^{(i)} = 1, \quad ...(1)$$

and defining $\mathbf{z}^{(i)} = \frac{1}{M}\sum_{k=1}^{M} \mathbf{z}_k^{(i)}$, the orthogonality constraints are, for $i \neq j$

$$\mathbf{z}^{(i)\mathrm{H}}\, \mathbf{z}^{(j)} = 0. \quad \text{...(2)}$$

Unlike the two data set case, here it is interesting to point out that, for $M > 2$, the energy constraint (1) is not equivalent to $\mathbf{h}_k^{(i)\mathrm{H}}\, \mathbf{R}_{kk}\, \mathbf{h}_k^{(i)} = \mathbf{1}$ for all k. However, as we will see later, the solutions of the proposed generalization with $M = 2$ data sets are the same of the conventional formulation of the CCA problem and then they satisfy $\mathbf{z}_k^{(i)\mathrm{H}}\, \mathbf{z}_l^{(j)} = \mathbf{0}$ and $\mathbf{h}_k^{(i)\mathrm{H}}\, \mathbf{R}_{kk}\, \mathbf{h}_k^{(i)} = \mathbf{1}$ for $k,l = 1, 2$ and $\mathrm{i} \neq \mathrm{j}$.

The proposed CCA (referred to as **LS-CCA**) can be rewritten as a function of distances. Specifically, to extract the i-th CCA eigenvector, the LS-CCA problem consists on minimizing, with respect to the M canonical vectors $\mathbf{h}_k^{(i)}$, the following cost function

$$J^{(i)} = \frac{1}{2M(M-1)} \sum_{k,l=1}^{M} \| X_k h_k^{(i)} - X_l h_l^{(i)} \|^2$$
$$= \frac{1}{M} \sum_{k=1}^{M} \| z_k^{(i)} \|^2 - \rho^{(i)},$$

subject to (1) and (2), which implies $J^{(i)} = 1 - \rho^{(i)}$.

The solutions of LS-CCA problem can be obtained by the method of Lagrange multipliers whose solutions are determined by the following GEV problem

$$\frac{1}{M-1}\,(\mathbf{R} - \mathbf{D})\mathbf{h}^{(i)} = \rho^{(i)}\mathbf{D}\mathbf{h}^{(i)} \qquad ...(3)$$

where $\mathbf{h}^{(i)} = [\mathbf{h}_1^{(i)T}, \ldots, \mathbf{h}_M^{(i)T}]^T$ stacks the canonical vectors,

$$R = \begin{bmatrix} R_{11} & \ldots & R_{1M} \\ . & . & . \\ . & . & . \\ . & . & . \\ R_{M1} & \ldots & R_{MM} \end{bmatrix}, \quad D = \begin{bmatrix} R_{11} & \ldots & 0 \\ . & . & . \\ . & . & . \\ . & . & . \\ 0 & \ldots & R_{MM} \end{bmatrix}$$

and $\rho^{(i)}$ is a generalized eigenvalue. Then, the LS-CCA solutions are obtained as the eigenvectors associated with the largest eigenvalues of (3). Here, we must note that, in the case of $M = 2$ data sets, the GEV problem (9) is reduced to CCA.

Chapter 3
Algorithm

3.1 LSCCA algorithm

Input: multiview training datasets $X_1 \in R^{N\times m_1}$, $X_2 \in R^{N\times m_2}$, ..., $X_M \in R^{N\times m_M}$ where $m_1, m_2, ..., m_M$ are the dimensionality of $X_1, X_2, ..., X_M$ views respectively.

Output: Weight matrices $w_1, w_2, \ldots, w_M$

Algorithm:

1. Reduce dimensions of all datasets to $\min(m_1, m_2, \ldots, m_M)$
2. Normalize all datasets $X_1, X_2, ..., X_M$, if it is needed.
3. Create a matrix R that will have all the covariance matrices having all the elements.
   $$R_{ij} = \frac{1}{M-1} X_i^T X_j$$
4. Initialize a weight array W having all the weight matrices.
   $$W_i = R_{ii}$$
5. In a loop, compute a matrix T
   $$T_{ij} = R_{ii}^{-\frac{1}{2}} R_{ij} R_{jj}^{-\frac{1}{2}}$$
6. Perform SVD decomposition on each matrix T,
   $$T_{ij} = UDV^T$$

7. Update the weights in W accordingly,

   $$W_i = W_i U$$

   $$W_j = W_j V^T$$

8. Finally, you have all the Weight Matrices in W.

Chapter 4

Documentation of API

4.1 Package Organization

class **LSCCA**(*views = 2, scale = True*)

Parameters
views : int, (default = 2)
scale : boolean, (default = True)

Attributes
views : int
scale : boolean
min_cols : int
W : array, [views]
rmat : array, [views, views]
tmat : array, [views, views]

4.2 Methods

fit (M)

Parameters: Array containing all views
Returns: Array containing all weight matrices

__init__ (views = 2, scale = True)

Initialize self.

normalize (M)

normalizing the data for all M views

reduce_dimensions (M, min_cols)

reducing the dimensions, i.e., features, so all M views have same number of features(columns)

Chapter 5

Example

5.1 Example 01

```
from LSCCA import LSCCA
import pandas as pd

# Declaring and Initializing all M views
m1 = pd.DataFrame([[0., 0., 1.], [1.,0.,0.], [2.,2.,2.], [3.,5.,4.]])
m2 = pd.DataFrame([[0.1, -0.2], [0.9, 1.1], [6.2, 5.9], [11.9, 12.3]])
m3 = pd.DataFrame([[0.3, -0.4], [1.9, 0.9], [4.3, 3.2], [8.4, 7.1]])

# taking all M views in a List
M = []
M.append(m1)
M.append(m2)
M.append(m3)

# Making an object by calling the constructor of LSCCA, and generating all
the weight matrices W
lscca = LSCCA(3, True)
W = lscca.fit(M)
```

```
# Printing all the weight matrices
count = 1
for i in W:
    print("w" + str(count) + ":")
    print(i, end="\n\n")
    count+=1
```

<u>Output</u>:

```
w1:
[[ 1.99351358 -0.16094602]
 [ 0.16094602  1.99351358]]

w2:
[[1.99712032 2.00000436]
 [1.99999564 1.99712905]]

w3:
[[ 1.99999966 -1.99934166]
 [ 1.99934098 -2.00000034]]
```

5.2 Example 02

```
from LSCCA import LSCCA
import pandas as pd


# Declaring and Initializing all M views
m1 = pd.DataFrame([[0., 0., 1.], [1.,0.,0.], [2.,2.,2.], [3.,5.,4.]])
m2 = pd.DataFrame([[0.1, -0.2], [0.9, 1.1], [6.2, 5.9], [11.9, 12.3]])
m3 = pd.DataFrame([[0.3, -0.4], [1.9, 0.9], [4.3, 3.2], [8.4, 7.1]])
m4 = pd.DataFrame([[0.9, -1.2], [1.8, 2.4], [-2.7, 3.6], [3.6, -4.8]])

# taking all M views in a List
M = []
M.append(m1)
M.append(m2)
M.append(m3)
M.append(m4)

# Making an object by calling the constructor of LSCCA, and generating all
the weight matrices W
lscca = LSCCA(4, True)
W = lscca.fit(M)

# Printing all the weight matrices
count = 1
for i in W:
    print("w" + str(count) + ":")
    print(i, end="\n\n")
    count+=1
```

Output:

```
w1:
[[-0.9949667  -0.88759171]
 [-0.88759171  0.9949667 ]]

w2:
[[-1.88426268e+00  1.30969766e-03]
 [-1.88426261e+00 -1.40117517e-03]]

w3:
[[-3.05977728e-04 -1.88530758e+00]
 [ 3.15032940e-04 -1.88530758e+00]]

w4:
[[-1.6690699   0.30841103]
 [ 1.69488245  0.09102281]]
```

Chapter 6
Learning Outcome

6.1 Learning

CCA is widely used in Multi-View Learning. However, the traditional CCA has one of the limitations that it can only handle two views. LS-CCA, which can handle more than two views, has been proposed for remedy. The solution to LS-CCA can be obtained by solving a linear system of equations. Furthermore, the least squares formulation can be readily extended using the regularization technique. We learnt the approach and mathematical formulation for the LS-CCA, and also identified and implemented the algorithm for more than two views. Moreover, when the number of views is two, LS-CCA degrades to CCA.

Appendix A
References

1. Javier Vıa. Ignacio Santamarıa. Jesus Perez. A learning algorithm for adaptive canonical correlation analysis of several data sets. Department of Communications Engineering, University of Cantabria, Spain, 2006

2. Chenfeng Guo. Dongrui Wu. Canonical Correlation Analysis (CCA) Based Multi-View Learning: An Overview. 2019.

3. Liang Sun. Shuiwang Ji. Jieping Ye. A Least Squares Formulation for Canonical Correlation Analysis, Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287 USA, 2008

4. Viivi Uurtio, João M. Monteiro, Jaz Kandola, John Shawe-Taylor, Delmiro Fernandez-Reyes, and Juho Rousu, 2017. A Tutorial on Canonical Correlation Methods. ACM Comput. Surv. 50, 6, Article 95 (October 2017), 33 pages. DOI: 10.1145/3136624