Regularised Canonical Correlation Analysis, (RCCA)

Submitted By

Anay Gupta - 0801CS171010
Aatmik Jain - 0801CS171003
Shashwat Jain - 0801CS171073

Contents

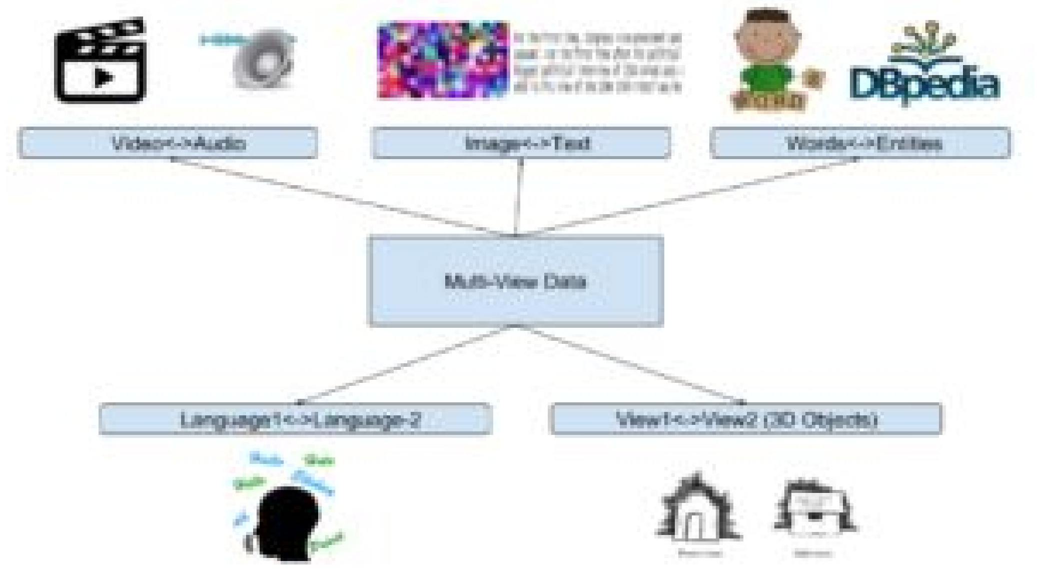
Chapter 1

Introduction

1.1 Multi View Learning

Multi-View Learning (MVL) is a machine learning framework where data is represented by multiple distinct feature groups, and each feature group is referred to as a particular view. It's aim is to improve the generalised performance and is also referred to as data fusion or data integration.

A multi view can be understood as



MVL approaches can be divided into three major categories

1) Co-training : which exchanges discriminative information between two views by training the two models alternately.

2) Multi-kernel learning : which maps data to different feature spaces with different kernels, and then combines those projected features from all spaces.

3) Subspace learning which assumes all views are generated from a latent common space where shared information of all views can be exploited.

We will be focusing on subspace learning which includes Regularised Canonical Correlation Analysis (RCCA), Regularised Generalised Canonical Correlation Analysis (RGCCA) and Multi View Canonical Correlation Analysis.

1.2 RCCA

CCA is a method for finding linear correlational relationships between two or more multidimensional datasets. CCA finds a canonical coordinate space that maximizes correlations between projections of the datasets into that space.

Regularized CCA (RCCA) is an improved version of CCA which in the presence of insufficient training data prevents overfitting by using a ridge regression optimization scheme. Denote p and q as the number of features in X and Y, and n as the sample size. When $n << p$ or $n << q$, the features in X and Y tend to be highly collinear. This leads to ill-conditioned matrices C_{xx} and C_{yy} , which denote the covariance matrix of X with itself and Y with itself, such that their inverses are no longer reliable resulting in an invalid computation of CCA and an unreliable metaspace. The condition placed on the data to guarantee that C_{xx} and C_{yy} will be invertible is $n \geq p + q + 1$. However, that condition is usually not met in the bioinformatics domain, where samples (n) are usually limited, and modern technology has enabled very high dimensional data streams to be routinely acquired resulting in very high dimensional feature sets (p and q). This creates a need for regularization, which works by adding small positive quantities to the diagonals of C_{xx} and C_{yy} to guarantee their invertibility.

Chapter 2

Mathematical Formula

The CCA finds a set of two projections so that the correlation between paired data sets is maximized in the common feature subspace. Let X = $[X_1, X_2, ... X_n]$ ε $R^{d_x \times N}$ and Y = $[Y_1, Y_2, ... Y_n]$ ε $R^{d_y \times N}$ are the two input vectors. Where d_x and d_y are the dimensionality of X and Y views, respectively, and N is the total number of samples. CCA aims to find two projection vectors w_x and w_y that can maximize the correlation coefficient between $w_x^T X$ and $w_y^T Y$ say ρ .

$$\rho(w_x^T X, w_y^T Y) = \frac{w_x^T X Y^T w_y}{\sqrt{(w_x^T X X^T w_x)(w_y^T Y Y^T w_y)}} \qquad (1)$$

Since (1) is invariant to the scaling of w_x and w_y , it can be transformed into the following constrained for

$$max_{w_x, w_y} = w_x^T X Y^T w_y$$

Given $w_x^T X X^T w_x = 1$ and $w_y^T Y Y^T w_y = 1$

When the feature dimensionality is high, especially when d_x > N (or d_y > N), the covariance matrix $X X^T$ (or $Y Y^T$) is singular, and hence the optimization problem is underdetermined. This is solved through RCCA. Regularizations can be added to the covariance matrices to remedy this problem by introducing non negative regularization coefficients r_x and r_y as

$$\hat{\Sigma}_{xx} = \frac{1}{N} X X^T + r_x I,$$
$$\hat{\Sigma}_{yy} = \frac{1}{N} Y Y^T + r_y I,$$

To calculate w_x and w_y we need to solve the following generalized eigenvalue decomposition problem.

$$\begin{bmatrix} \mathbf{0} & \Sigma_{xy} \\ \Sigma_{yx} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w_x} \\ \mathbf{w_y} \end{bmatrix} = \lambda \begin{bmatrix} \hat{\Sigma}_{xx} & \mathbf{0} \\ \mathbf{0} & \hat{\Sigma}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{w_x} \\ \mathbf{w_y} \end{bmatrix}$$

where

$$\Sigma_{xy} = \frac{1}{N} X Y^T,$$
$$\Sigma_{yx} = \frac{1}{N} Y X^T.$$

Chapter 3

Algorithm

Input: list of two views of training dataset, $X \in \mathbb{R}^{d1xn}$, $Y \in \mathbb{R}^{d2xn}$, where d_1 and d_2 are the dimensionality of X and Y views.
Output: weights ($[w_x\ w_y]^T$) and correlation between variates

1. Compute Σ_{xy}, Σ_{yx}, Σ_{xx}, Σ_{yy}
2. $\begin{bmatrix} \mathbf{0} & \Sigma_{xy} \\ \Sigma_{yx} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w_x} \\ \mathbf{w_y} \end{bmatrix} = \lambda \begin{bmatrix} \hat{\Sigma}_{xx} & \mathbf{0} \\ \mathbf{0} & \hat{\Sigma}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{w_x} \\ \mathbf{w_y} \end{bmatrix}$
    a. Compute left : $\begin{bmatrix} \mathbf{0} & \Sigma_{xy} \\ \Sigma_{yx} & \mathbf{0} \end{bmatrix}$
    b. Compute right: $\begin{bmatrix} \hat{\Sigma}_{xx} & \mathbf{0} \\ \mathbf{0} & \hat{\Sigma}_{yy} \end{bmatrix}$
3. Make left and right matrices symmetric
4. Solve the generalized eigenproblem in (2)
5. Obtain Eigenvalue: λ and Eigenvector: $[w_x\ w_y]^T$

Chapter 4
Documentation of API

4.1 Package Organization

class rcca.RCCA(n_comp=2,reg_param=0.1)

Parameters:

n_comp: the number of components
reg_param: regularization parameter

4.2 Methods

__init__(self,n_comp,reg_param):

To Initialize class
Where self represents the class object itself.
Parameters:

n_comp: the number of components (Default=2)
reg_param: regularization parameter (Default=0.1)

fit(self,data):

To fit the standardized data to RCCA so as to calculate weights and correlation of variates.
Parameters:

data: datasets in the form of list of length 2.

transform(self,data):

To get the reduced data with the weights associated with it by returning dot product of the standardized data and weights.
Parameters:

data: datasets in the form of list of length 2.

Chapter 5

Example 1

Input:

```
X = [[ 0.20605173  0.98662189  0.94325234 -0.87510997 -0.56730019  0.44088274
   0.71026552]
 [ 1.49014402 -1.53244402  0.77853998 -0.33613035  1.62787032 -1.71722515
   1.23909268]
 [-1.23440957  0.7873193  -0.14610033  1.69824255 -0.04768574  0.78349023
  -0.80122375]
 [-0.46178619 -0.24149717 -1.57569198 -0.48700223 -1.01288439  0.49285218
  -1.14813445]]
Y = [[ 0.63667996 -0.96875234  0.04466362  1.26061466 -0.11553741 -0.25707614
   0.38416636  0.3999494 ]
 [ 1.3036356  -0.96099212  1.61708986 -1.34665837  1.56442178 -1.49957281
   0.90277497 -1.47776481]
 [-1.05003452  0.5933509  -0.86488594 -0.50179929 -0.22966816  0.61159849
   0.40755374  1.27219094]
 [-0.89028104  1.33639356 -0.79686754  0.58784299 -1.2192162   1.14505045
  -1.69449506 -0.19437554]]
```

Output:

```
Reduced X = [[-0.06674136,  0.79987307],
       [-1.05679607, -0.37609887],
       [ 0.50034901,  0.27949292],
       [ 0.62318842, -0.70326711]]

Reduced Y = [[-0.07757339,  0.7852136 ],
       [-1.06407989, -0.34157419],
       [ 0.47288942,  0.26129317],
       [ 0.66876385, -0.70493258]]
```

Example 2

Input:

```
X = [[-0.67391964 -1.54819536 -1.35695821 -1.57551684 -0.45312978 -1.63984178
  -1.15178275]
 [ 1.5015488   1.24823521  1.39651441  1.20309764  1.40404458  0.96542181
   1.22121878]
 [-1.10254475  0.14753838 -0.34198113  0.16436097 -1.30347091  0.06202047
  -0.80270582]
 [ 0.27491559  0.15242177  0.30242494  0.20805822  0.35255611  0.6123995
   0.73326978]]
Y = [[-1.13330805 -1.53331638 -0.97998527 -1.50864538 -0.72838087 -1.57974697
  -1.29069487 -1.44969387]
 [ 1.23310141  1.0257892   1.63414196  0.7231984   1.48157184  1.02675446
   1.13722477  0.36804825]
 [-0.82128972 -0.22992303 -0.60568958 -0.27529961 -1.08042709 -0.10903872
  -0.64056895 -0.22662411]
 [ 0.72149635  0.73745021 -0.0484671   1.06074659  0.32723613  0.66203123
   0.79403905  1.30826973]]
```

Output:

```
Reduced X = [[-0.8659283 ,  0.61039765],
       [ 0.92323521,  0.28466866],
       [-0.35311979, -0.81557902],
       [ 0.29581288, -0.0794873 ]]
Reduced Y = [[-0.87475415,  0.55043539],
       [ 0.8618747 ,  0.43010532],
       [-0.39346037, -0.56955951],
       [ 0.40633983, -0.4109812 ]]
```

Chapter 6
Learning Outcomes

- Successfully analysed and implemented the Concepts of RCCA using two methods of fit and transform.
- Realised the Use of RCCA and it’s concepts in calculation of eigenvalues and subsequent weights.
- Used the methods in the package on our locally generated data and got satisfactory results as expected from the analysis of the mathematical equations.

References:

1. Pyrcca: Regularized Kernel Canonical Correlation Analysis in Python and Its Applications to Neuroimaging, Natalia Y. Bilenko1 and Jack L. Gallant 1, 2 * , doi: 10.3389/fninf.2016.00049
2. Canonical Correlation Analysis (CCA) Based Multi-View Learning: An Overview by Chenfeng Guo, Dongrui Wu, https://arxiv.org/abs/1907.01693
3. On the regularization of canonical correlation analysis, T. D. Bie and B. D. Moor, https://www.researchgate.net/publication/229057909_On_the_Regularization_of_Canonical_Correlation_Analysis