Discriminative Canonical Correlation Analysis (DisCCA)

Shelam Mehta(0801CS171076)
gs0801cs171076@sgsitsindore.in

December 20, 2020

Contents

Chapter 1
Introduction

1.1 Canonical Correlation Analysis

In statistics, canonical-correlation analysis (CCA), also called canonical variates analysis, is a way of inferring information from cross-covariance matrices. If we have two vectors $X = (X_1, ..., X_n)$ and $Y = (Y_1, ..., Y_m)$ of random variables, and there are correlations among the variables, then canonical-correlation analysis will find linear combinations of X and Y which have a maximum correlation with each other.

1.2 Discriminative CCA (DisCCA)

Traditional CCA is unsupervised. In supervised classification, we have the label information, which should be taken into consideration to help extract more discriminative features. Discriminative CCA (DisCCA) is one such approach. It maximizes the within-class similarity and minimizes the between-class similarities.

Chapter 2
Mathematical Formulation

2.1 Formulation

Given data in two views X and Y. Normalizing data into Zero mean and representing transformed data as $X^{\wedge}$ and $\hat{Y}$ respectively**.** Rearrange $X^{\wedge}$ and $\hat{Y}$ according to the classes such that tuples belonging to the same class are kept together

$\mathbf{X^{\wedge}=[x_1^{(1)},\ldots.,x_{n1}^{(1)},\ldots\ldots..,x_1^{(c)},\ldots..,x_{nc}^{(c)}]}$
$\mathbf{\hat{Y}=[y_1^{(1)},\ldots.,y_{n1}^{(1)},\ldots\ldots..,y_1^{(c)},\ldots..,y_{nc}^{(c)}]}$

Objective function of DisCCA is:

$\mathbf{max_{wx,wy}(w^T{}_x C_w w_y - w^T{}_x\, C_b w_y)}$ **(1)**
$\mathbf{S.t.\ \ w^T{}_x X^{\wedge} X^{\wedge T} w_x=1,\ \ w^T{}_y \hat{Y}\hat{Y}^T w_y=1}$

Where matrices $\mathbf{C_w}$ and $\mathbf{C_b}$ are within-class similarity and between-class similarity respectively and are defined as

$$C_{\mathrm{w}} = \sum_{i=1}^{c}\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} \mathbf{x}_k^{(i)}\left(\mathbf{y}_l^{(j)}\right)^T = \hat{X}A\hat{Y}^T,$$
$$C_{\mathrm{b}} = \sum_{i=1}^{c}\sum_{\substack{j=1\\ j\neq i}}^{c}\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} \mathbf{x}_k^{(i)}\left(\mathbf{y}_l^{(j)}\right)^T = -\hat{X}A\hat{Y}^T, \qquad \mathbf{(2)}$$

Here A is block matrix of form

A = [1 (n1×n1) . .
. .
1(ni×ni) . .
. .
1 (nc×nc)]

Using eqn(2), eqn (1) can be rewritten as:
$\mathbf{max_{wx,wy}(w^T{}_x X^{\wedge} A \hat{Y}^T w_y)}$ **(3)**
$\mathbf{S.t.\ \ w^T{}_x X^{\wedge} X^{\wedge T} w_x=1,\ \ w^T{}_y \hat{Y}\hat{Y}^T w_y=1}$

Solving this eqn by performing singular value decomposition on the matrix:

$\mathbf{T = \Sigma_{xx}^{-1/2}\Sigma_{xy}\Sigma_{yy}^{-½}}$

Where

$\mathbf{\Sigma_{xx}=}$ $(1/N)*XX^T+rx*I$

$\mathbf{\Sigma_{xy}=}$ $(1/N)*XAY^T$

$\mathbf{\Sigma_{yy}=}$$(1/N)*YY^T+ry*I$

For finding pseudo inverse root of $\mathbf{\Sigma_{xx}}$ and $\mathbf{\Sigma_{yy}}$ using

$$\mathbf{A^{(-1/2)}= P\Lambda P^T}$$

Where

P is matrix containing Eigen vectors of A in row form

Λ is diagonal matrix containing eigen values in diagonal

Chapter 3
Algorithm

- Converting X and Y into nd Array of dimension n*dx and n*dy
- Bringing all the tuples belonging to same class together in X and Y
- Replacing X and Y by X^T and Y^T respectively
- Mean normalizing X and Y
- Creating block diagonal matrix of order n*n
- Finding $\boldsymbol{\Sigma}_{xx}$, $\boldsymbol{\Sigma}_{yy}$, $\boldsymbol{\Sigma}_{xy}$ and $\boldsymbol{\Sigma}_{yx}$ using the formula given above
- Finding eigenvectors from eqn we get after solving Lagranges method using SVD
- Then multiplying these vectors by $\boldsymbol{\Sigma}^{-1/2}_{xx}$ and $\boldsymbol{\Sigma}^{-1/2}_{yy}$ respectively to get Wx and Wy

Chapter 4
Documentation of API

4.1 Package Organisation

```
from DisCCA import DisCCA
D_CCA=DisCCA()
```

4.2 Methods

DisCCA.fit(*X*: *numpy.array*, *Y*: *numpy.array*, *target*: *numpy.array*, output_dimensions: *int*)

to fit weight according to training data

Parameters

- X (*numpy ndarray*) – training data of view 1
- Y (*numpy ndarray*) – training data of view 2
- target (*array*) – list of target
- outdim_size (*int*) – output dimension of data get output of the model

Returns None

DisCCA.transform(*X*: *numpy.array*, *Y*: *numpy.array*)

to convert the data into new dimension

Parameters

- X (*numpy ndarray*) – data of view 1
- Y (*numpy ndarray*) – data of view 2

Returns transformed X and Y into new dimension.

DisCCA.fit(*X*: *numpy.array*, *Y*: *numpy.array*, *target*: *numpy.array*, output_dimensions: *int*)

to fit weight according to training data

Parameters

- X (*numpy ndarray*) – training data of view 1

- Y (*numpy ndarray*) – training data of view 2
- target (*array*) – list of target
- outdim_size (*int*) – output dimension of data get output of the model

Returns List containing transformed matrices .

DisCCA.get_within_class_similarity()

to get within-class similarity

Parameters

None

Returns within-class similarity

DisCCA.get_between_class_similarity()

to get between class similarity

Parameters

None

Returns between class similarity

Chapter 5

Example

```
# Program containing example use of DisCCA

from DisCCA import DisCCA
import pandas as pd

# Reading data
first_dataframe=pd.read_csv("first_view",delim_whitespace=True,h
eader=None)
second_dataframe=pd.read_csv("second_view",delim_whitespace=True
,header=None)

#Generating Target values
target=[]
for i in range(10):
    for j in range(200):
        target+=[i]

D_CCA=DisCCA()
D_CCA.fit(first_dataframe,second_dataframe,target, 47)

transformed_first_dataframe,transformed_first_dataframe=D_CCA.tr
ansform(first_dataframe, second_dataframe)
```

Chapter 6
Learning Outcome

Some of the learning I had after completion of the project are :

- I got to learn how to solve maxima and maxima function with some constraint using Lagrange multiplier
- The need for pseudo-inverse of a matrix and how to calculate it using SVD
- Comparison of normal CCA vs DisCCA

References

- [1907.01693] Canonical Correlation Analysis (CCA) Based Multi-View Learning: An Overview
- https://www.researchgate.net/publication/220765533_A_Novel_Method_of_Combined_Feature_Extraction_for_Recognition/link/548a37ec0cf214269f1ac38b/download