Kernel Canonical Correlation Analysis (KCCA)

Savi R Bhide (0801CS171069)
Shikhar Mahajan (0801CS171077)

December 13, 2020

Contents

Chapter 1

Introduction

1.1 Canonical Correlation Analysis (CCA)

CCA statistically finds the correlation between two sets of random variables X and Y(Hotelling, 1936). Denote

$$X = (x_1, \dots x_p) \in R^{N\times p},\ Y = (y_1, \dots y_q) \in R^{N\times q}.$$

X and Y can be two feature spaces, or a feature space and a label space. To obtain the correlation between the two sets of variables, CCA finds a linear projection u in the space of X, and a linear projection v in the space of Y to maximize the following sample correlation.

Such that the projected data u′X and v′Y have a maximum correlation.

$$\rho_{CCA} = \operatorname*{argmax}_{u\in R^p,\ v\in R^q} \frac{u'X'Yv}{\sqrt{(u'X'Xu)(v'Y'Yv)}}$$

1.2 Kernel Canonical Correlation Analysis (KCCA)

The KCCA provides a nonlinear extension of CCA, which catches the nonlinear correlation by mapping the data into a higher-dimensional feature space before performing CCA.

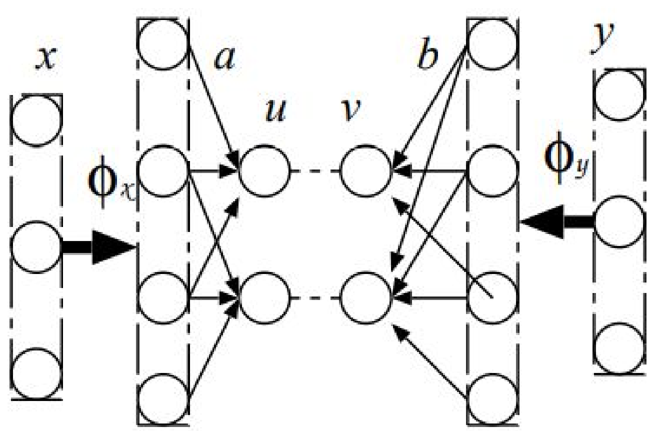


Figure 2: Kernel CCA

Chapter 2

Mathematical Formulation

2.1 Formulation

a. Input-X and Input-Y are normalized.

b. Normalized x and y are transformed into the Hilbert space,

φx(x) ∈ Hx and φy(y) ∈ Hy.

Objective function of KCCA is:

$$\max_{\mathbf{w}_x,\mathbf{w}_y} \mathbf{w}_x^T \Phi_x \Phi_y^T \mathbf{w}_y$$
$$s.t.\ \mathbf{w}_x^T \Phi_x \Phi_x^T \mathbf{w}_x = 1, \mathbf{w}_y^T \Phi_y \Phi_y^T \mathbf{w}_y = 1.$$

Expressing wx and wy as linear combinations of the columns of φx and φy, respectively.

wx = φx.a

wy = φy.b

where a and b are linear coefficients.

c. Let Kx = (φx^T).(φx) and Ky = (φy^T).(φy) be kernel matrices, i.e., (Kx)ij = κ (xi , xj), where κ is a kernel function, such as a radial basis function (RBF).

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2})$$

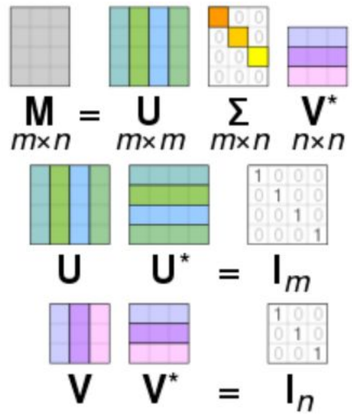
d. After applying kernel function, we can further add regularizations to kernel matrices to make it more stable.

Replacing Kx.Kx by Kx.Kx +rx.Kx and Ky.Ky by Ky.Ky +ry.Ky

Where rx, ry are regularization parameter

e. Now solving it by singular value decomposition to find a, b.

$$\begin{bmatrix} \mathbf{0} & K_x K_y \\ K_y K_x & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$
$$=\lambda \begin{bmatrix} K_x K_x + r_x K_x & \mathbf{0} \\ \mathbf{0} & K_y K_y + r_y K_y \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$



f. Calculating wx, wy will give the KCCA transformed matrices. Where, wx and wy are calculated as follows:

wx = φx.a

wy = φy.b

g. Finally transforming matrices to same space by projecting it by using following formula:

Xnew = (wx^T).X

Ynew = (wy^T).Y

Chapter 3

Algorithm

In this section, we give an overview of the KCCA algorithms where we formulate the optimization problem as a standard eigen problem.

3.1 KCCA algorithm

1. Normalized the given input data (X, Y) of dimension n*p and n*q respectively.

2. Computing φ by converting the given input data into higher dimensions by applying a Hilbert transformer.

3. Computing Kernel matrices (Kx and Ky) by applying rbf function to the input data.

4. Finding eigenvectors by solving Lagrange's equation using SVD.

5. Final resultant projections (wx, wy) are obtained by multiplying computed eigenvectors (a, b) and φ.

6. Final transform matrix is obtained by multiplying resultant projection matrices (wx, wy) with given input (X, Y).

Chapter 4

Documentation API

4.1 Package organization

from KCCA import KCCA

kcca_object = KCCA()

Parameters:

X : ndarry, [n, p]

Matrix of one feature space.

Y : ndarry, [n, q]

Matrix of second feature space.

Methods :

1. *fit(X, Y)*
   Fit model to data.
   Parameters:
   X : ndarray, (n*p)
   where p is a feature and n is a number of samples.
   Y : ndarray, n*q
   where q is a feature and n is a number of samples.
   Output : projection matrices wx, wy are ndarrays of dimension n*n each.

2. *Kcca_object.fit_transform(X, Y)*
   Gives the transformed kcca matrix
   Parameters:
   X : ndarray, (n*p)
   where p is a feature and n is a number of samples.
   Y : ndarray, n*q
   where q is a feature and n is a number of samples.
   Output : KCCA transformed matrices Xnew, Ynew.

Chapter 5

Example

```
#importing package
from KCCA import KCCA
import numpy as np

#inputData()
sigma = 1
mu = 0
x = np.random.normal(mu,sigma, 8)
X = x.reshape(4,2)
y = np.random.normal(mu,sigma, 12)
Y = y.reshape(4,3)

#make Object by calling KCCA
kcca=KCCA()
```

```
#fitting data
wx,wy=kcca.fit(X,Y)
print("Projections: ")
print("Wx :: ",wx)
print("Wy :: ",wy)
```

```
Projections:
Wx ::  [[ 0.51120985 -0.13264769  0.09310304  0.02670864]
 [ 0.24827265  0.77708283  0.01482608  0.00856369]
 [-0.5077931  -1.71087247  0.18165262  0.18181526]
 [-0.25168941  1.06643733 -0.28958174 -0.21708759]]
Wy ::  [[-1.61009058e+00 -3.07593479e+00  4.57181019e-02 -7.25117175e-04]
 [-9.03899148e-01  7.57346880e+00 -2.04991655e-02  5.59934452e-04]
 [ 1.43570799e+00 -1.89886742e+00 -1.81070486e-02  2.44607364e-04]
 [ 1.07828173e+00 -2.59866659e+00 -7.11188779e-03 -7.94246411e-05]]
```

```
#Transformed matrix
Xnew,Ynew= kcca.fit_transform(X,Y)
print("Output transformed Matrix: ")
print("Xnew :: ",Xnew)
print("Ynew ::",Ynew)
```

```
Output transformed Matrix:
Xnew ::  [[-0.02818607 -1.39089002]
 [-1.61762106 -4.48637073]
 [ 0.28580003  0.31684119]
 [ 0.2293378   0.33089151]]
Ynew :: [[ 4.00737276e-01 -7.62394697e-01  2.65304475e-01]
 [-1.53164800e+01 -1.28629348e+01 -9.50325616e+00]
 [ 6.84003271e-02  7.84534460e-02  4.21640915e-02]
 [-1.48118418e-03 -1.56865924e-03 -9.23200984e-04]]
```

Chapter 6

Learning Outcomes

- We got a deeper knowledge of how CCA works by combining various views into one single space.
- We implemented Hilbert transform and the theory behind Hilbert space.
- Studied a non-linear kernel like Gaussian RBF and implemented it.
- Learned about singular value decomposition (svd) for computing Lagrange's equations.

References

a. *A kernel method for canonical correlation analysis - Shotaro Akaho*
https://arxiv.org/pdf/cs/0609071.pdf

b. *Canonical Correlation Analysis (CCA) Based Multi-View Learning: An Overview - Chenfeng Guo and Dongrui Wu*
https://arxiv.org/abs/1907.01693