Randomized Canonical Correlation Analysis (RCCA)

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

December 13, 2020

Contents

Chapter 1

Introduction

1.1 Canonical Correlation Analysis (CCA)

Canonical Correlation Analysis (CCA) is a fundamental statistical technique for characterizing the linear relationships between two multidimensional variables. First introduced in 1936 by Hotelling, it has found numerous applications. For the machine learning community, more familiar applications include learning with privileged information, semi-supervised learning, monolingual and multilingual word representation learning, locality sensitive hashing and clustering. Because these applications involve unlabeled or partially labeled data, the amount of data available for analysis can be vast, motivating the need for scalable approaches.

1.1 Randomized Canonical Correlation Analysis (RCCA)

Kernel Canonical Correlation Analysis technique has very high computational complexity. LopezPaz et al. handled the nonlinear characteristics of data by performing random nonlinear projection, which greatly reduced the computational difficulty, with little scarification of performance.

RCCA is a low-rank approximation of KCCA when the latter is equipped with a pair of shift - invariant kernels. RCCA can be understood as linear CCA performed on a pair of randomized nonlinear mappings of the data X ∈ R n×p and Y ∈ R n×q .

$$z_x : \mathbb{R}^{n\times p} \rightarrow \mathbb{R}^{n\times m_x}, \; z_y : \mathbb{R}^{n\times q} \rightarrow \mathbb{R}^{n\times m_y}$$

Schematically,

RCCA(X,Y) := CCA(zx(X), zy(Y)) ≈ KCCA(X,Y)

Chapter 2

Mathematical Formulation

2.1 Formulation

1. *Cholesky decomposition* :

The Cholesky decomposition of a Hermitian positive-definite matrix A, is a decomposition of the form

$$A = L.L^*$$

where L is a lower triangular matrix with real and positive diagonal entries, and L* denotes the conjugate transpose of L. Every Hermitian positive-definite matrix (and thus also every real-valued symmetric positive-definite matrix) has a unique Cholesky decomposition.The converse holds trivially: if A can be written as LL* for some invertible L, lower triangular or otherwise, then A is Hermitian and positive definite.

When A is a real matrix (hence symmetric positive-definite), the factorization may be written

$$A = LL^T,$$

where L is a real lower triangular matrix with positive diagonal entries.

2. *Singular value decomposition* :

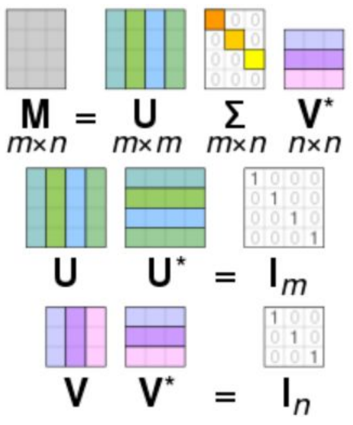
In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any mxn matrix via an extension of the polar decomposition.

M = U.S.(V_T)

Where, a rotation or reflection (V_T),

followed by a coordinate-by-coordinate scaling (S),

followed by another rotation or reflection(U).

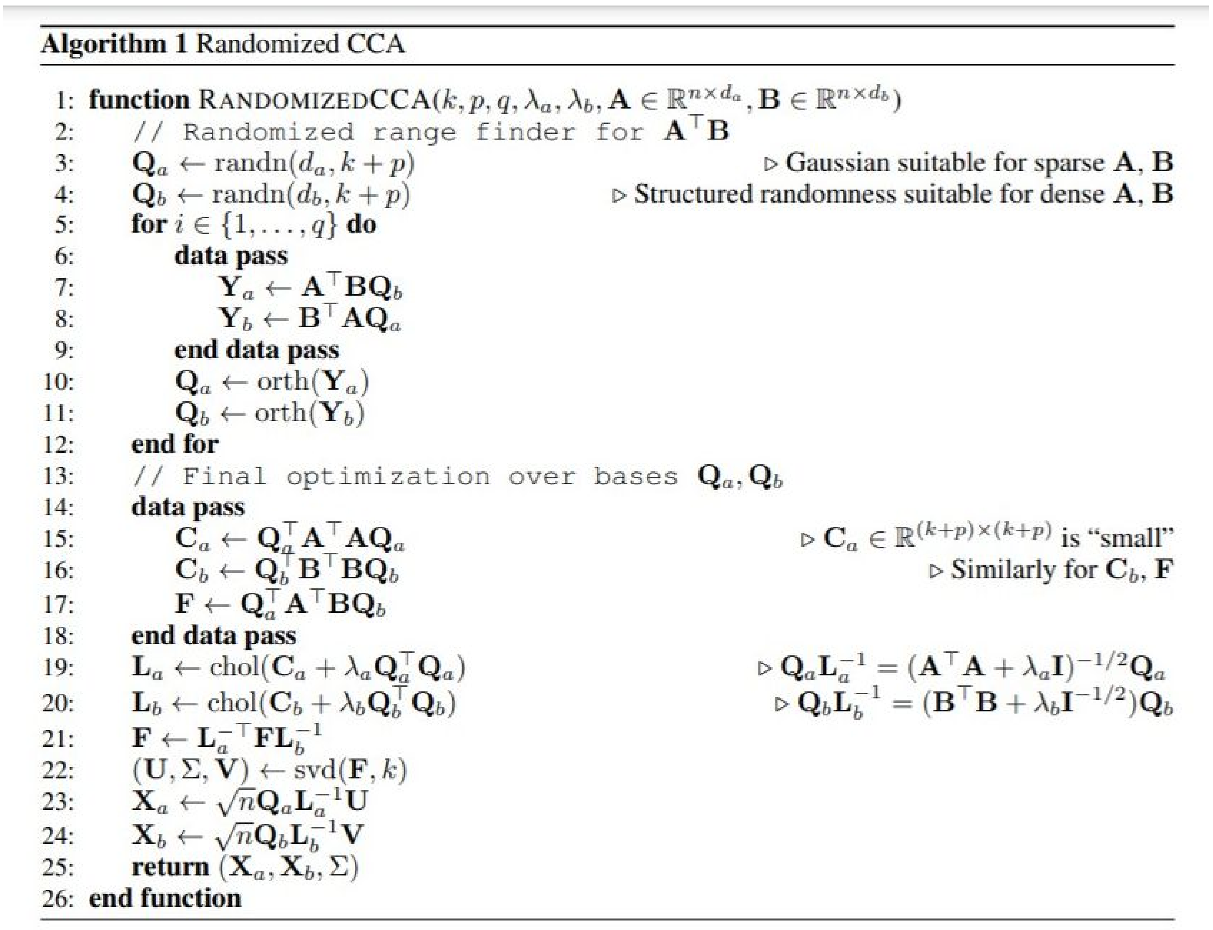


Chapter 3

Algorithm

In this section, we give an overview of the RCCA algorithms where we formulate the optimization problem.

3.1 RCCA algorithm



Algorithm 1 Randomized CCA

```
function RandomizedCCA($k, p, q, \lambda_a, \lambda_b, \mathbf{A} \in \mathbb{R}^{n \times d_a}, \mathbf{B} \in \mathbb{R}^{n \times d_b}$)
    // Randomized range finder for $\mathbf{A}^\top \mathbf{B}$
    $\mathbf{Q}_a \leftarrow \mathrm{randn}(d_a, k+p)$                ▷ Gaussian suitable for sparse $\mathbf{A}, \mathbf{B}$
    $\mathbf{Q}_b \leftarrow \mathrm{randn}(d_b, k+p)$                ▷ Structured randomness suitable for dense $\mathbf{A}, \mathbf{B}$
    for $i \in \{1, \ldots, q\}$ do
        data pass
            $\mathbf{Y}_a \leftarrow \mathbf{A}^\top \mathbf{B} \mathbf{Q}_b$
            $\mathbf{Y}_b \leftarrow \mathbf{B}^\top \mathbf{A} \mathbf{Q}_a$
        end data pass
        $\mathbf{Q}_a \leftarrow \mathrm{orth}(\mathbf{Y}_a)$
        $\mathbf{Q}_b \leftarrow \mathrm{orth}(\mathbf{Y}_b)$
    end for
    // Final optimization over bases $\mathbf{Q}_a, \mathbf{Q}_b$
    data pass
        $\mathbf{C}_a \leftarrow \mathbf{Q}_a^\top \mathbf{A}^\top \mathbf{A} \mathbf{Q}_a$                ▷ $\mathbf{C}_a \in \mathbb{R}^{(k+p) \times (k+p)}$ is “small”
        $\mathbf{C}_b \leftarrow \mathbf{Q}_b^\top \mathbf{B}^\top \mathbf{B} \mathbf{Q}_b$                ▷ Similarly for $\mathbf{C}_b$, $\mathbf{F}$
        $\mathbf{F} \leftarrow \mathbf{Q}_a^\top \mathbf{A}^\top \mathbf{B} \mathbf{Q}_b$
    end data pass
    $\mathbf{L}_a \leftarrow \mathrm{chol}(\mathbf{C}_a + \lambda_a \mathbf{Q}_a^\top \mathbf{Q}_a)$                ▷ $\mathbf{Q}_a \mathbf{L}_a^{-1} = (\mathbf{A}^\top \mathbf{A} + \lambda_a \mathbf{I})^{-1/2} \mathbf{Q}_a$
    $\mathbf{L}_b \leftarrow \mathrm{chol}(\mathbf{C}_b + \lambda_b \mathbf{Q}_b^\top \mathbf{Q}_b)$                ▷ $\mathbf{Q}_b \mathbf{L}_b^{-1} = (\mathbf{B}^\top \mathbf{B} + \lambda_b \mathbf{I}^{-1/2}) \mathbf{Q}_b$
    $\mathbf{F} \leftarrow \mathbf{L}_a^{-\top} \mathbf{F} \mathbf{L}_b^{-1}$
    $(\mathbf{U}, \Sigma, \mathbf{V}) \leftarrow \mathrm{svd}(\mathbf{F}, k)$
    $\mathbf{X}_a \leftarrow \sqrt{n} \mathbf{Q}_a \mathbf{L}_a^{-1} \mathbf{U}$
    $\mathbf{X}_b \leftarrow \sqrt{n} \mathbf{Q}_b \mathbf{L}_b^{-1} \mathbf{V}$
    return $(\mathbf{X}_a, \mathbf{X}_b, \Sigma)$
end function
```

Chapter 4

Documentation API

4.1 Package organization

from RandomizedCCA import RandomizedCCA

rcca_object = RandomizedCCA ()

Parameters:

X : ndarry, [n, p]

Matrix of one feature space.

Y : ndarry, [n, q]

Matrix of second feature space.

n_passes : int

Number of iterations

n_features : int

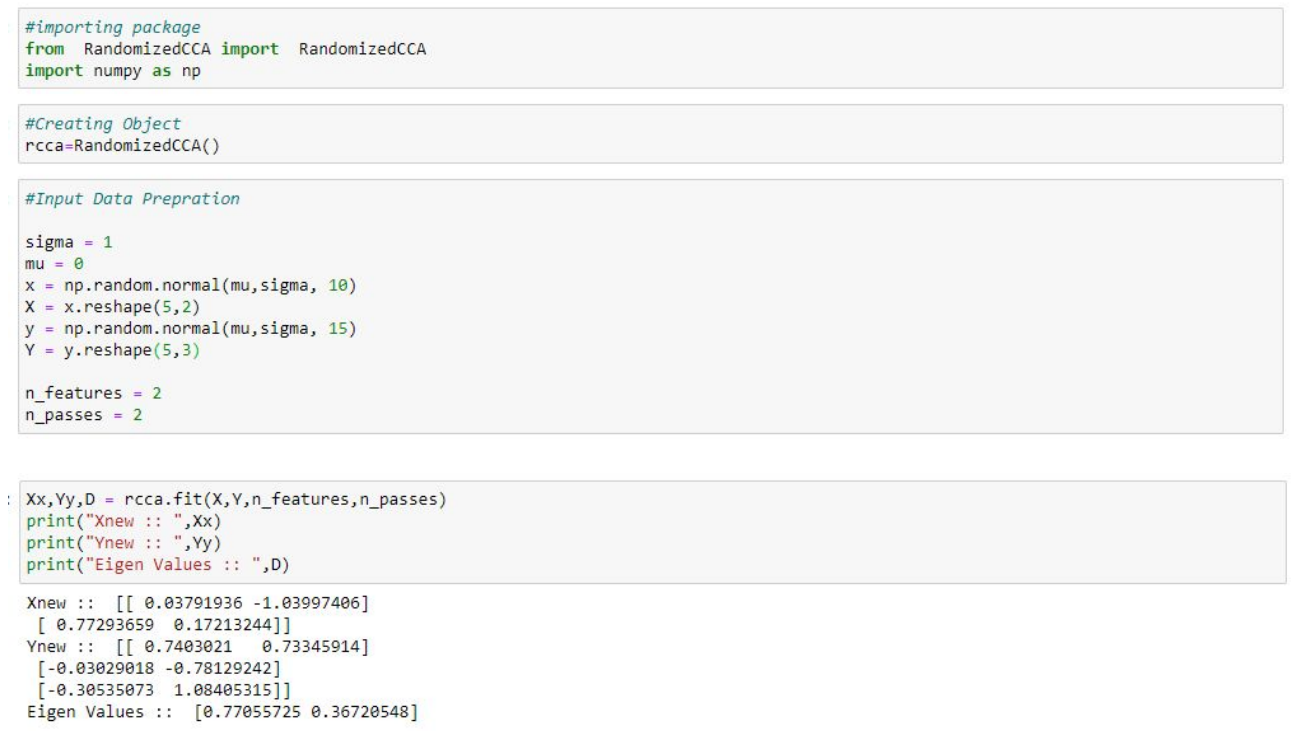
Number of features to be extracted must be less than or equal to max(p, q)

Methods :

1. ***fit(X, Y, n_features, n_passes)***
   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.
   n_passes : int
   Number of iterations
   n_features : int
   Number of features to be extracted must be less than or equal to max(p, q)
   Output : projection matrices wx, wy are ndarrays of dimension n*n each.

Chapter 5

Example



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

```
#Creating Object
rcca=RandomizedCCA()
```

```
#Input Data Prepration

sigma = 1
mu = 0
x = np.random.normal(mu,sigma, 10)
X = x.reshape(5,2)
y = np.random.normal(mu,sigma, 15)
Y = y.reshape(5,3)

n_features = 2
n_passes = 2
```

```
Xx,Yy,D = rcca.fit(X,Y,n_features,n_passes)
print("Xnew :: ",Xx)
print("Ynew :: ",Yy)
print("Eigen Values :: ",D)
```

```
Xnew ::  [[ 0.03791936 -1.03997406]
 [ 0.77293659  0.17213244]]
Ynew ::  [[ 0.7403021   0.73345914]
 [-0.03029018 -0.78129242]
 [-0.30535073  1.08405315]]
Eigen Values ::  [0.77055725 0.36720548]
```

Chapter 6

Learning Outcomes

- Studied thoroughly CCA and multi-view analysis.
- Learned Kernel CCA implementation and how the randomized CCA is evolved through it.
- We used and learned about the use of Cholesky decomposition by scipy linear algebra library.
- We learned how singular value decomposition is used to find eigenvalues and vectors by solving Lagrange’s equations.

References:

a. *A Randomized Algorithm for CCA - Paul Mineiro and Nikos Karampatziakis*
https://stanford.edu/~rezab/nips2014workshop/submits/randcca.pdf

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