# Data Science Project - 2
# Deep Generalized Canonical Correlation Analysis (DG-CCA)

Madiha Taj Qureshi (0801CS171039)
CSE IV year
(Sept - Jan 2020)

# Contents

# 1. Introduction

### 1.1.1. Multiview Learning

Multi-view learning refers to the technique of performing machine learning on data that is represented by multiple distinct feature sets called views. Data from real world applications is multiple viewed because it is often collected from different measuring methods and it describes information more minutely than single viewed data. Information from different views can be effectively utilized to gain performance improvements.

### 1.1.2. CCA

CCA is a co-regularization style algorithm, which is one of the types of multiview learning methods. It is a subspace learning technique to find linear combinations(or projections) for 2 views (or vectors) such that the correlation between the two projections is maximized in the common subspace i.e for two input views CCA finds directions to maximize the correlation between them.

Two inherent drawbacks of traditional CCA are -

It can only be applied to datasets with *only 2 views.*

It can only calculate a *linear* correlation between the two views.

Since many real-world datasets can have more than one view and the relationship between the views can be nonlinear, traditional CCA has many extensions, two of which are discussed in the following sections.

### 1.1.3. DCCA

Deep Canonical Correlation Analysis (DCCA) is a non-linear two-view representation learning technique based on Deep Neural Networks (DNN). It is a parametric alternative of the Kernelized Canonical Correlation Analysis (KCCA). DCCA simultaneously computes representations of the two views by passing each of them through a DNN, such that the representations are maximally correlated. It does so by performing CCA on the outputs of the DNN. The objective of DCCA is to find parameters (weights and biases) for the 2 DNNs such that the correlation between their outputs is maximum.

### 1.1.4. GCCA

Generalized Canonical Correlation Analysis (GCCA) is an extension of CCA that can be applied to datasets with more than two views. It estimates pairwise correlations of multiple views. Its objective is to find a shared representation of different views.

### 1.1.5. DGCCA

Deep Generalized Canonical Correlation Analysis (DGCCA) is an extension of CCA that combines the advantages of DCCA & GCCA. It learns a shared representation from data with >2 views and simultaneously learns a DNN for each view in order to maximize the correlation between the learned representations.

At train time - DGCCA passes each view to a different DNN and then back propagates the gradient of the GCCA objective to update the DNN parameters. This trains the DNNs to reduce the GCCA reconstruction error among their outputs.
At test time - New data is projected by feeding them through the learned DNN for each view.

DGCCA optimization problem(formulated below) can be solved using stochastic gradient descent (SGD) with mini-batches.
**Constraint :** The nonlinear mappings from views to shared space must be differentiable.
DGCCA does not reduce to DCCA for 2 views, since the objective and constraints of the GCCA are different from that of the CCA problem.

## 2. Mathematical Formulation

Following notations are used in the formulations
**D** : mean-zero dataset
$D = \{ (x_1,y_1) , (x_2,y_2) , . . . . , (x_n,y_n) \}$
**n** : no of instances in the dataset
First view of the dataset : $\mathbf{X} = [x_1, x_2, . . . , x_n ]_{dx \times n}$
$\mathbf{d_x}$ : No of features in X
Second view of the dataset : $\mathbf{Y} = [y_1, y_2, . . . , y_n ]_{dy \times n}$
$\mathbf{d_y}$ : No of features in Y
$\|\cdot\|_2$ : 2-norm of a vector
$\|\cdot\|_F$ : The Frobenius-norm of a matrix

### 2.1.1. Formulation of CCA

CCA aims to find linear projections $W_x$ and $W_y$ for X and Y respectively such that $corr(W_x^T X, W_y^T Y)$ is maximized, where
$W_x = [w_{x,1}, w_{x,2}, . . . , w_{x,dx}]$ and $W_y = [w_{y,1}, w_{y,2}, . . . , w_{y,dy}]$

Objective function -

$$\boxed{\begin{array}{c}\underset{w_x,w_y}{\text{maximize}} \quad w_x^T XY^T w_y \\ \text{s.t.} w_x^T XX^T w_x = w_y^T YY^T w_y = 1\end{array}}$$

### 2.1.2. Formulation of DCCA

$c_1, c_2$ : no of neurons in each intermediate layer in DNN of X & Y respectively
$o_1, o_2$ : no of neurons in the output layer in DNN of X & Y respectively
$x_1, x_2$ : an instance of first and second view respectively
d : no of layers in first and second DNN
$W_k^1$ , $W_k^2$ : matrix of weights for kth layer of first and second DNN of dimension ($c_1$ ✖ n) and ($c_2$ ✖ n) respectively
$b_k^1$, $b_k^2$ : vector of biases for kth layer of first DNN and second DNN of dimension $c_1$ and $c_2$ respectively
s : non-linear activation function

Let $h_1$ be output of the first layer of the first DNN, then
$h_1 = s(W_1^1 \ x_1 + b_1^1)$ is used as input for second layer
$h_2 = s(W_2^1 \ h_1 + b_2^1)$
Similarly for the second DNN

$f_1, f_2$ : output of first and second DNN respectively
where
$f_1(x_1) = s(W_d^1 \ h_{d-1} + b_d^1)$ and $f_2(x_2) = s(W_d^2 \ h_{d-1} + b_d^2)$

$\theta_1, \theta_2$ : vectors of all parameters $W^1_k$ and $b^1_k$ of the first and all parameters $W^2_k$ and $b^2_k$ of the second DNN respectively

Objective function -

$$\boxed{(\theta^*_1, \theta^*_2) = \underset{(\theta_1,\theta_2)}{\text{argmax}} \quad \text{corr}(f_1(X_1; \theta_1), f_2(X_2; \theta_2))}$$

### 2.1.3. Formulation of GCCA

J : no of views
$X_j$ : input data matrix of view j
G : set of multivariate latent variables
$G = [g_1, g_2, \ldots, g_n]^T$

Objective function -

$$\underset{U_j,G}{\text{minimize}} \quad {}^{J}{}_{j=1}\Sigma \, ||G - U^T{}_j \, X_j||^{2}{}_F \qquad \text{s.t. } GG^T = I_r \text{ (i.e. G is orthogonal)}$$

### 2.1.4. Formulation of DGCCA

$K_j$ : No. of layers in the DNN of jth view
$c_j$ : no of neurons in each layer on DNN of jth view
$o_j$ : no of neurons in output layer on DNN of jth view
$h^j{}_k$ : e output of the k th layer for the j th view
$h^j{}_k = s(W^j{}_k \, h^j{}_{k-1})$
$W_k^j$ : matrix of weights for kth layer of jth view
$f_j(X_j)$ : output of final layer
$U_j$ : linear transformation of the output of the jth networ

Objective function -

$$\underset{U_j,G}{\text{minimize}} \quad {}^{J}{}_{j=1}\Sigma \, ||G - U^T{}_j \, f_j(X_j)||^{2}{}_F \qquad \text{s.t. } GG^T = I_r$$

## 3. Algorithm

### 3.1.1. DG-CCA Algorithm

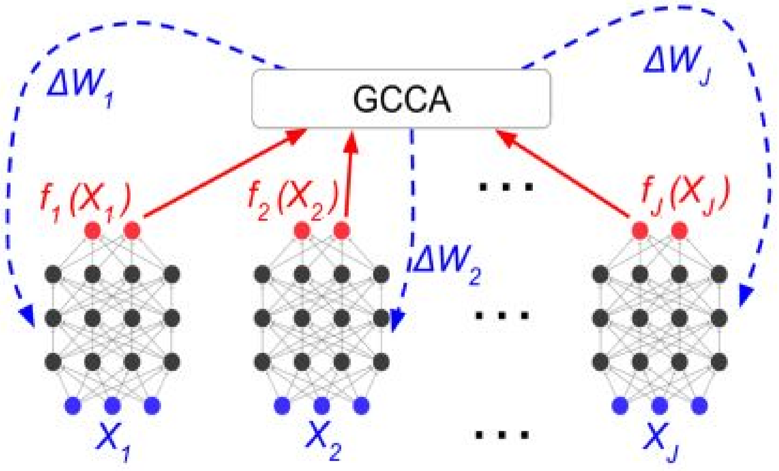
**Algorithm : dgcca for 3 views**

**Inputs :**
multiview data X1 , X2 , X3
Learning Rate M
Epoch number T

```
START

// At Train time

for iteration t = 1, 2, . . . , T do
    //forward feeding inputs
    for each view j = 1, 2, 3 do
```

```
        Oj ← forward pass of Xj with weights Wj
         mean-center Oj
    end for

    //applying gcca on output of network
    U1, U2, U3 ,G ← gcca(O1 , O2, O3 )

   // calculating and backpropogating gcca_loss to update weights
   for each view j = 1, 2, 3 do
     gcca_loss ← $U_jU^T_j O_j − U_jG$
     ∇Wj ← backprop(gcca_loss )
     Wj ← Wj − M∇Wj
   end for
 end for


// At Test time

    for each view j = 1, 2, 3 do
        Oj ← forward pass of Xj with weights Wj
         mean-center Oj
    end for

    U1, U2, U3 ,G ← gcca(O1 , O2, O3 )

   for each view j = 1, 2, 3 do
      Oj ← $U^T_j$ Oj
   end for

END
```

**Output :** learnt representations with maximum correlation O1, O2 , O3



# 4. Documentation of API

### 4.1.1. Package organization

**Package name** : DeepGeneralizedCCA
**Path in github repository** : DataScience/Groups/Group_ID_3/DeepGeneralizedCCA/
(https://github.com/shekhar-sharma/DataScience/tree/main/Groups/Group_ID_3/DeepGeneralizedCCA)

Files in Package -

1. **dgcca.py** : contains source code to implement DGCCA
2. **dgcca_exampe.ipynb** : notebook containing exapmle of using dgcca_pckg
3. **DGCCA_Documnetation.html** : contains complete documentation of the API
4. **dgcca.html** : Notebook view of the source code file (dgcca.py)
5. **requirement.txt** : contains dependencies
6. **READ_ME.md** : Markdown file

### 4.1.2. Methods

Class-wise methods in the *dgcca.py* file -

1. **Class DNN : Creates a new Deep Neural Network**
*forward(self, l)* : forward propagates input tensor into the DNN and returns the output tensor (overriden)

2. **Class : DGCCA_architecture - Defines the architecture for three DNNs**
*forward(self, x1, x2, x3)* : forward propagates x1 into the first DNN, x2 into the second DNN and x3 into the third DNN and returns the outputs. (overriden)

3. **Class DGCCA : Implements the DGCCA Algorithm**
*fit_transform(self, train_x1, train_x2, train_x3, test_x1, test_x2, test_x3)* : Learn and apply the dimension reduction on the train data batch-wise. Trains the networks in mini-batches. Back propagates the ggca loss to tune network acc to data. Each view needs to have the same number of features as its corresponding view in the training data.

*predict(self, x1, x2, x3)* - returns gcca loss as ndarray and output as list for given inputs x1, x2, x3 for view first, second, third respectively.

*test(self, x1, x2, x3)* - returns gcca loss mean and output as list for given inputs x1, x2, x3 for view first, second, third respectively.

## 5. Example

An example of using the dgcca package to the 'Fetal Health Classification' dataset is provided in the file **dgcca_exampe.ipynb** (Dataset Source : https://www.kaggle.com/andrewmvd/fetal-health-classification).

**Dataset Description** - Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing doctors to take action to prevent child and maternal mortality. This dataset includes 2126 records of features extracted from Cardiotocogram exams, which are classified into 3 classes: Normal, Suspect, Pathological.

It contains three sections -

1. Data Understanding
2. Data Preprocessing
3. Applying DGCCA

## 6. Learning Outcomes

- Learned the basics of cannonical correlation analysis, its variants and DCCA, GCCA, DGCCA and their application in multiview learning.
- Learned working in pytorch (python library) and building APIs.