WPI

Kernel Coherence Encoders

Data Science Master's Thesis Report

by Fangzheng (Andy) Sun

Adviser: Professor Randy C. Paffenroth

Reader: Professor Jian Zou

Overview

Our research

- Modifies Autoencoders (special neural networks mapping data to back to itself)
- Explores Canonical Correlation Analysis(CCA) and Kernel CCA(KCCA)
- Introduces element-wise coherence

Three models

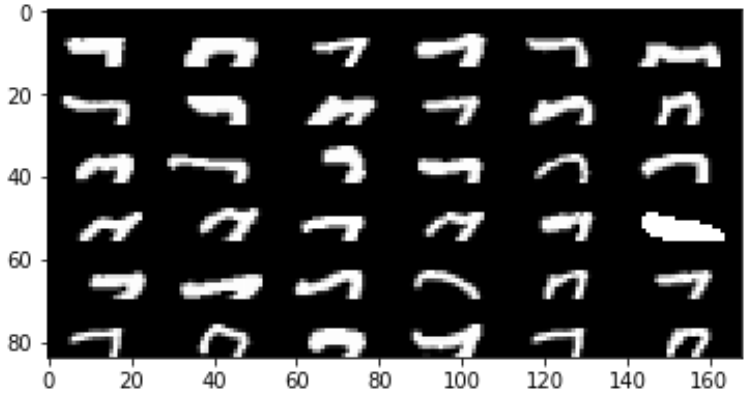
- Non-Coherence Encoder (NCE)
- Coherence Encoder (CE)
- Kernel Coherence Encoder (KCE)

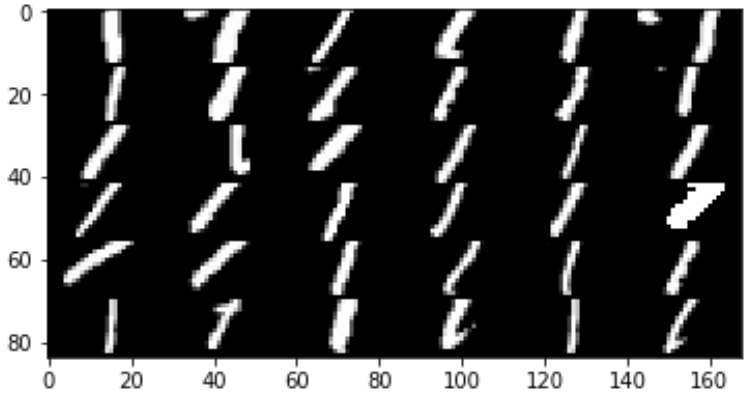
Motivation

- reconstruct one data set from another dependent data set

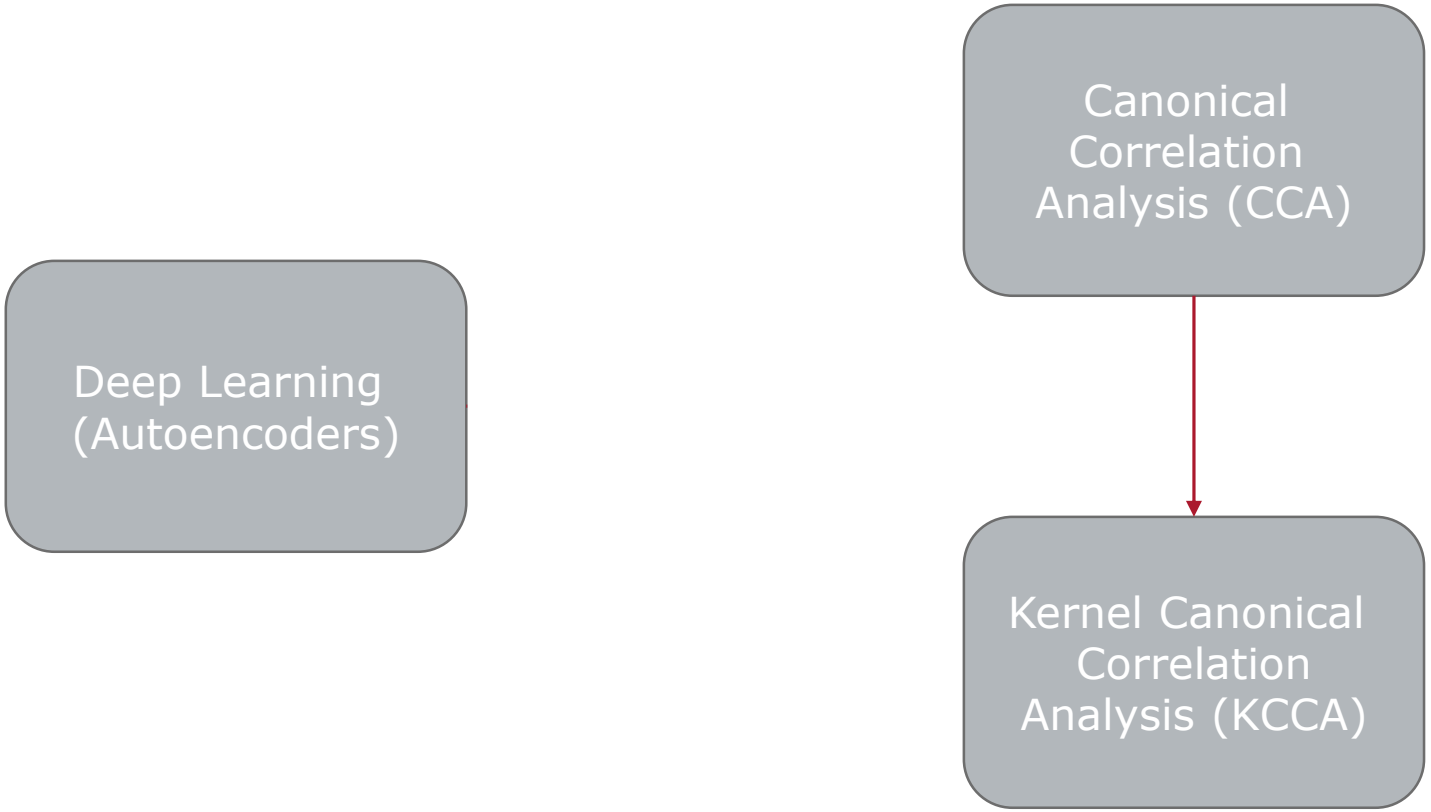
Experiment

- MNIST Digit images

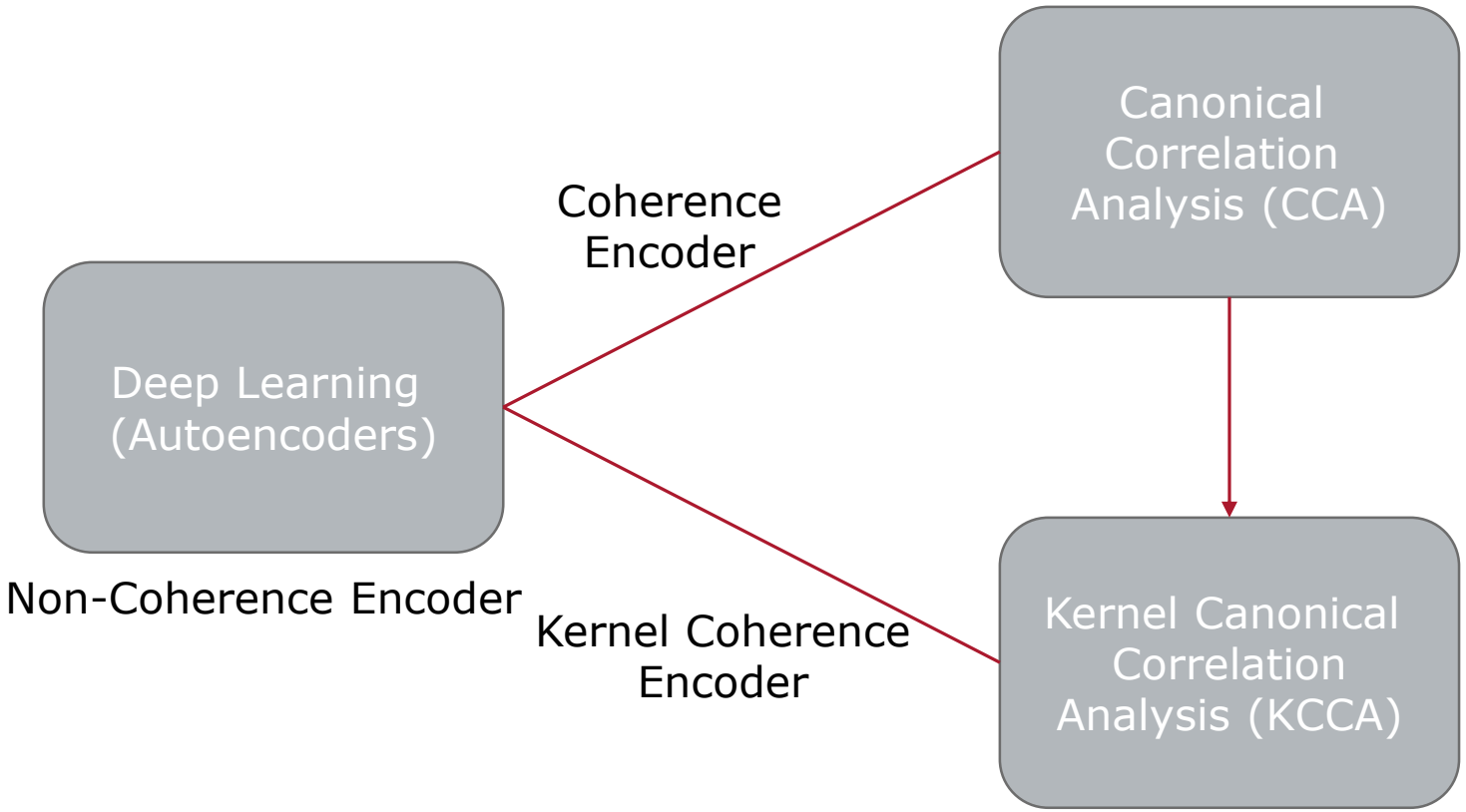




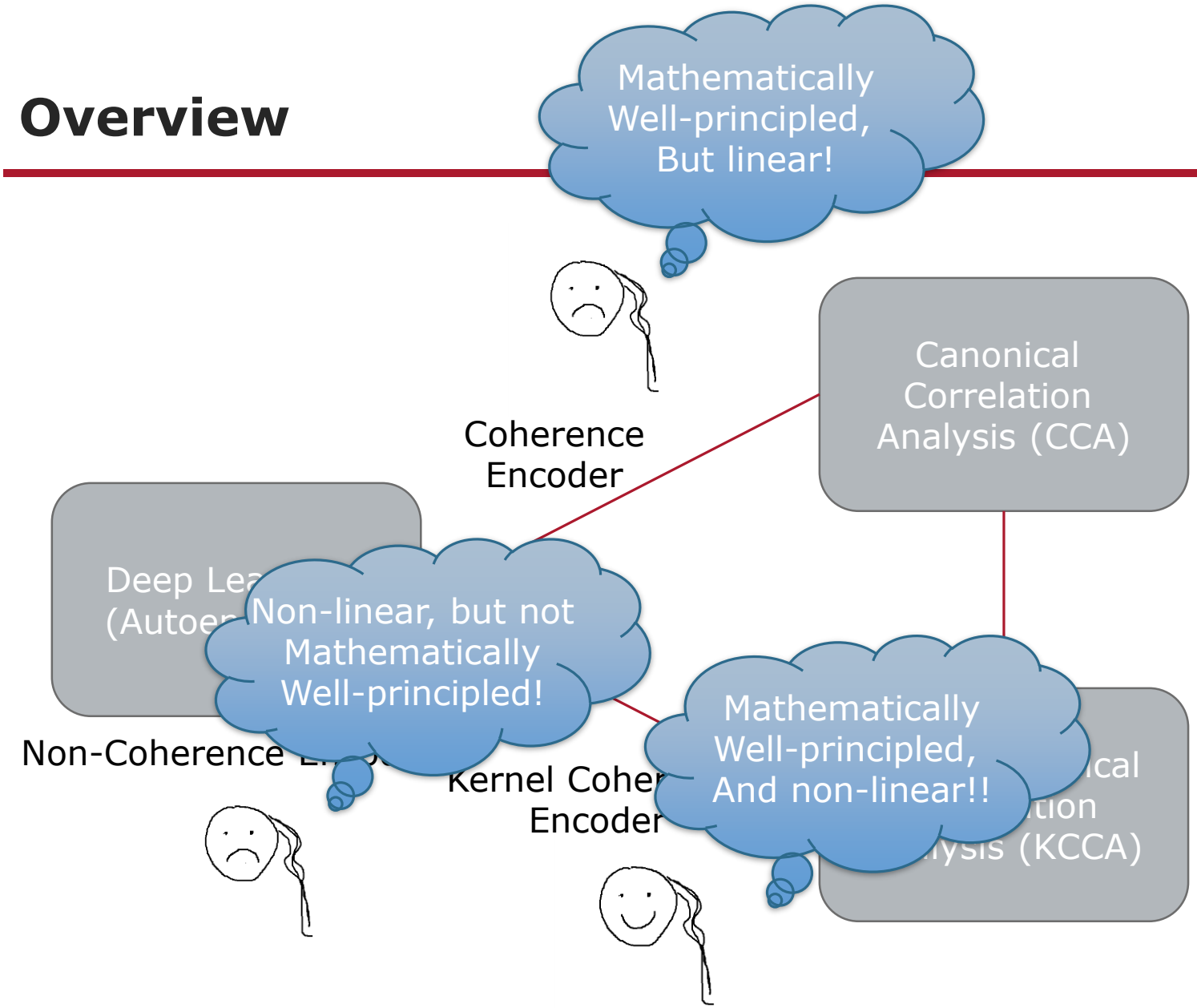
Overview


Deep Learning
(Autoencoders)
Canonical
Correlation
Analysis (CCA)
Kernel Canonical
Correlation
Analysis (KCCA)

Overview


Canonical Correlation Analysis (CCA)
Coherence Encoder
Deep Learning (Autoencoders)
Non-Coherence Encoder
Kernel Coherence Encoder
Kernel Canonical Correlation Analysis (KCCA)

Overview


Mathematically
Well-principled,
But linear!
Canonical
Correlation
Analysis (CCA)
Coherence
Encoder
Deep Lea
(Autoen
Non-linear, but not
Mathematically
Well-principled!
Non-Coherence
Kernel Coher
Encoder
Mathematically
Well-principled,
And non-linear!!
ical
tion
ysis (KCCA)

Contribution

- People try to connect Autoencoders, kernel methods and other mathematical approaches for data reconstruction.
- We come up with NCE, CE and KCE.
- NCE is not mathematically well principled and CE is based on linear CCA.
- KCE is the only one to be non-linear and mathematically well-principled model.
- Moreover, experimental results demonstrate that KCE has the best performance among the three models.

Background

- Principal Component Analysis (PCA)
- Canonical Correlation Analysis (CCA)
- Kernel methods
- Kernel Canonical Correlation Analysis (KCCA)
- Artificial Neural Networks
- Autoencoders

Principal Component Analysis (PCA)

- PCA is a popular unsupervised learning approach for discovering a low-dimensional set of features from a large set of variables.
- For high-dimensional data, principal components allow us to summarize this set with a smaller number of representative variables that collectively explain most of the variability in the original set.
- For $X = (x_1, x_2, \ldots, x_p)$,

$$
\begin{aligned}
z_1 &= \alpha_{11}x_1 + \alpha_{21}x_2 + \cdots + \alpha_{p1}x_p \\
z_2 &= \alpha_{12}x_1 + \alpha_{22}x_2 + \cdots + \alpha_{p2}x_p \\
&\vdots \\
z_k &= \alpha_{1k}x_1 + \alpha_{2k}x_2 + \cdots + \alpha_{pk}x_p
\end{aligned}
$$

- subject to the constraint that $\sum_{j=1}^{p} \alpha_{jk}^2 = 1$ for each z_k.

PCA – Optimization

- Each principal component vector z_k solves the following optimization problem:

$$\min_{D'} ||D - D'||_F \ subject\ to\ rank(D') \leq r$$

- Where D is data matrix and r is a desired rank.
- This can be solved via an singular value decomposition of the data matrix, a standard technique in linear algebra.

Canonical Correlation Analysis (CCA)

- Infers information from cross-covariance matrices
- Given two vectors of random variables of random variables with correlations among them:

$$X = (x_1, x_2, \dots, x_m), \qquad Y = (y_1, y_2, \dots, y_n)$$

- CCA finds linear combinations of x_i and y_j which have maximum correlation with each other. (canonical variables of X and Y)

Canonical Correlation Analysis

- Infers information from cross-covariance matrices
- Given two vectors of random variables of random variables with correlations among them:

$$X = (x_1, x_2, \dots, x_m), \qquad Y = (y_1, y_2, \dots, y_n)$$

- CCA finds linear combinations of x_i and y_j which have maximum correlation with each other. (canonical variables of X and Y)
- We are inspired by, and closely follow the notation and derivations, in *Empirical canonical correlation analysis in subspaces*

Picture provided by professor Randy Paffenroth

CCA – Linear Algebra (Canonical Variables)

- Define $\mathrm{Z} = [X^T \;\; Y^T]^T$
- Inner product in Hilbert space

$$[R_{xy}]_{ij} = E[x_i y_j]$$

- Covariance matrix

$$R_{zz} = \begin{bmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{bmatrix}$$

- Coherence matrix of x and y

$$C = R_{xx}^{-\frac{1}{2}} R_{xy} R_{yy}^{-\frac{T}{2}}$$

- SVD on C

$$C = R_{xx}^{-\frac{1}{2}} R_{xy} R_{yy}^{-\frac{T}{2}} = F \Sigma G^T$$

- Canonical variables

$$u = \left(F^T R_{xx}^{-\frac{1}{2}} x \right), \qquad v = \left(G^T R_{yy}^{-\frac{1}{2}} y \right)$$

CCA – Optimization (Coherence)

- By Hadamard ratio, dependence

$$L = \det(I - \Sigma\Sigma^T) = \prod_{i=1}^{m}(1 - \sigma_i^2)$$

- Linear Coherence

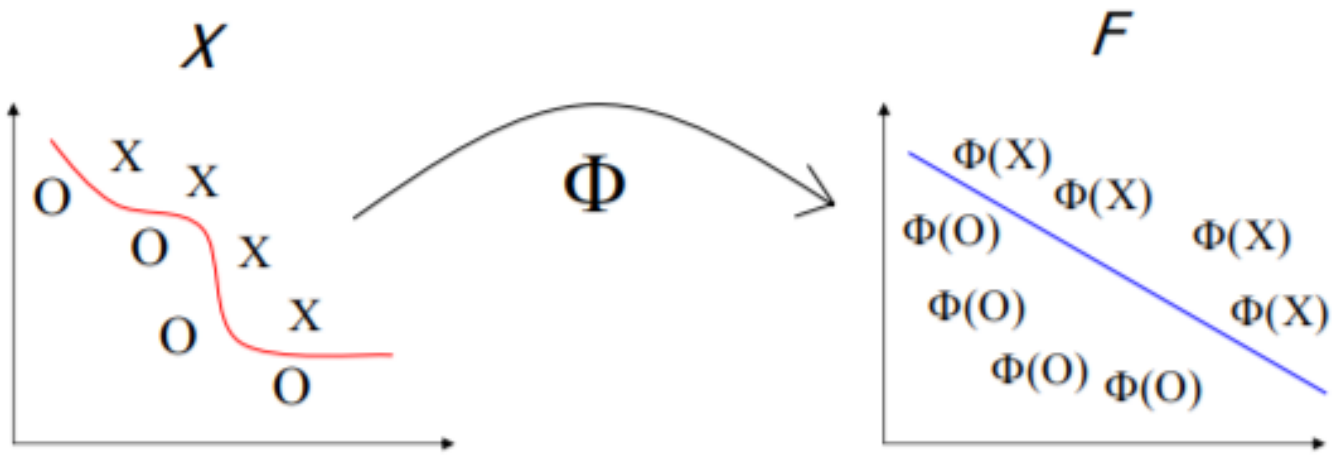
$$H = 1 - L = 1 - \det(I - \Sigma\Sigma^T) = \prod_{i=1}^{m}(1 - \sigma_i^2)$$

$$L = \frac{det(R_{zz})}{det(R_{xx})det(R_{yy})} = \frac{det\left(\begin{bmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{bmatrix}\right)}{det(R_{xx})det(R_{yy})}$$

$$H = 1 - L = 1 - \frac{det\left(\begin{bmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{bmatrix}\right)}{det(R_{xx})det(R_{yy})}$$

Kernel Methods

- Map data into a (possibly high dimensional) vector space (Reproducing Kernel Hilbert Space)* where linear relations exist among the data, then apply a linear algorithm in this space



http://bioinfo.icgeb.res.in/lipocalinpred/algorithm.html

*See back-up slide for completely monotonic function definition

Positive-Definite Kernel

- Makes sure there is a corresponding inner product space for x and y of any type.*
- Mercer's theorem* provides a mathematical foundation for the kernel methods and infers significance of positive definite kernel.
- Definition: A symmetric function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel on X if only if

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

- holds for any $n \in \mathbb{N}, x_1, \ldots, x_n \in \mathcal{X}, c_1, \ldots, c_n \in \mathbb{R}$.

*See back-up slides for more explanation on positive definite kernel and Mercer's Theorem

Schoenberg's Theorem

- Schoenberg (Ann. of Math. 39 (1938), 811-841) observed that if f(t) is a completely monotonic function*, then the radial kernel

$$K(x, y) = f(||x - y||^2)$$

  is positive definite on any Hilbert space.

- Here is one completely monotonic function:

$$f(x) = e^{-\alpha x + b}, \alpha > 0$$

*See back-up slide for completely monotonic function definition

KCCA – Linear Algebra

- Kernel Function:

$$k(x,y) = f\left(\sum_{k=1}^{N} ||x-y||_2^2\right) = e^{-\alpha||x-y||_2^2+b}, \alpha > 0$$

- Covariance matrix:

$$R_{zz} = \begin{bmatrix} \overbrace{\begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix}}^{\text{Rxx}} & \cdots & \overbrace{\begin{bmatrix} k(x_1,y_1) & \cdots & k(x_1,y_n) \\ \vdots & \ddots & \vdots \\ k(x_n,y_1) & \cdots & k(x_n,y_n) \end{bmatrix}}^{\text{Rxy}} \\ \vdots & \ddots & \vdots \\ \underbrace{\begin{bmatrix} k(y_1,x_1) & \cdots & k(y_1,x_n) \\ \vdots & \ddots & \vdots \\ k(y_n,x_1) & \cdots & k(y_n,x_n) \end{bmatrix}}_{\text{Ryx}} & \cdots & \underbrace{\begin{bmatrix} k(y_1,y_1) & \cdots & k(y_1,y_n) \\ \vdots & \ddots & \vdots \\ k(y_n,y_1) & \cdots & k(y_n,y_n) \end{bmatrix}}_{\text{Ryy}} \end{bmatrix}$$

KCCA – Optimization

If one σ_i is large, the result of dependence formula tends to 0 and coherence tends to 1.

- Dependence

$$L = \det(I - \Sigma\Sigma^T) = \prod_{i=1}^{m}(1 - \sigma_i^2)$$

- Coherence

$$H = 1 - L = 1 - \det(I - \Sigma\Sigma^T) = 1 - \prod_{i=1}^{m}(1 - \sigma_i^2)$$

- Problem: sensitive to high-dimensional space.

1. The coherence tends to small if any x is predictable for any y or any y is predictable for any x. (One pair of points ruin the whole calculation)
2. High-dimensional covariance matrix can be close to singular and makes it difficult to compute its determinant, as is required by using a coherence metric.

KCCA – Optimization

- An element-wise modification to the calculation of dependence and coherence

$$L_{i,j} = \frac{det\left(\begin{bmatrix} R_{x_i x_i} & R_{x_i y_j} \\ R_{y_j x_i} & R_{y_j y_j} \end{bmatrix}\right)}{det(R_{x_i x_i}) det(R_{y_j y_j})} = \frac{R_{x_i x_i} R_{y_j y_j} - R_{x_i y_j} R_{y_j x_i}}{R_{x_i x_i} R_{y_j y_j}}$$

$L_{i,j}$ represents the element-wise dependence between x_i and y_j. Now we have

$$L = \frac{\sum_{i,j=1}^{k} (L_{i,j})^2}{n^2}$$

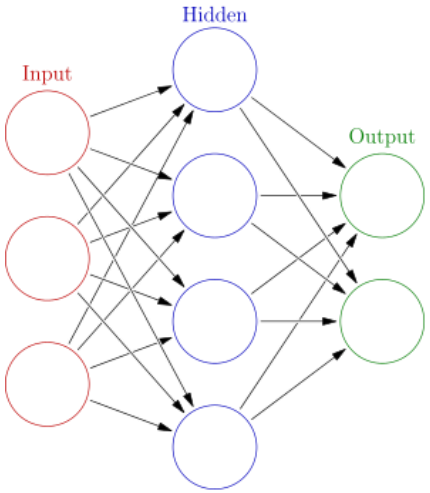
$$C = 1 - L = 1 - \frac{\sum_{i,j=1}^{k} (L_{i,j})^2}{n^2}$$

KCCA – Optimization

1. Coherence is small if and only if all x is predictable from all y and all y is predictable from all x.
2. No problem with determinant calculation.

Artificial Neural Networks

- An ANN is an interconnected group of nodes
- An circular node represents an artificial neuron and an arrow represents a connection between two layers
- $h_{i+1} = \sigma(W_i h_i + b_i)$



https://commons.wikimedia.org/w/index.php?curid=24913461

Autoencoders

- Contains encoding phase and decoding phase
- Maps data back to original feature space

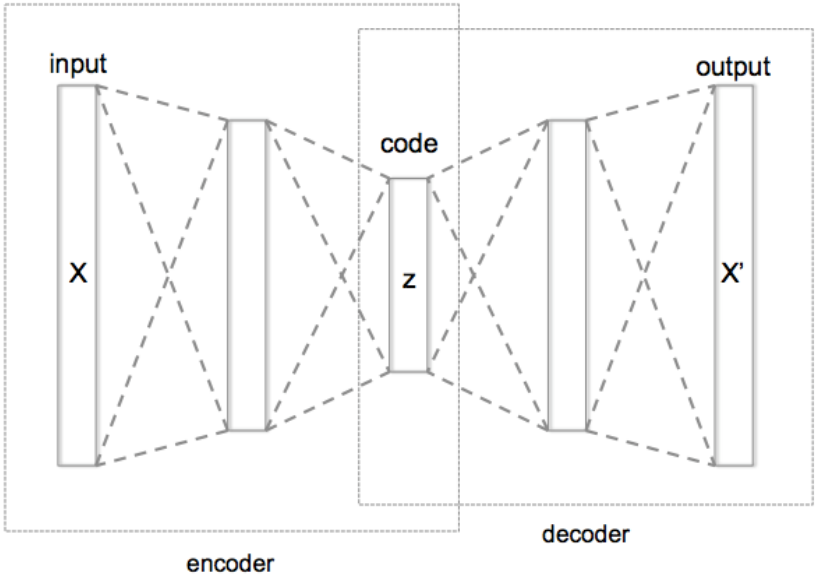
$$\phi : \mathcal{X} \to \mathcal{F}$$

$$\psi : \mathcal{F} \to \mathcal{X}$$

$$\mathcal{AE}_{\phi,\psi} = \underset{\phi,\psi}{\operatorname{argmin}} \|X - \psi(\phi(X))\|_F^2$$

Model Design

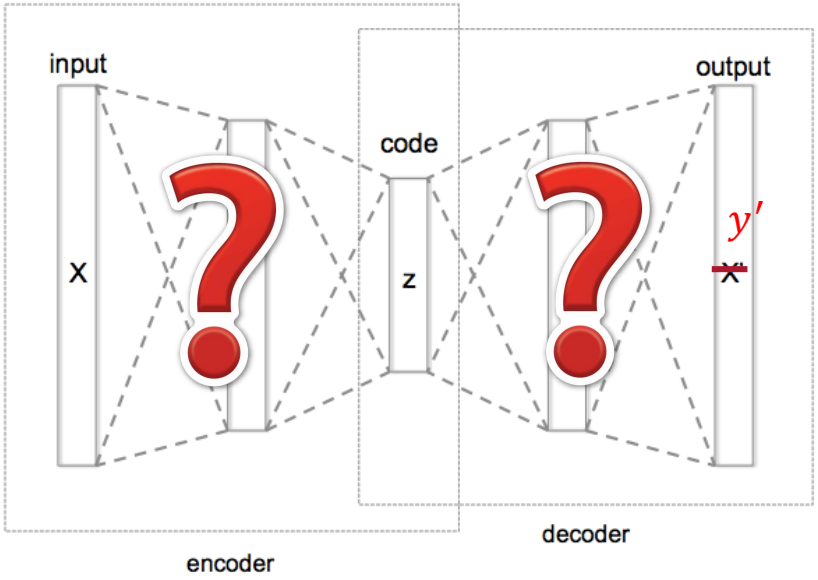
- Non-Coherence Encoder (NCE)
- Coherence Encoder(CE)
- Kernel Coherence Encoder (KCE)



https://commons.wikimedia.org/w/index.php?curid=45555552

Model Design

- Non-Coherence Encoder (NCE)
- Coherence Encoder(CE)
- Kernel Coherence Encoder (KCE)



https://commons.wikimedia.org/w/index.php?curid=45555552

Non-Coherence Encoder (NCE)

- **Encoder Phase**
- PCA

- **Decoder Phase**
- Linear ANNs
- PCA Reconstruction
- Non-linear ANNs

Linear ANNs

- a pair of two-layer linear ANNs
- map z_x to z'_y and z_y to z'_x

$$z'_y = W_{1,2}(W_{1,1}z_x + b_{1,1}) + b_{1,2}$$

$$z'_x = W_{2,2}(W_{2,1}z_y + b_{2,1}) + b_{2,2}$$

$$\gamma_1(W_{1,1}, W_{1,2}, b_{1,1}, b_{1,2}) = \underset{W_{1,1},W_{1,2},b_{1,1},b_{1,2}}{\operatorname{argmin}} \left\| z_y - z'_y \right\|^2$$

$$\gamma_2(W_{2,1}, W_{2,2}, b_{2,1}, b_{2,2}) = \underset{W_{2,1},W_{2,2},b_{2,1},b_{2,2}}{\operatorname{argmin}} \left\| z_x - z'_x \right\|^2$$

PCA Reconstruction

- Principal components mapped back to the original feature space

$$PCA\ reconstruction = Principal\ Components \cdot Eigenvectors^T + Mean$$

$$y'' = z'_y \cdot Eigenvectors_y^T + Mean_y$$

$$x'' = z'_x \cdot Eigenvectors_x^T + Mean_x$$

Non-linear ANNs

- a pair of two-layer non-linear ANNs
- Similar to 2-layer autoencoders
- map y'' to y' and x'' to x'

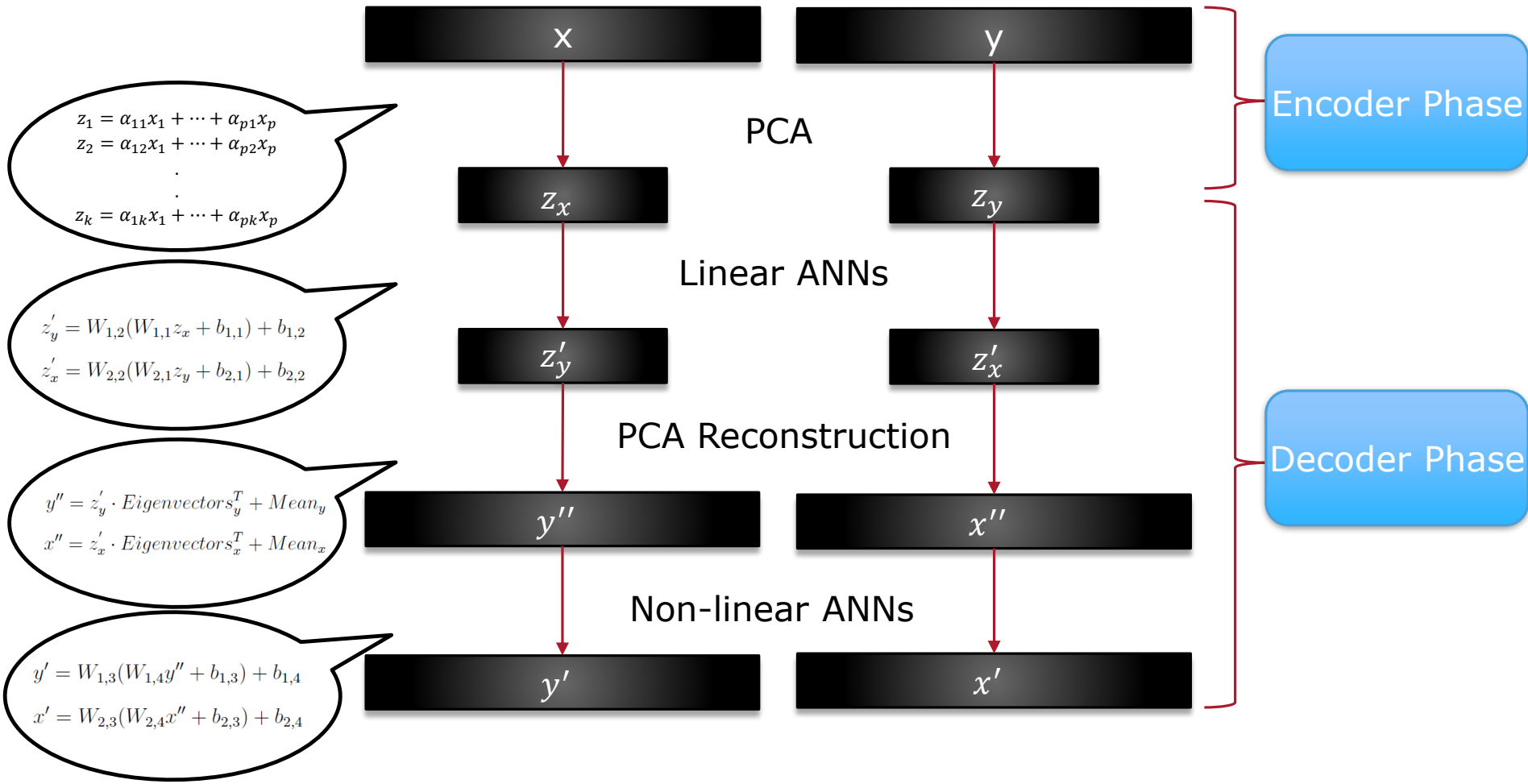
$$y' = W_{1,3}(W_{1,4}y'' + b_{1,3}) + b_{1,4}$$

$$x' = W_{2,3}(W_{2,4}x'' + b_{2,3}) + b_{2,4}$$

$$\mathcal{E}_1(W_{1,3}, W_{1,4}, b_{1,3}, b_{1,4}) = \underset{W_{1,3},W_{1,4},b_{1,3},b_{1,4}}{\operatorname{argmin}} \|y - y'\|^2$$

$$\mathcal{E}_2(W_{2,3}, W_{2,4}, b_{2,3}, b_{2,4}) = \underset{W_{2,3},W_{2,4},b_{2,3},b_{2,4}}{\operatorname{argmin}} \|x - x'\|^2$$

Flowchart of NCE



Coherence Encoder (CE)

- **Encoder Phase**
- PCA
- CCA

- **Decoder Phase**
- Linear ANNs
- PCA Reconstruction
- Non-linear ANNs

CCA

- $Z = [z_x{}^T \;\; z_y{}^T]^T$
- Covariance matrix

$$R_{zz} = \begin{bmatrix} R_{z_x z_x} & R_{z_x z_y} \\ R_{z_y z_x} & R_{z_y z_y} \end{bmatrix}$$

- Canonical variables

$$u = \left(F^T R_{z_x z_x}^{-\frac{1}{2}} z_x \right), \qquad v = \left(G^T R_{z_y z_y}^{-\frac{1}{2}} z_y \right)$$

- Dependence

$$L = \frac{det(R_{zz})}{det(R_{z_x z_x}) det(R_{z_y z_y})} = \frac{det\left(\begin{bmatrix} R_{z_x z_x} & R_{z_x z_y} \\ R_{z_y z_x} & R_{z_y z_y} \end{bmatrix} \right)}{det(R_{z_x z_x}) det(R_{z_y z_y})}$$

- Coherence

$$H = 1 - L = 1 - \frac{det\left(\begin{bmatrix} R_{z_x z_x} & R_{z_x z_y} \\ R_{z_y z_x} & R_{z_y z_y} \end{bmatrix} \right)}{det(R_{z_x z_x}) det(R_{z_y z_y})}$$

Linear ANNs

- maps u to z'_y and v to z'_x

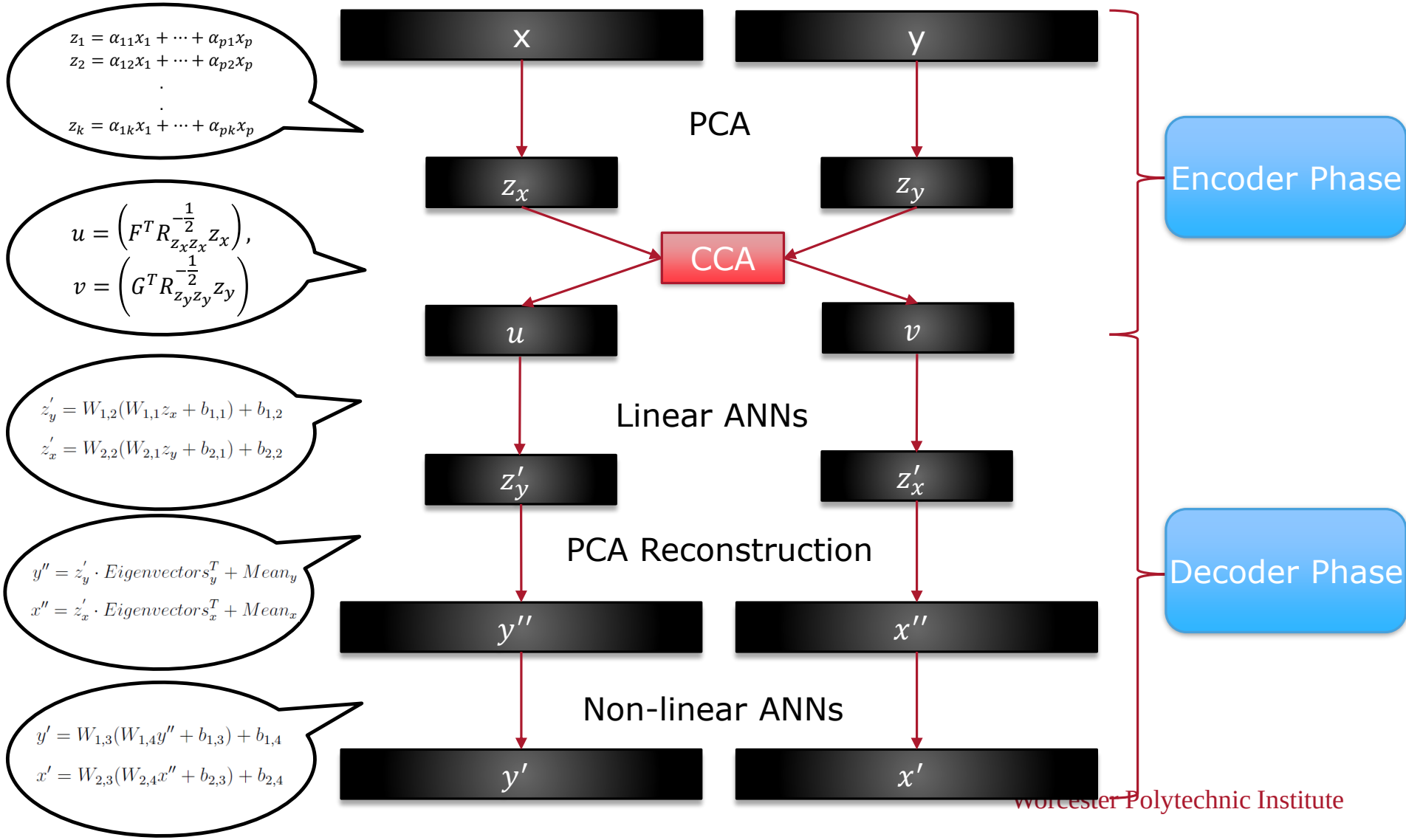
$$z'_y = W_{1,2}(W_{1,1}u + b_{1,1}) + b_{1,2}$$

$$z'_x = W_{2,2}(W_{2,1}v + b_{2,1}) + b_{2,2}$$

$$\gamma_1(W_{1,1}, W_{1,2}, b_{1,1}, b_{1,2}) = \underset{W_{1,1},W_{1,2},b_{1,1},b_{1,2}}{\operatorname{argmin}} \left\| z_y - z'_y \right\|^2$$

$$\gamma_2(W_{2,1}, W_{2,2}, b_{2,1}, b_{2,2}) = \underset{W_{2,1},W_{2,2},b_{2,1},b_{2,2}}{\operatorname{argmin}} \left\| z_x - z'_x \right\|^2$$

Flowchart of CE



Kernel Coherence Encoder (KCE)

- **Encoder Phase**
- PCA
- KCCA

- **Decoder Phase**
- Linear ANNs
- PCA Reconstruction
- Non-linear ANNs

KCCA

- obtains canonical variables u and v
- trains an optimal kernel function

$$K(z_{x,i}, z_{y,j}) = e^{-\alpha\|z_{x,i} - z_{y,j}\|^2 + \beta}$$

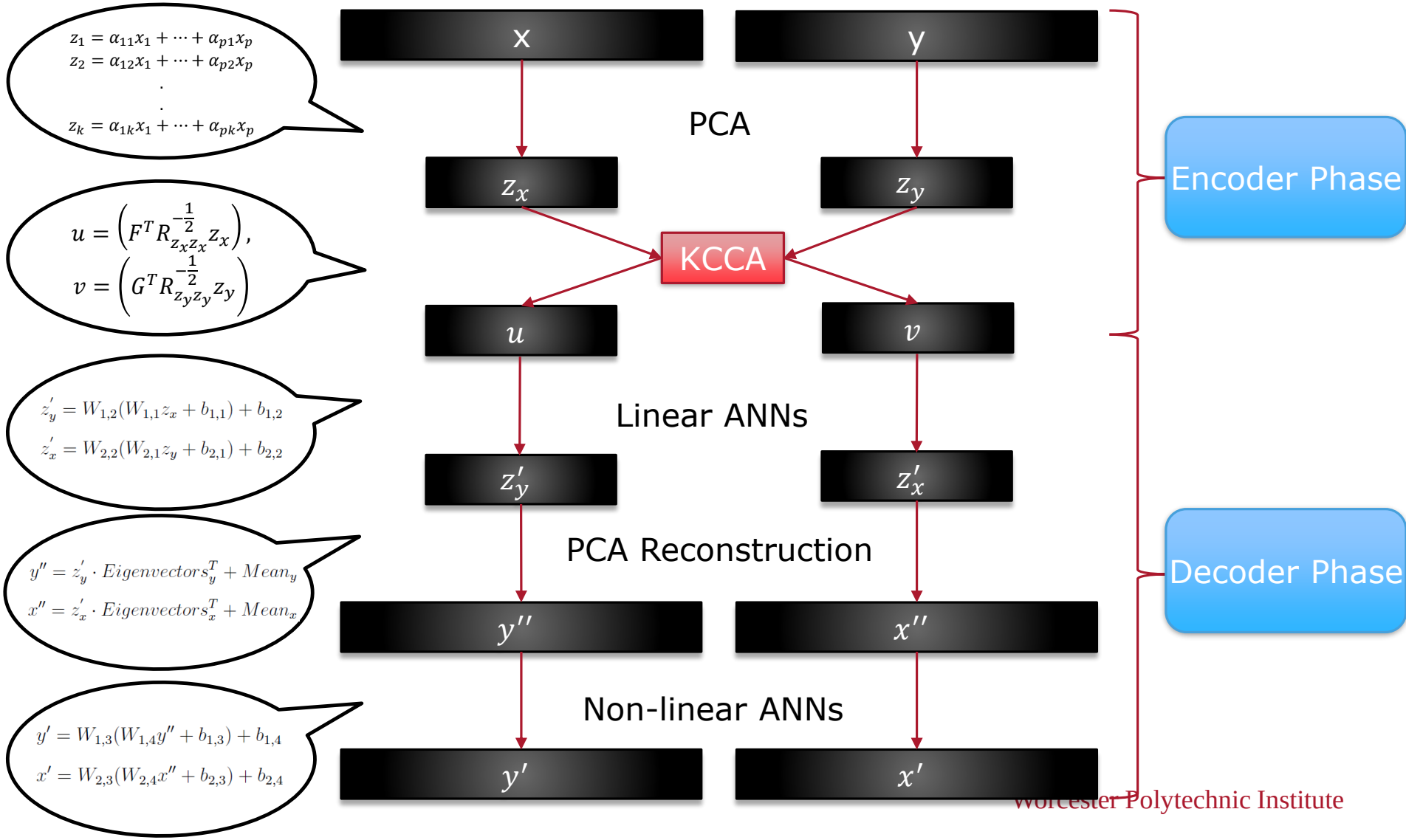
- Element-wise coherence calculation
- We define κ as the kernel function training process

$$\kappa = \underset{\alpha,\beta}{\operatorname{argmin}} L$$

Element-wise Coherence

- Proposing KCCA method and the element-wise coherence calculation to autoencoders in this way is novel.
- The connection between x and y comes only from the element-wise coherence between z_x and z_y in the Reproducing Kernel Hilbert Space.

Flowchart of KCE



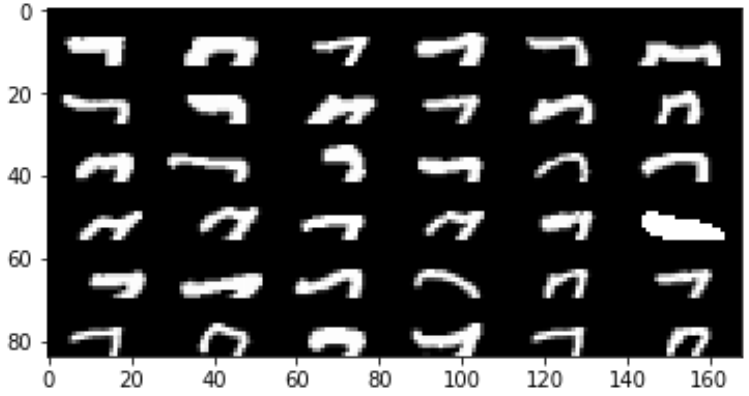
Experiment – MNIST Data Set

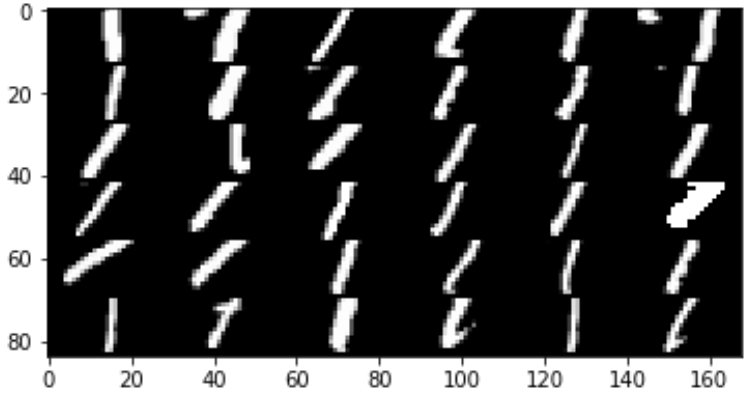
- MNIST data set, 55000 for training and 10000 for testing
- 28 * 28 = 784 pixels for each image, grayscale 0-1

Picture resource: Cheng, Chunling, et al. "A multilayer improved RBM network based image compression method in wireless sensor networks." International Journal of Distributed Sensor Networks 12.3 (2016): 1851829.

Experiment – MNIST Data Set

- Cut each image to upper and lower parts - x and y
- Group by digit
- Each image 14 * 28 = 392 dimension





Training

- TensorFlow*
- Gradient Descent Optimization*

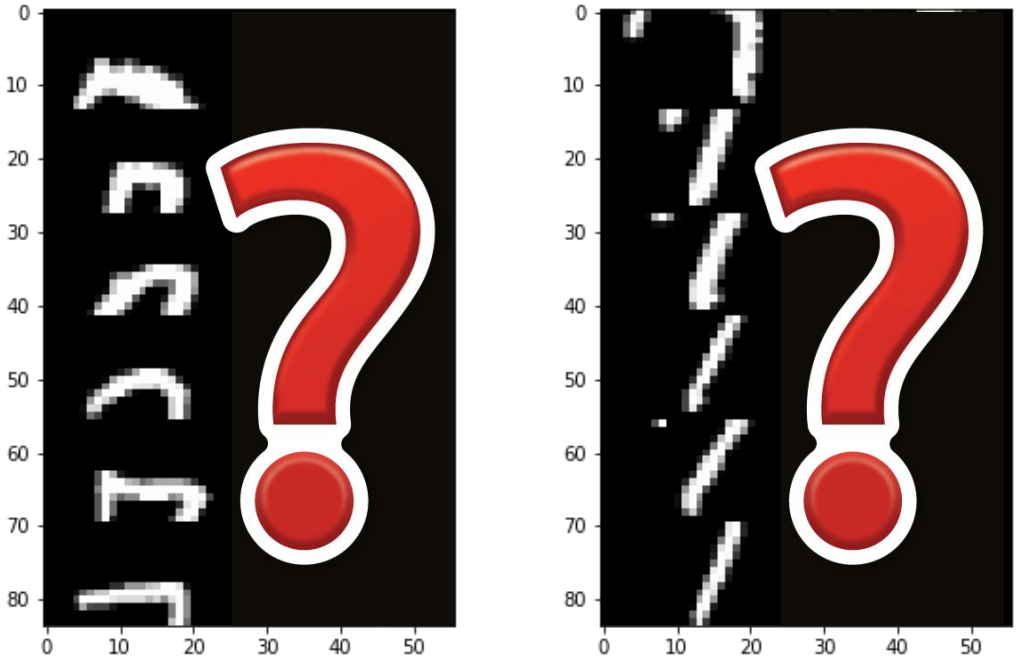
  tf.train.GradientDescentOptimizer(learning rate).minimize(cost)
- Training parameters:
- hidden layer dimension = 100
- learning rate = 0.01
- training epochs = 2000
- batch size = 100

*See back-up slides for TensorFlow and Gradient Descent Optimization

Experiment – MNIST Data Set

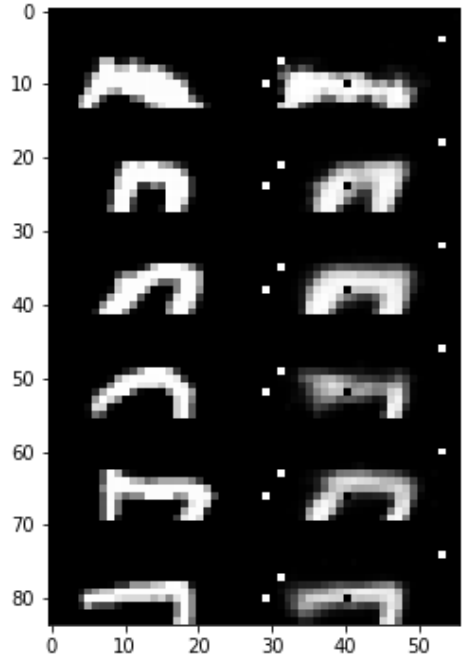
- For each digit, 6 random samples for test (12 reconstructed images)

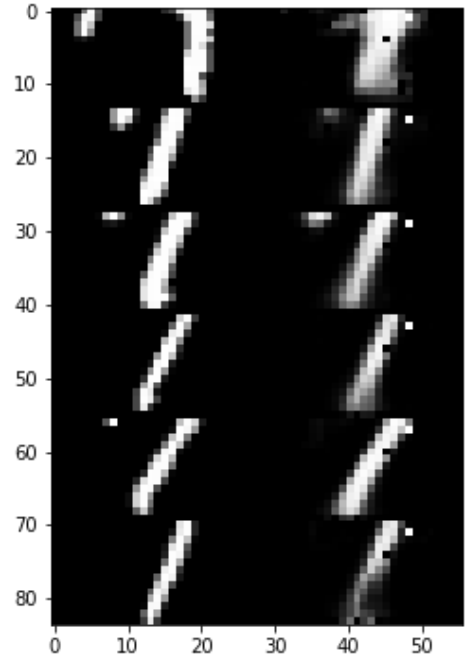
1. "Paired Autoencoders"
2. NCE
3. CE
4. KCE



Experiment – MNIST Data Set

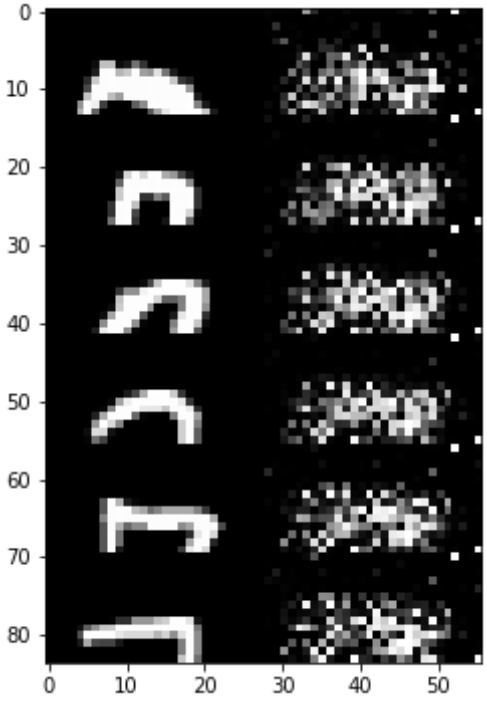
- Mimic Autoencoders
- Directly mapping between x and y
- Two independent ANNs

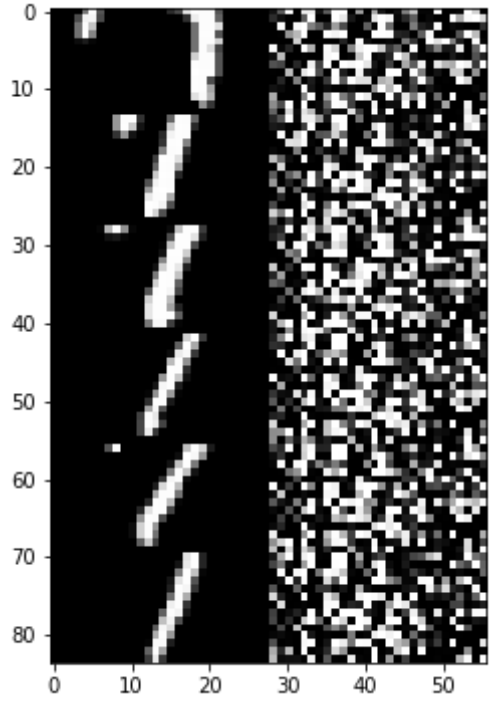




NCE Reconstruction

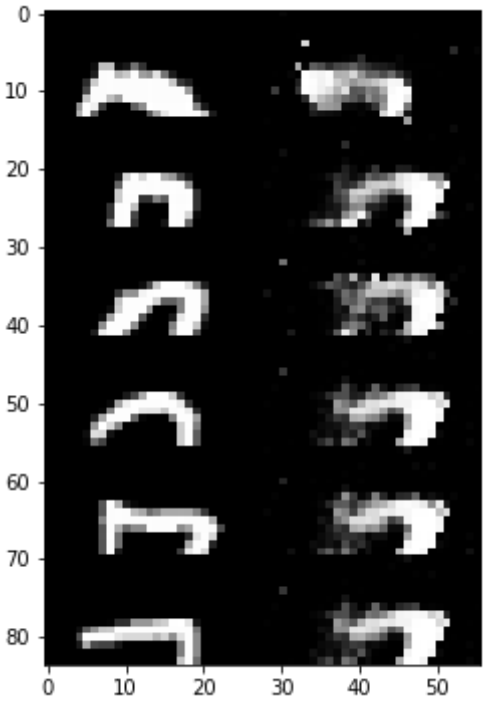
- Patterns are hardly reconstructed

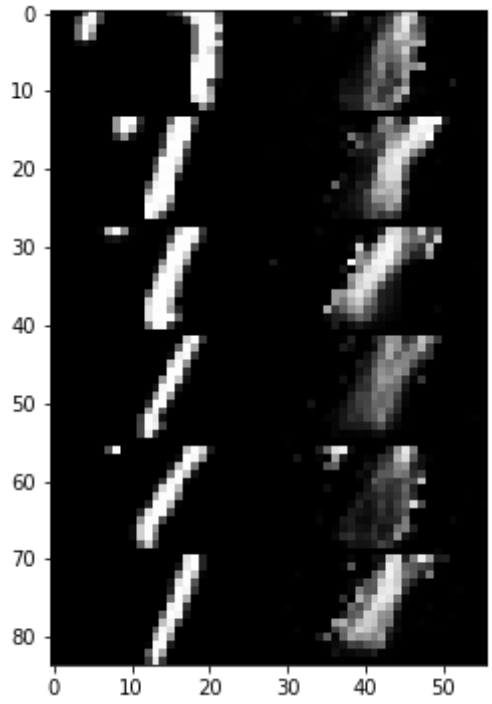




CE Reconstruction

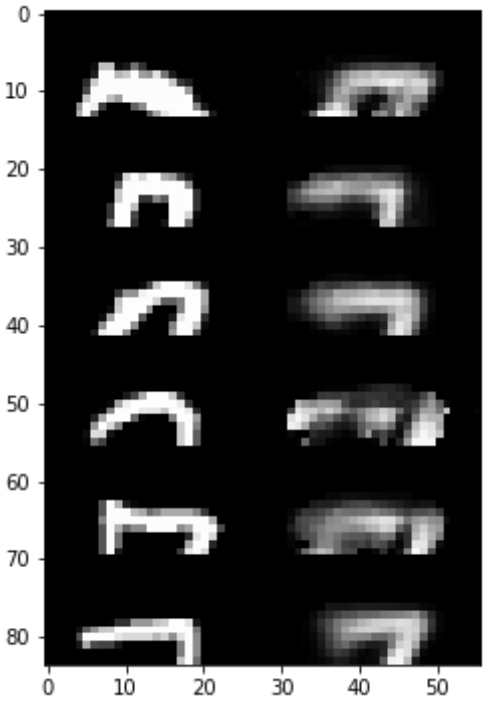
- Patterns are reconstructed, with some flaws

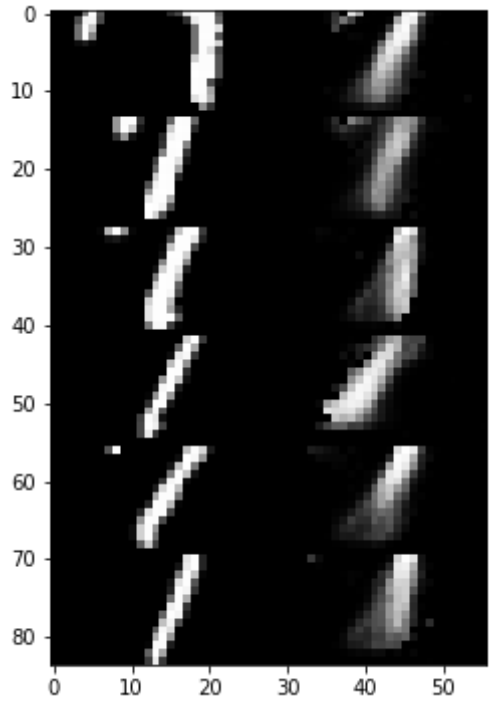




KCE Reconstruction

- Patterns are reconstructed, with fewer flaws





Quantitative Measures

- Support visual observations
- Equivalent to the Image Similarity Analysis
- Totally 120 samples for all digits
- 5 metrics:*

1. L^2 norm (or Frobenius norm)
2. Pearson correlation score
3. Cross-correlation
4. Bhattacharyya distance
5. Fast Fourier Transform (FFT) rank

- 2-sample t-test on results of the above 5 measures, NCE vs KCE and CE vs KCE. (Only include these three models because they have the same structures for decoder phase)

*See back-up slides for detailed descriptions of the 5 metrics

Quantitative Measures (For All Digits)

	metric	NCE average	KCE average	p-value
0	2-norm	8.3188	4.94215	4.15186e-37
1	Pearson Coefficient	0.295733	0.618759	2.35119e-32
2	cross correlation	0.344092	0.712149	5.46184e-61
3	Bhattacharyya distance	0.693693	0.505529	2.48646e-33
4	fft	0.204558	0.4972	2.14213e-30

	metric	CE average	KCE average	p-value
0	2-norm	5.41653	4.94215	0.00297353
1	Pearson Coefficient	0.550588	0.618759	0.00663332
2	cross correlation	0.689174	0.712149	0.113121
3	Bhattacharyya distance	0.539608	0.505529	0.0272345
4	fft	0.42421	0.4972	0.00352892

Summary

- We developed 3 novel models
- NCE leverages the non-linear abilities of autoencoders, but is not mathematically well principled
- CE is mathematical well principled, but linear
- KCE combines the advantages of both
  - It leverages the full power of non-linear autoencoders and kernel methods
  - It mathematically well-founded
- In an example of mathematical rigor being useful, KCE actually works better than the other two models

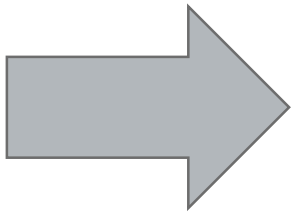
Future Works

- First, we believe that during the PCA projection, some information of original data gets lost. We will run more experiments to explore the dimensionality trade-off for better reconstruction results.
- Second, we have observed that our model is able to avoid the "black-to-white" mapping error problem. It is worth exploring its reason and continuing to produce a mathematical proof.
- Third, the current kernel function might be too simple (one layer with 2 parameters). We are going to explore the positive definite kernel functions with multiple layers (true "deep learning").
- Finally, so far we have only done the experiments on MNIST data. We should explore more samples including face images or even other types of data to justify the effectiveness of our proposed model KCE.

Future Works

- Reconstruct more complicated images!!!

Randy at 20

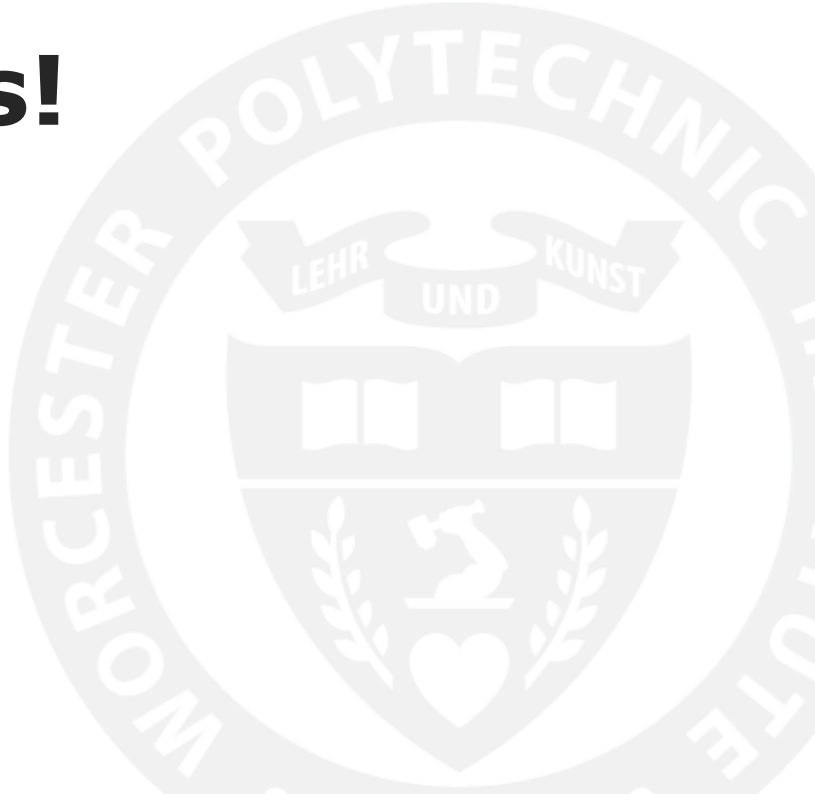


Picture provided by professor Randy Paffenroth


WORCESTER POLYTECHNIC INSTITUTE
LEHR
UND
KUNST
1865

WPI

Thanks!


WORCESTER POLYTECHNIC INSTITUTE
LEHR
UND
KUNST

Reproducing Kernel Hilbert Space (RKHS)

- In KPCA or KCCA, a kernel function maps data from original space to an inner product space RKHS to ensure both the existence of an inner product and the evaluation of every function in this space at every point in the domain.
- The corresponding kernel function should be both symmetric and positive definite

Positive Definite Kernel

- Suppose a kernel function k is not positive definite, it may not represent an inner product in any Hilbert space.
- Here is one way to see: A kernel k is positive definite if and only if for all samples of n points, its corresponding kernel matrix K is a positive definite matrix. With a positive definite K, by Cholesky decompose $K = LL^T$, each row of L is one mapped point in the inner product space.
- If k is not positive definite, the matrix K may also not be positive definite. Consequently, Cholesky does not work and there is no corresponding inner product space.

Mercer's Theorem

- **Definition:** Suppose K is a continuous symmetric non-negative definite kernel. Then there is an orthonormal basis $\{e_i\}_i$ of $L^2[a, b]$ consisting of eigenfunctions of T_K such that the corresponding sequence of eigenvalues $\{\lambda_i\}_i$ is nonnegative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[a, b]$ and K has the representation

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j \, e_j(s) e_j(t)$$

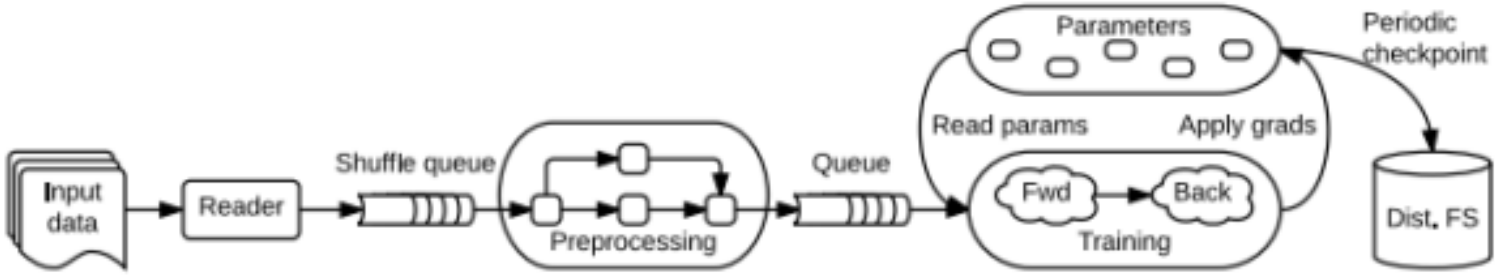
- where the convergence is absolute and uniform.

Completely Monotonic Function

- **Definition:** A function $f(x)$ with domain $(0,\infty)$ is said to be completely monotonic, if it possesses derivatives $f^{(n)}(x)$ for all n=0,1,2,3,... and if $(-1)^n f^{(n)}(x) \geq 0$.

TensorFlow

- an open source software library for numerical computation using data flow graphs, machine learning system that operates at large scale and in heterogeneous environments.
- comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.
- gives great flexibility to the application developer and enables developers to experiment with novel optimizations and training algorithms.



Gradient Descent Optimization

- the most common way to optimize artificial neural networks
- Batch gradient descent is applied by TensorFlow class

$$for\ i\ in\ range(\#\ epochs):$$

$$params\_grad = evaluate\_gradient(cost\_function, data, params)$$

$$params = params - learning\_rate * params\_grad$$

$$G(W_i) = \frac{\partial\ cost}{\partial\ h}\ \frac{\partial\ h}{\partial f(W_i)}\ \frac{\partial f(W_i)}{dW_i}$$

$$G(b_i) = \frac{\partial\ cost}{\partial\ h}\ \frac{\partial\ h}{\partial f(b_i)}\ \frac{\partial f(b_i)}{db_i}$$

$$W_{i+1} = W_i - \alpha G(W_i)$$

$$b_{i+1} = b_i - \alpha G(b_i)$$

L^2 Norm

$$\|x - x'\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2}$$

$$\|y - y'\|_2 = \sqrt{\sum_{i=1}^{n} (y_i - y'_i)^2}$$

Pearson Correlation Score

- Pearson Correlation Coefficient is a measure of the linear correlation between two variables x and y. It has a value between -1 and +1, where 1 indicates total positive linear correlation, 0 is no linear correlation, and −1 is the total negative linear correlation.
- Accordingly, Pearson Correlation score measures how highly correlated are two variables. A score of 1 indicates that the data objects are perfectly correlated but in this case, a score of -1 means that the data objects are not correlated.
- In other words, the Pearson Correlation score quantifies how well two data objects fit a line.

$$Pearson(x, x') = \frac{\sum xx' - \frac{\sum x \sum x'}{n}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{n})(\sum x'^2 - \frac{(\sum x')^2}{n})}}$$

$$Pearson(y, y') = \frac{\sum yy' - \frac{\sum y \sum y'}{n}}{\sqrt{(\sum y^2 - \frac{(\sum y)^2}{n})(\sum y'^2 - \frac{(\sum y')^2}{n})}}$$

Cross-Correlation

- In signal processing, cross-correlation measures similarity of two series as a function of the displacement of one relative to the other.
- For 1-dimensional data, it's same as Pearson Correlation Coefficient.
- For 2-dimensional data, this measure has one advantage: it takes care of the data shifting just like cross-correlation method takes care of the time-delay between signals in signal processing.
- Here cc2d are two arrays containing cross-correlation coefficients (range -1 to 1) in different phases. We select the maximum values

$$CC2d_x = c2d(\frac{x - \overline{x}}{\sqrt{var(x)}}, \frac{x' - \overline{x'}}{\sqrt{var(x')}})/(m \cdot n)$$

$$CC2d_y = c2d(\frac{y - \overline{y}}{\sqrt{var(y)}}, \frac{y' - \overline{y'}}{\sqrt{var(y')}})/(m \cdot n)$$

Bhattacharyya Distance

- Measures the similarity of two discrete or continuous probability distributions.
- In image processing, this measure can be used to determine the relative closeness of the two samples being considered.
- A zero Bhattacharyya distance means that two data are exactly the same. Larger Bhattacharyya distance refers to greater gap or difference between them.

$$d(x, x') = \sqrt{1 - \frac{1}{\sqrt{\overline{x}\overline{x'}n^2}} \sum_{i=1}^{n} \sqrt{x_i x'_i}}$$

$$d(y, y') = \sqrt{1 - \frac{1}{\sqrt{\overline{y}\overline{y'}n^2}} \sum_{i=1}^{n} \sqrt{y_i y'_i}}$$

FFT Rank

- FFT is an algorithm that samples a signal (or signal-like data) over a period of time (or space) and transforms it into its frequency domain.
- In the frequency domain, each point represents a particular frequency contained in the spatial domain. We use the real part of FFT results (magnitude).
- $\overline{F}_x, \overline{F}_y, \overline{F}_{x'}, \overline{F}_{y'}$ represent the average frequency values in frequency domain.
- A rank ranges from -1 to 1, where 1 is obtained if two datasets are exactly the same and -1 if they are fully independent from each other.

$$rank(x, x') = real\left(\frac{(\sum_{i=1}^{n} x_i x'_i - n\overline{F}_x \overline{F}_{x'})^2}{(\sum_{i=1}^{n} |x_i|^2 - n\overline{F}_x^2)(\sum_{i=1}^{n} |x'_i|^2 - n\overline{F}_{x'}^2)}\right)$$

$$rank(y, y') = real\left(\frac{(\sum_{i=1}^{n} y_i y'_i - n\overline{F}_y \overline{F}_{y'})^2}{(\sum_{i=1}^{n} |y_i|^2 - n\overline{F}_y^2)(\sum_{i=1}^{n} |y'_i|^2 - n\overline{F}_{y'}^2)}\right)$$