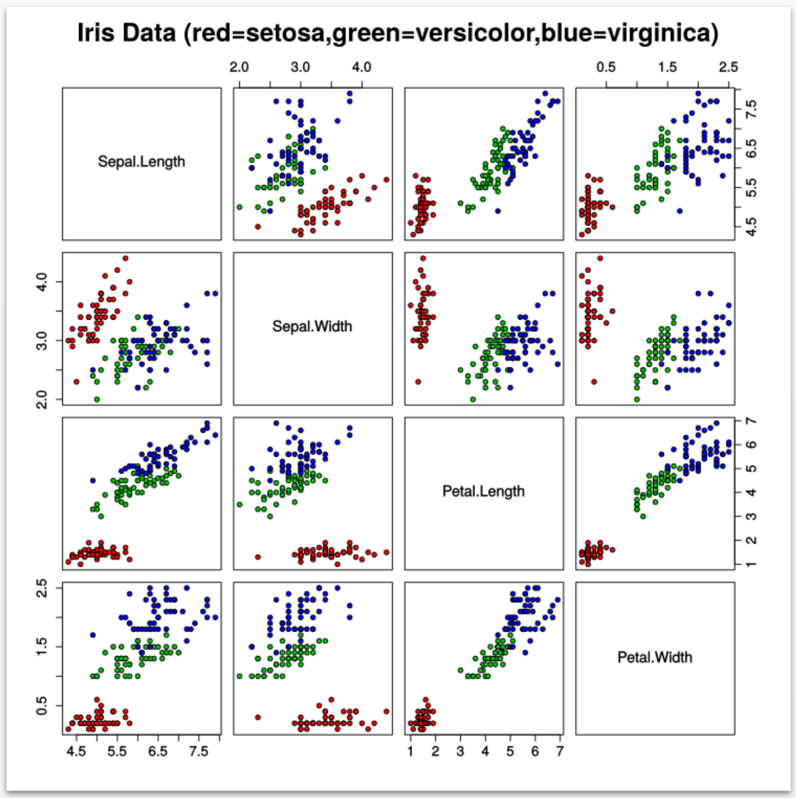
Dimensionality Reduction

Shantanu Jain

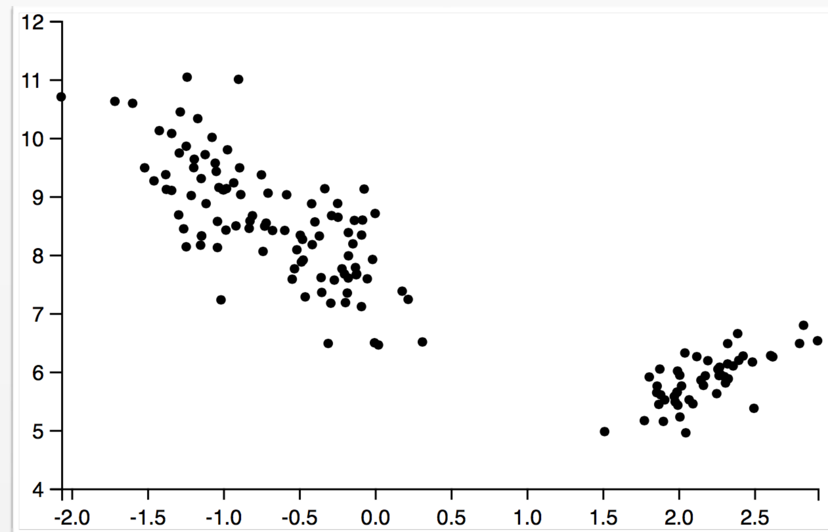
Dimensionality Reduction

Goal: Map high dimensional data onto lower-dimensional data in a manner that preserves *distances/similarities*

Original Data (4 dims)



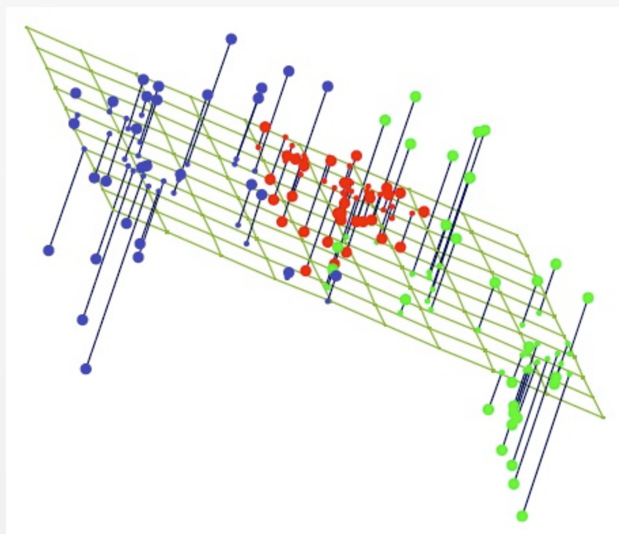
Projection with PCA (2 dims)



Objective: projection should "preserve" relative distances

Linear Dimensionality Reduction

Idea: Project high-dimensional vector
onto a lower dimensional space



$$\mathbf{x} \in \mathbb{R}^{361}$$

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

$$\mathbf{z} \in \mathbb{R}^{10}$$

Problem Setup

Given n data points in d dimensions: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Problem Setup

Given n data points in d dimensions: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Want to reduce dimensionality from d to k

Choose k directions $\mathbf{u}_1, \ldots, \mathbf{u}_k$

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x} \qquad \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Problem Setup

Given n data points in d dimensions: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Want to reduce dimensionality from d to k

Choose k directions $\mathbf{u}_1, \ldots, \mathbf{u}_k$

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x} \qquad \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

For each $\mathbf{u}_j$, compute "similarity" $z_j = \mathbf{u}_j^\top \mathbf{x}$

Problem Setup

Given n data points in d dimensions: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Want to reduce dimensionality from d to k

Choose k directions $\mathbf{u}_1, \ldots, \mathbf{u}_k$

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

For each $\mathbf{u}_j$, compute "similarity" $z_j = \mathbf{u}_j^\top \mathbf{x}$

Project $\mathbf{x}$ down to $\mathbf{z} = (z_1, \ldots, z_k)^\top = \mathbf{U}^\top \mathbf{x}$

How to choose $\mathbf{U}$?

Background: Changes of Basis

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\bar{\mathbf{U}} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_{\mathrm{d}} \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times \mathrm{d}}$$

$\bar{z} = \bar{U}^T x$ is a representation of x w.r.t. the basis vectors in $\bar{U}$

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{i,j} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}$$

$$\bar{U}^T \bar{U} = I_{d \times d}$$

Background: Changes of Basis

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\bar{\mathbf{U}} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_\mathrm{d} \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times \mathrm{d}}$$

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{i,j}$$

Background: Changes of Basis

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\bar{\mathbf{U}} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_\mathsf{d} \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times \mathsf{d}}$$

Change of basis

$$\bar{\mathbf{z}} = (z_1, \ldots, z_\mathsf{d})^\top$$

$$z_j = \mathbf{u}_j^\top \mathbf{x}$$

$$\bar{\mathbf{z}} = \bar{\mathbf{U}}^\top \mathbf{x}$$

Inverse Change of basis

$$\mathbf{x} = \bar{\mathbf{U}}\bar{\mathbf{z}} = \sum_{j=1}^{\mathsf{d}} z_j \mathbf{u}_j$$

Properties of orthonormal matrices

For an orthonormal matrix $\bar{U} \in \mathbf{R}^{d \times d}$

$$\bar{U}^T \bar{U} = \bar{U} \bar{U}^T = I_{d \times d}$$

An orthonormal matrix has d orthogonal vectors of dimension d and unit length

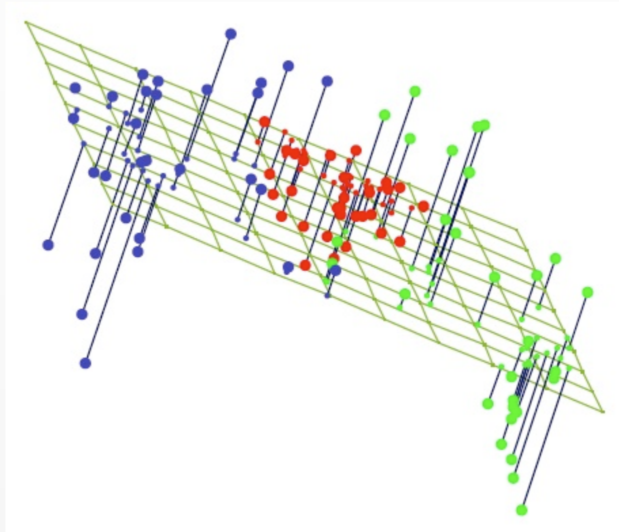
For a semi-orthonormal matrix $U \in \mathbf{R}^{d \times k}$, where $k < d$

$$U^T U = I_{k \times k}$$

$$U U^T \neq I_{d \times d}$$

An semi-orthonormal matrix has k orthogonal vectors of dimension d and unit length

Principal Component Analysis



$$\mathbf{x} \in \mathbb{R}^{361}$$

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

$$\mathbf{z} \in \mathbb{R}^{10}$$

U is $d \times k$

We are back to the PCA setting with U containing fewer than d columns

Optimize two equivalent objectives

1. Minimize the reconstruction error
2. Maximizes the projected variance

PCA Objective 1: Reconstruction Error

$\mathbf{U}$ serves two functions:

- Encode: $\mathbf{z} = \mathbf{U}^\top \mathbf{x}, \qquad z_j = \mathbf{u}_j^\top \mathbf{x}$

PCA Objective 1: Reconstruction Error

$\mathbf{U}$ serves two functions:

- Encode: $\mathbf{z} = \mathbf{U}^\top \mathbf{x}, \quad z_j = \mathbf{u}_j^\top \mathbf{x}$
- Decode: $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{z} = \sum_{j=1}^{k} z_j \mathbf{u}_j$

PCA Objective 1: Reconstruction Error

$\mathbf{U}$ serves two functions:

- Encode: $\mathbf{z} = \mathbf{U}^\top \mathbf{x}$, $\quad z_j = \mathbf{u}_j^\top \mathbf{x}$
- Decode: $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{z} = \sum_{j=1}^{k} z_j \mathbf{u}_j$

Want reconstruction error $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ to be small

PCA Objective 1: Reconstruction Error

$\mathbf{U}$ serves two functions:

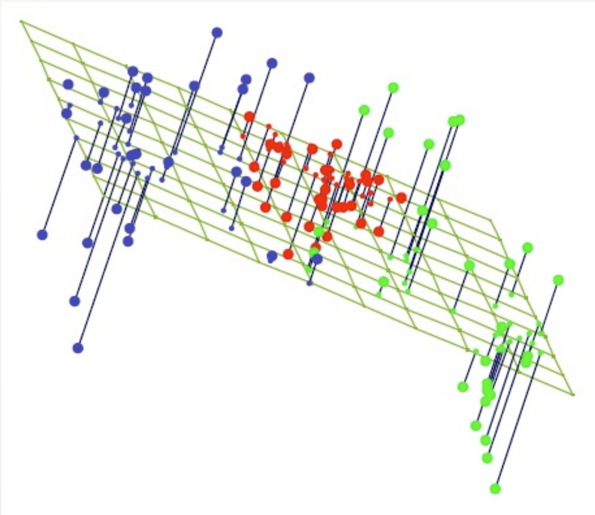
- Encode: $\mathbf{z} = \mathbf{U}^\top \mathbf{x}$, $\quad z_j = \mathbf{u}_j^\top \mathbf{x}$
- Decode: $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{z} = \sum_{j=1}^{k} z_j \mathbf{u}_j$

Want reconstruction error $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ to be small

Objective: minimize total squared reconstruction error

$$\min_{U \in \mathbf{R}^{d\times k}, U^T U = I} \mathrm{RE}(U)$$

$$\begin{aligned}\mathrm{RE}(U) &= \frac{1}{n}\sum_{i=1}^{n} \| x_i - UU^T x_i \|^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \| (I - UU^T) x_i \|^2 \\ &= \hat{\mathbf{E}}\left[\| (I - UU^T) x \|^2 \right]\end{aligned}$$



Mathematically, the expectation is w.r.t the empirical distribution of the data that gives an equal probability of 1/n to each point.

Total Variance

- Define the **Total Variance** of $x \in \mathbf{R}^d$ as the sum of variances across all dimensions.
- It is estimated from the observed data as the sum of the diagonal elements of the covariance matrix

$$\text{Var}_T(x) = \text{tr}\left(\frac{1}{n}XX^T\right)$$

- It can also be expressed as

$$\text{Var}_T(x) = \frac{1}{n}\sum_{i=1}^{n} \| x_i \|^2$$

$$= \hat{\mathbf{E}}\left[\| x \|^2\right]$$

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Assuming that the matrix X is centered; $\hat{\mathbf{E}}[x] = \dfrac{1}{n}\sum_{i=1}^{n} x_i = 0$

$$\| x_i \|^2 = x_{i1}^2 + x_{i2}^2 + \ldots x_{id}^2$$

Variance across dimension j

$$\text{Var}(x_{\cdot j}) = \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2$$

This is because the mean for each dimension is 0.

Projected Variance

- Let $z = U^T x$ be the projection of x

$$\text{Var}_T(z) = \text{tr}\left(\frac{1}{n} Z Z^T\right)$$

$$= \text{tr}\left(\frac{1}{n} U^T X X^T U\right)$$

- It can also be expressed as

$$\text{Var}_T(z) = \frac{1}{n} \sum_{i=1}^{n} \| U^T x_i \|^2$$

$$= \hat{\mathbf{E}}\left[\| U^T x \|^2\right]$$

$$\max_{U \in \mathbf{R}^{d \times k}, U^T U = I} \text{Var}_T(z; U)$$

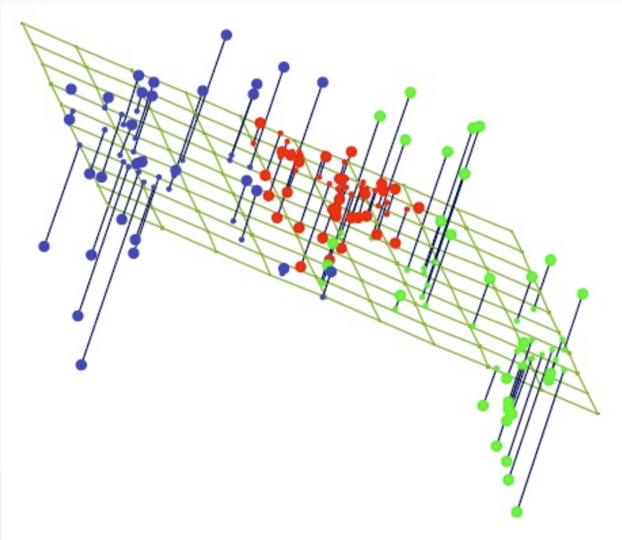
$Z = U^T X$: contains the projections of all points

$Z = [z_1, z_2, \ldots, z_n], z_i \in \mathbf{R}^k$

Note that the variance formulas are true for the Z matrix as well since $\hat{\mathbf{E}}[z] = \hat{\mathbf{E}}[U^T x] = U^T \hat{\mathbf{E}}[x] = 0$

The steps above come from linearity of expectation and because we have assumed that X is centered; i.e., $\hat{\mathbf{E}}[x] = 0$

Projected Variance



$$\mathbf{x} \in \mathbb{R}^{361}$$

$$\downarrow \ \mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

$$\mathbf{z} \in \mathbb{R}^{10}$$

Equivalence of two objectives

Key intuition:

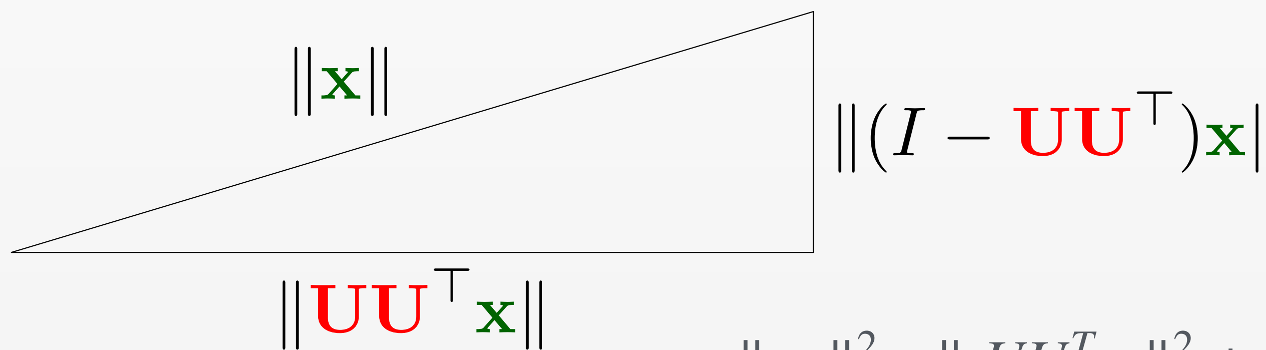
$$\underbrace{\text{variance of data}}_{\text{fixed}} = \underbrace{\text{captured variance}}_{\text{want large}} + \underbrace{\text{reconstruction error}}_{\text{want small}}$$

Equivalence of two objectives

Key intuition:

$$\underbrace{\text{variance of data}}_{\text{fixed}} = \underbrace{\text{captured variance}}_{\text{want large}} + \underbrace{\text{reconstruction error}}_{\text{want small}}$$

Pythagorean decomposition: $\mathbf{x} = \mathbf{U}\mathbf{U}^\top\mathbf{x} + (I - \mathbf{U}\mathbf{U}^\top)\mathbf{x}$



$$\| x \|^2 = \| UU^T x \|^2 + \| (I - UU^T)x \|^2$$
$$= \| U^T x \|^2 + \| (I - UU^T)x \|^2$$

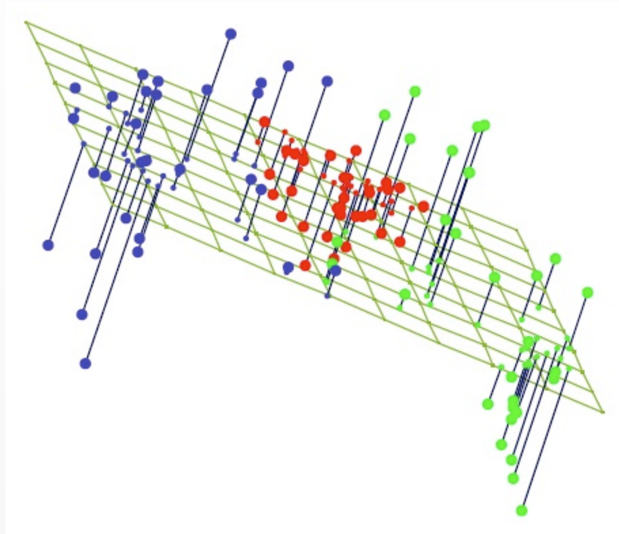
Take expectations;

$$\hat{\mathbb{E}}[\|\mathbf{x}\|^2] = \hat{\mathbb{E}}[\|\mathbf{U}^\top\mathbf{x}\|^2] + \hat{\mathbb{E}}[\|\mathbf{x} - \mathbf{U}\mathbf{U}^\top\mathbf{x}\|^2]$$

Dimensionality Reduction

Shantanu Jain

Principal Component Analysis



$$\mathbf{x} \in \mathbb{R}^{361}$$

$$\downarrow \; \mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

$$\mathbf{z} \in \mathbb{R}^{10}$$

Optimize two equivalent objectives

1. Minimize the reconstruction error

$$\hat{\mathbb{E}}[||\boldsymbol{x} - \boldsymbol{U}\boldsymbol{z}||^2] = \hat{\mathbb{E}}[||(I - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}||^2]$$

2. Maximizes the projected variance

$$\hat{\mathbb{E}}[\boldsymbol{z}^\top \boldsymbol{z}] = \hat{\mathbb{E}}[\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}]$$

Total variance unaltered by basis change

$$\bar{z}^T\bar{z} = x^T(\bar{U}\bar{U}^T)x = x^Tx$$

$\bar{U}\bar{U}^T = I_{d\times d}$

$\bar{U}^{-1} = \bar{U}^T$ when $\bar{U}$ contains all d orthonormal basis, otherwise the inverse in undefined

$$\hat{\mathbf{E}}[\, \| \bar{z} \|^2 \,] = \hat{\mathbf{E}}[\, \| x \|^2 \,]$$

$$\mathrm{Var}_T(x) = \mathrm{Var}_T(\bar{z})$$

$$\mathrm{tr}\left(\frac{1}{n}XX^T\right) = \mathrm{tr}\left(\frac{1}{n}\bar{U}^TXX^T\bar{U}\right)$$

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d\times n}$$

Orthonormal Basis

$$\bar{\mathbf{U}} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_\mathrm{d} \\ | & & | \end{pmatrix} \in \mathbb{R}^{d\times \mathrm{d}}$$

Change of basis

$$\mathbf{z} = \bar{\mathbf{U}}^\top\mathbf{x} \qquad \mathbf{x} = \bar{\mathbf{U}}\mathbf{z}$$

$$\bar{U}^T\bar{U} = I_{d\times d}$$

Eigenvectors of the Covariance

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Orthonormal Basis

$$\bar{\mathbf{U}} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_d \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times d}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X} \mathbf{X}^\top$$

$$\mathbf{C}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$

$$C\bar{U} = \bar{U}\Lambda$$

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_d \end{pmatrix}$$

Claim: Eigenvectors of a symmetric matrix are orthogonal

Proof: Eigenvectors are Orthogonal

For any real matrix A and any vectors $\mathbf{x}$ and $\mathbf{y}$, we have

$$\langle A\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, A^T\mathbf{y}\rangle.$$

Now assume that A is symmetric, and $\mathbf{x}$ and $\mathbf{y}$ are eigenvectors of A corresponding to distinct eigenvalues λ and μ. Then

$$\lambda\langle \mathbf{x}, \mathbf{y}\rangle = \langle \lambda\mathbf{x}, \mathbf{y}\rangle = \langle A\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, A^T\mathbf{y}\rangle = \langle \mathbf{x}, A\mathbf{y}\rangle = \langle \mathbf{x}, \mu\mathbf{y}\rangle = \mu\langle \mathbf{x}, \mathbf{y}\rangle.$$

Therefore, $(\lambda - \mu)\langle \mathbf{x}, \mathbf{y}\rangle = 0$. Since $\lambda - \mu \neq 0$, then $\langle \mathbf{x}, \mathbf{y}\rangle = 0$, i.e., $\mathbf{x} \perp \mathbf{y}$.

Now find an orthonormal basis for each eigenspace; since the eigenspaces are mutually orthogonal, these vectors together give an orthonormal subset of $\mathbb{R}^n$. Finally, since symmetric matrices are diagonalizable, this set will be a basis (just count dimensions). The result you want now follows.

(from stack exchange)

Eigenvectors of the Covariance

$$C = \frac{1}{n} X X^T$$

Eigen-decomposition

$$\boldsymbol{C} = \bar{\boldsymbol{U}} \boldsymbol{\Lambda} \bar{\boldsymbol{U}}^\top$$

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_d \end{pmatrix}$$

$$C\bar{U} = \bar{U}\Lambda$$

$$\Rightarrow C\bar{U}\bar{U}^T = \bar{U}\Lambda\bar{U}^T$$

$$\Rightarrow C = \bar{U}\Lambda\bar{U}^T$$

Because $\bar{U}$ is orthonormal matrix containing all d orthonormal basis

Total variance

- Consider a change of basis with the orthogonal matrix containing the eigen-vectors
  - $\bar{z} = \bar{U}^T x$
- Now the covariance of $\bar{z}$ is given by

$\bar{U}$ contains all d eigenvectors of $\frac{1}{n}XX^T$, which is orthonormal by definition.

$$C_{\bar{z}} = \frac{1}{n}\bar{Z}\bar{Z}^T$$

$\bar{Z} = \bar{U}^T X$, containing the transformed dataset

- $$= \frac{1}{n}\bar{U}^T X X^T \bar{U}$$
  $$= \bar{U}^T \bar{U} \Lambda \bar{U}^T \bar{U}$$
  $$= \Lambda$$

Because $\frac{1}{n}XX^T = U\Lambda U^T$

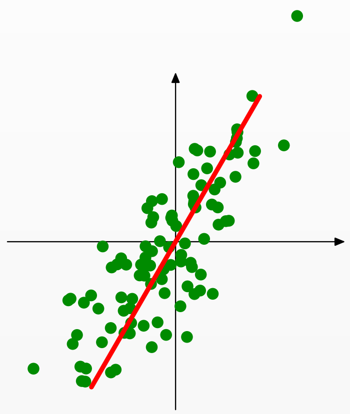
- The covariance matrix of the transformed points is diagonal. In other words the new dimensions are uncorrelated
- The variance across the i^{th} new dimensions is given by λ_i

Because U is orthonormal (and full).

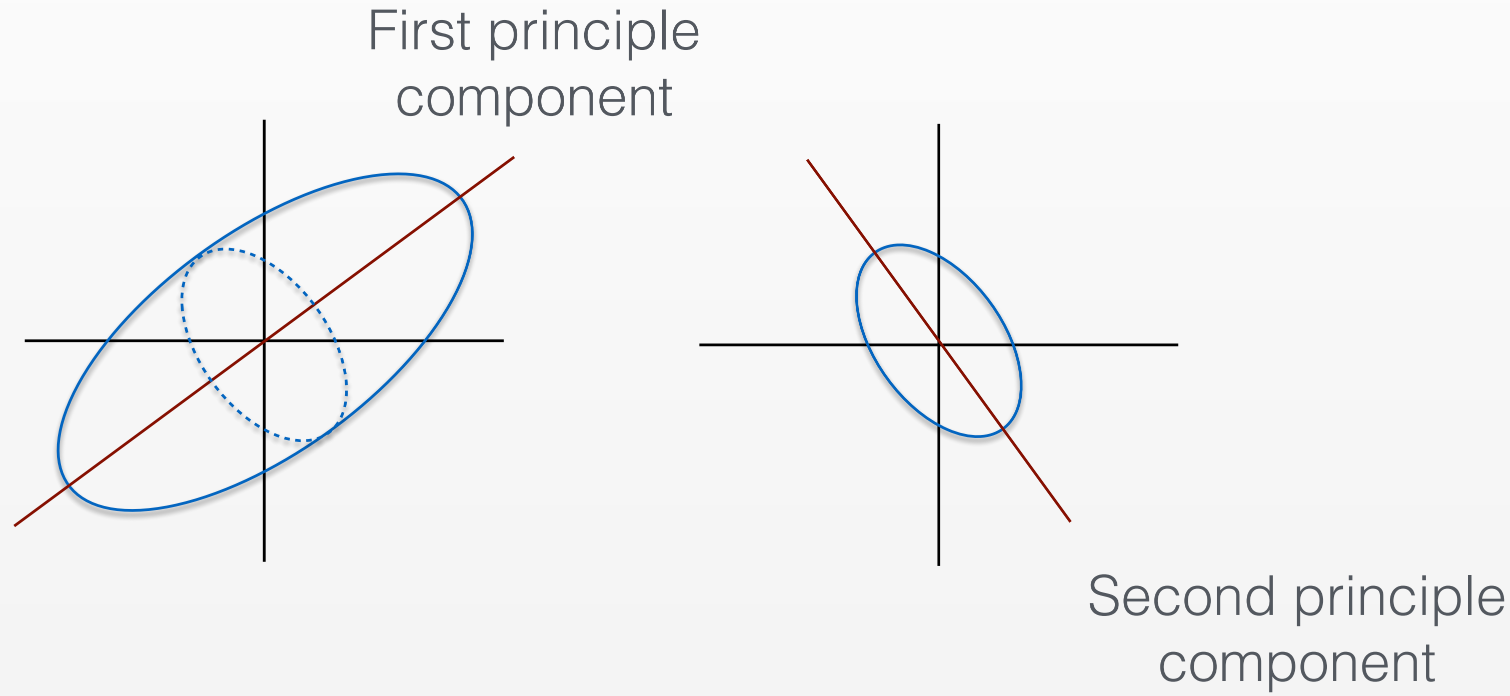
- It follows that
  - $\text{Var}_T(x) = \text{Var}_T(\bar{z}) = \sum_{i=1}^{d} \lambda_i$

- The total variance can be expressed as sum of the eigenvalues of the covariance matrix.

Principal Component Analysis



The variance is maximized by the direction capturing the maximum correlation



Principal Component Analysis

Idea: Take **top-*k*** eigenvectors to maximize variance

1) Sort the eigenvalues in descending order

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$$

2) Sort the corresponding eigenvectors accordingly.
3) Construct a projection matrix with the top-k eigenvectors.

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

- The top eigen-vector captures the maximum variance that can be captured by a single dimension.
- The second eigen-vector captures the second largest variance that can be captured by a single dimension under the constraint that it is uncorrelated to the first dimension

Principal Component Analysis

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Eigenvectors of Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X} \mathbf{X}^\top$$

$$\mathbf{C} \mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Truncated decomposition

$$\boldsymbol{C} \simeq \boldsymbol{U} \boldsymbol{\Lambda}^{(k)} \boldsymbol{U}^\top$$

$$\boldsymbol{\Lambda}^{(k)} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ldots & \\ & & & \lambda_k \end{pmatrix}$$

Principal Component Analysis

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

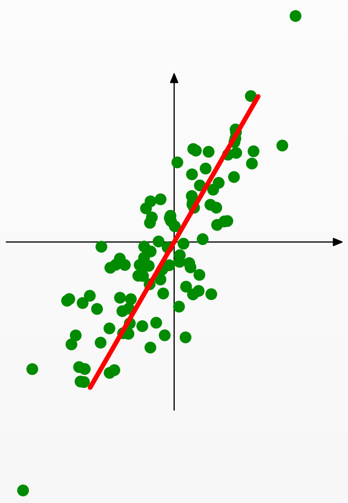
Projection / Encoding

$$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$$

Reconstruction / Decoding

$$\tilde{\mathbf{x}} = \mathbf{U}\mathbf{z}$$

Finding one principal component



Objective: maximize variance of projected data

Input data:

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \ldots & \mathbf{x}_n \\ | & & | \end{pmatrix}$$

PCA: Complexity

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

$$\mathbf{C} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^\top = \frac{1}{n} \mathbf{X}\mathbf{X}^\top$$

Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

$$\mathbf{C}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$

Using eigen-value decomposition

- Computation of covariance **C**: $O(n\, d^2)$
- Eigen-value decomposition: $O(d^3)$
- Total complexity: $O(n\, d^2 + d^3)$

PCA: Complexity

Data

$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots\cdots & \mathbf{x}_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times n}$$

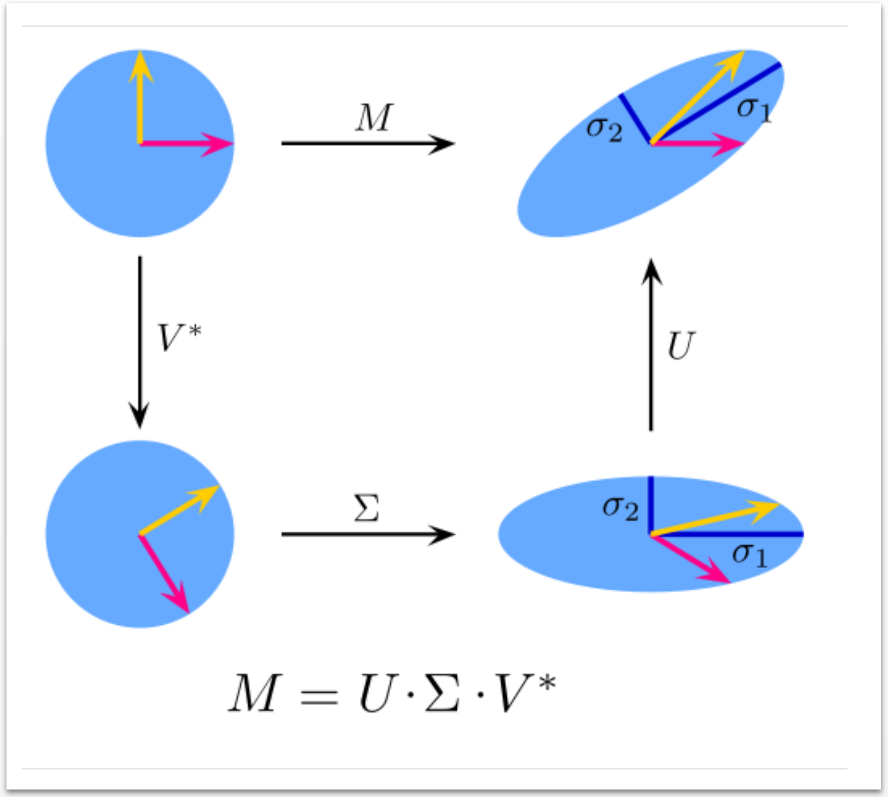
Truncated Basis

$$\mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdot\cdot & \mathbf{u}_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d \times k}$$

Using singular-value decomposition

- Full decomposition: $O(\min\{nd^2, n^2d\})$
- Rank-k decomposition: $O(k\ d\ n \log(n))$ (with power method)

Singular Value Decomposition



$$\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}^{\top}_{n\times n}$$

Idea: Decompose the **d x n** matrix *X* into

1. A n x n basis *V* (unitary matrix)
2. A d x n matrix ***Σ*** (diagonal projection)
3. A d x d basis *U* (unitary matrix)

Relationship Between SVD and PCA

The eigen-vectors of $\frac{1}{n}XX^T$ can be obtained as the left singular vectors of X

PCA (all *d* components)

$$\frac{1}{n}\boldsymbol{X}\boldsymbol{X}^\top = \underbrace{\boldsymbol{U}}_{d\times d}\underbrace{\boldsymbol{\Lambda}}_{d\times d}\underbrace{\boldsymbol{U}^\top}_{d\times d}$$

SVD (all *d* components)

$$\boldsymbol{X} = \underbrace{\boldsymbol{U}}_{d\times d}\,\underbrace{\boldsymbol{\Sigma}}_{d\times n}\,\underbrace{\boldsymbol{V}^\top}_{n\times n}$$

$$\begin{aligned}\frac{1}{n}\boldsymbol{X}\boldsymbol{X}^\top &= \frac{1}{n}\boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \\ &= \frac{1}{n}\boldsymbol{U}\,\boldsymbol{\Sigma}\,I\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \\ &= \frac{1}{n}\boldsymbol{U}\,\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\end{aligned}$$

Relationship Λ and Σ

$$\boxed{\boldsymbol{\Lambda} = \frac{1}{n}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top}$$

Computing Principal Components

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing Principal Components

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing C already takes $O(nd^2)$ time (very expensive)

Computing Principal Components

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing C already takes $O(nd^2)$ time (very expensive)

Method 2: singular value decomposition (SVD)

Find $\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}_{n\times n}^\top$

where $\mathbf{U}^\top\mathbf{U} = I_{d\times d}$, $\mathbf{V}^\top\mathbf{V} = I_{n\times n}$, Σ is diagonal

Computing Principal Components

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing C already takes $O(nd^2)$ time (very expensive)

Method 2: singular value decomposition (SVD)

Find $\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}_{n\times n}^\top$

where $\mathbf{U}^\top\mathbf{U} = I_{d\times d}$, $\mathbf{V}^\top\mathbf{V} = I_{n\times n}$, Σ is diagonal

Computing top k singular vectors takes only $O(ndk)$

Computing Principal Components

Method 1: eigendecomposition

$\mathbf{U}$ are eigenvectors of covariance matrix $C = \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

Computing C already takes $O(nd^2)$ time (very expensive)

Method 2: singular value decomposition (SVD)

Find $\mathbf{X} = \mathbf{U}_{d\times d}\Sigma_{d\times n}\mathbf{V}_{n\times n}^\top$

where $\mathbf{U}^\top\mathbf{U} = I_{d\times d}$, $\mathbf{V}^\top\mathbf{V} = I_{n\times n}$, Σ is diagonal

Computing top k singular vectors takes only $O(ndk)$

Relationship between eigendecomposition and SVD:

Left singular vectors are principal components

Probabilistic Interpretation

Generative Model [Tipping and Bishop, 1999]:

For each data point $i = 1, \ldots, n$:
Draw the latent vector: $\mathbf{z}_i \sim \mathcal{N}(0, I_{k \times k})$
Create the data point: $\mathbf{x}_i \sim \mathcal{N}(\mathbf{U}\mathbf{z}_i, \sigma^2 I_{d \times d})$

PCA finds the $\mathbf{U}$ that maximizes the likelihood of the data

$$\max_{\mathbf{U}} p(\mathbf{X} \mid \mathbf{U})$$

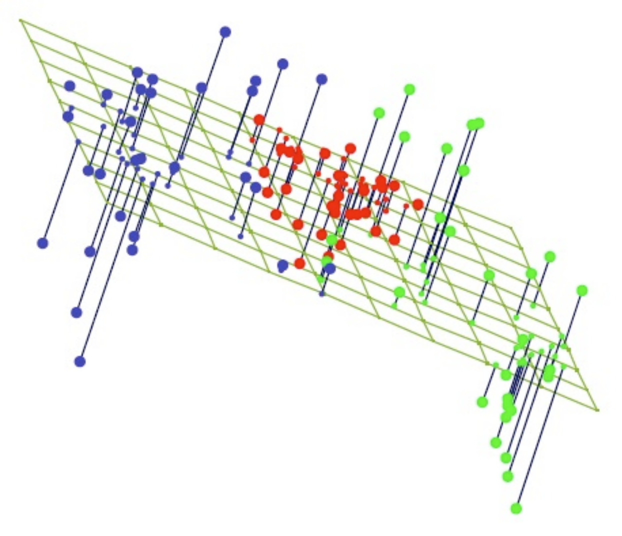
Dimensionality Reduction

Jan-Willem van de Meent

PCA: Applications

Borrowing from:
Percy Liang (Stanford)

Principal Component Analysis



$\mathbf{x} \in \mathbb{R}^{361}$

$\mathbf{z} = \mathbf{U}^\top \mathbf{x}$

$\mathbf{z} \in \mathbb{R}^{10}$

Optimize two equivalent objectives

1. Minimize the reconstruction error

$$\hat{\mathbb{E}}[||\boldsymbol{x} - \boldsymbol{U}\boldsymbol{z}||^2] = \hat{\mathbb{E}}[||(I - \boldsymbol{U}\boldsymbol{U}^\top)\boldsymbol{x}||^2]$$

2. Maximizes the projected variance

$$\hat{\mathbb{E}}[\boldsymbol{z}^\top \boldsymbol{z}] = \hat{\mathbb{E}}[\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}]$$

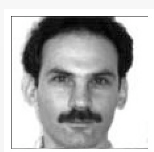
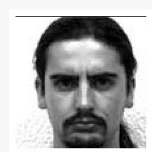
Eigen-faces [Turk & Pentland 1991]

- d = number of pixels
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
- $\mathbf{x}_{ji}$ = intensity of the j-th pixel in image i

Eigen-faces [Turk & Pentland 1991]

- $d =$ number of pixels
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
- $\mathbf{x}_{ji} =$ intensity of the j-th pixel in image i

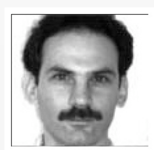
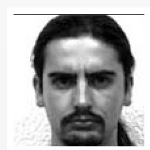
$$\mathbf{X}_{d \times n} \approx \mathbf{U}_{d \times k} \quad \mathbf{Z}_{k \times n}$$

( $\ldots$ ) $\approx$ () $\begin{pmatrix} | & & | \\ \mathbf{z}_1 & \ldots & \mathbf{z}_n \\ | & & | \end{pmatrix}$

Eigen-faces [Turk & Pentland 1991]

- d = number of pixels
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
- $\mathbf{x}_{ji}$ = intensity of the j-th pixel in image i

$$\mathbf{X}_{d\times n} \approxeq \mathbf{U}_{d\times k} \quad \mathbf{Z}_{k\times n}$$

( $\ldots$ ) $\approxeq$ () $\begin{pmatrix} | & & | \\ \mathbf{z}_1 & \ldots & \mathbf{z}_n \\ | & & | \end{pmatrix}$

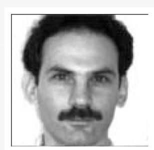
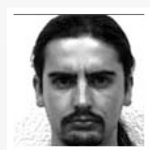
Idea: $\mathbf{z}_i$ more "meaningful" representation of i-th face than $\mathbf{x}_i$

Can use $\mathbf{z}_i$ for nearest-neighbor classification

Eigen-faces [Turk & Pentland 1991]

- d = number of pixels
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
- $\mathbf{x}_{ji}$ = intensity of the j-th pixel in image i

$$\mathbf{X}_{d\times n} \approxeq \mathbf{U}_{d\times k} \; \mathbf{Z}_{k\times n}$$

( $\cdots$ ) $\approxeq$ () $\begin{pmatrix} | & & | \\ \mathbf{z}_1 & \cdots & \mathbf{z}_n \\ | & & | \end{pmatrix}$

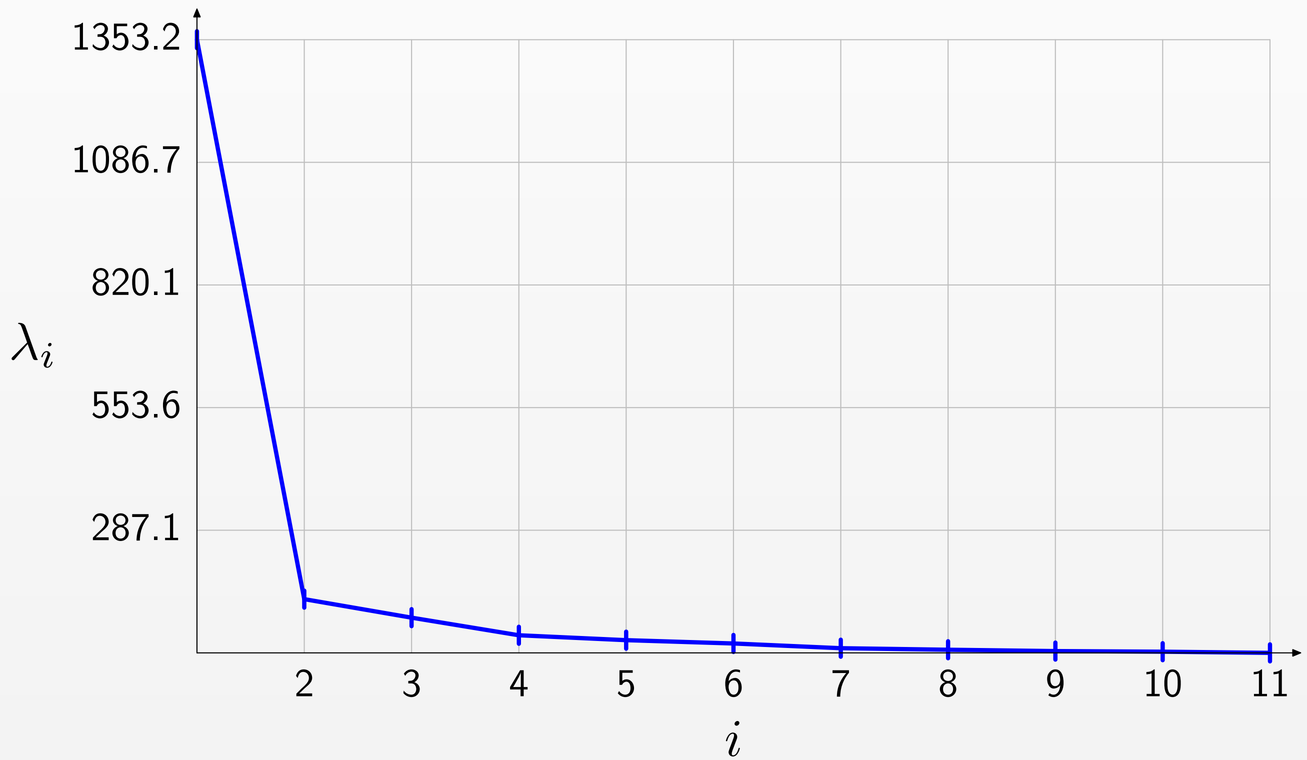
Idea: $\mathbf{z}_i$ more "meaningful" representation of i-th face than $\mathbf{x}_i$

Can use $\mathbf{z}_i$ for nearest-neighbor classification

Much faster: $O(\underbrace{dk}_{\text{projecting}} + \underbrace{nk}_{\text{searching}})$ time instead of $O(dn)$ when $n, d \gg k$

Aside: How many components?

- Magnitude of eigenvalues indicate fraction of variance captured.
- Eigenvalues on a face image dataset:



- Eigenvalues typically drop off sharply, so don't need that many.
- Of course variance isn't everything...

Latent Semantic Analysis [Deerwater 1990]

- $d =$ number of words in the vocabulary
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of word counts

Latent Semantic Analysis [Deerwater 1990]

- d = number of words in the vocabulary
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of word counts
- $\mathbf{x}_{ji}$ = frequency of word j in document i

$$\mathbf{X}_{d\times n} \approx \mathbf{U}_{d\times k} \quad \mathbf{Z}_{k\times n}$$

$$\begin{array}{r} \text{stocks:}\\ \text{chairman:}\\ \text{the:}\\ \cdots\\ \text{wins:}\\ \text{game:}\end{array}\begin{pmatrix} 2 & \cdots & 0\\ 4 & \cdots & 1\\ 8 & \cdots & 7\\ \vdots & \cdots & \vdots\\ 0 & \cdots & 2\\ 1 & \cdots & 3\end{pmatrix} \approx \begin{pmatrix} 0.4 & \cdots & -0.001\\ 0.8 & \cdots & 0.03\\ 0.01 & \cdots & 0.04\\ \vdots & \cdots & \vdots\\ 0.002 & \cdots & 2.3\\ 0.003 & \cdots & 1.9\end{pmatrix}\begin{pmatrix} | & & |\\ \mathbf{z}_1 & \ldots & \mathbf{z}_n\\ | & & |\end{pmatrix}$$

Latent Semantic Analysis [Deerwater 1990]

- d = number of words in the vocabulary
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of word counts
- $\mathbf{x}_{ji}$ = frequency of word j in document i

$$\mathbf{X}_{d\times n} \approx \mathbf{U}_{d\times k} \quad \mathbf{Z}_{k\times n}$$

$$\begin{array}{r}\text{stocks:}\\ \text{chairman:}\\ \text{the:}\\ \cdots\\ \text{wins:}\\ \text{game:}\end{array}\begin{pmatrix} 2 & \cdots\cdots & 0 \\ 4 & \cdots\cdots & 1 \\ 8 & \cdots\cdots & 7 \\ \vdots & \cdots\cdots & \vdots \\ 0 & \cdots\cdots & 2 \\ 1 & \cdots\cdots & 3 \end{pmatrix} \approx \begin{pmatrix} 0.4 & \cdot\cdot & -0.001 \\ 0.8 & \cdot\cdot & 0.03 \\ 0.01 & \cdot\cdot & 0.04 \\ \vdots & \cdot\cdot & \vdots \\ 0.002 & \cdot\cdot & 2.3 \\ 0.003 & \cdot\cdot & 1.9 \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{z}_1 & \ldots & \mathbf{z}_n \\ | & & | \end{pmatrix}$$

How to measure similarity between two documents?

$\mathbf{z}_1^\top \mathbf{z}_2$ is probably better than $\mathbf{x}_1^\top \mathbf{x}_2$

Latent Semantic Analysis [Deerwater 1990]

- $d =$ number of words in the vocabulary
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of word counts
- $\mathbf{x}_{ji} =$ frequency of word j in document i

$$\mathbf{X}_{d \times n} \approx \mathbf{U}_{d \times k} \quad \mathbf{Z}_{k \times n}$$

$$\begin{array}{r} \text{stocks:} \\ \text{chairman:} \\ \text{the:} \\ \cdots \\ \text{wins:} \\ \text{game:} \end{array} \begin{pmatrix} 2 & \cdots\cdots\cdots & 0 \\ 4 & \cdots\cdots\cdots & 1 \\ 8 & \cdots\cdots\cdots & 7 \\ \vdots & \cdots\cdots\cdots & \vdots \\ 0 & \cdots\cdots\cdots & 2 \\ 1 & \cdots\cdots\cdots & 3 \end{pmatrix} \approx \begin{pmatrix} 0.4 & \cdot\cdot & -0.001 \\ 0.8 & \cdot\cdot & 0.03 \\ 0.01 & \cdot\cdot & 0.04 \\ \vdots & \cdot\cdot & \vdots \\ 0.002 & \cdot\cdot & 2.3 \\ 0.003 & \cdot\cdot & 1.9 \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{z}_1 & \ldots & \mathbf{z}_n \\ | & & | \end{pmatrix}$$

How to measure similarity between two documents?

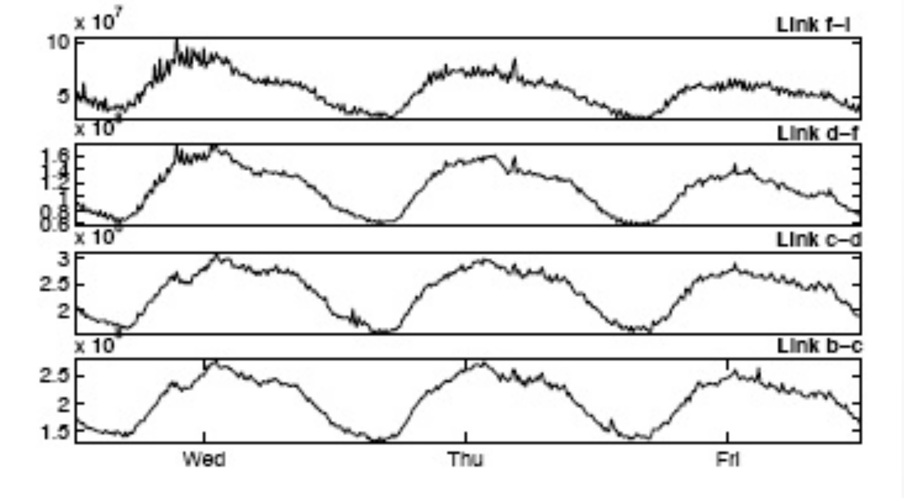
$$\mathbf{z}_1^\top \mathbf{z}_2 \text{ is probably better than } \mathbf{x}_1^\top \mathbf{x}_2$$

Applications: information retrieval

Note: no computational savings; original $\mathbf{x}$ is already sparse

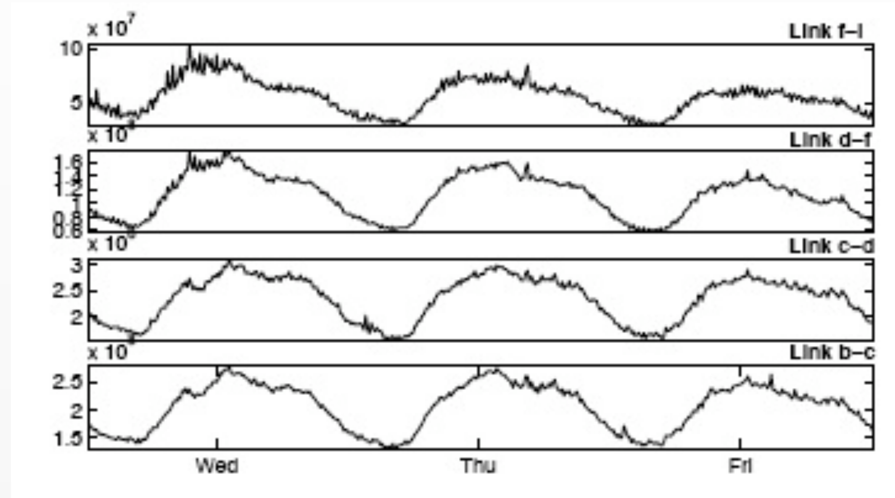
Network anomaly detection [Lakhina 2005]

$\mathbf{x}_{ji}$ = amount of traffic on link j in the network during each time interval i

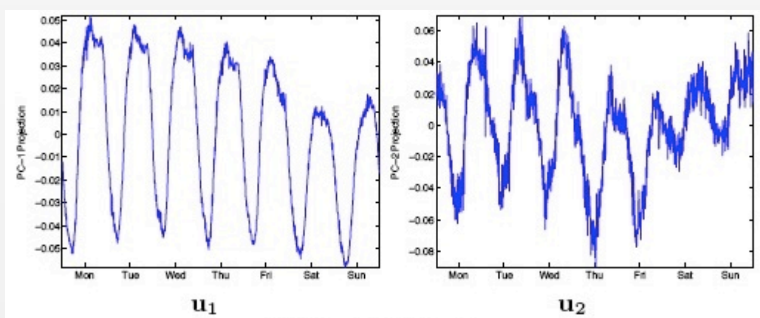


Network anomaly detection [Lakhina 2005]

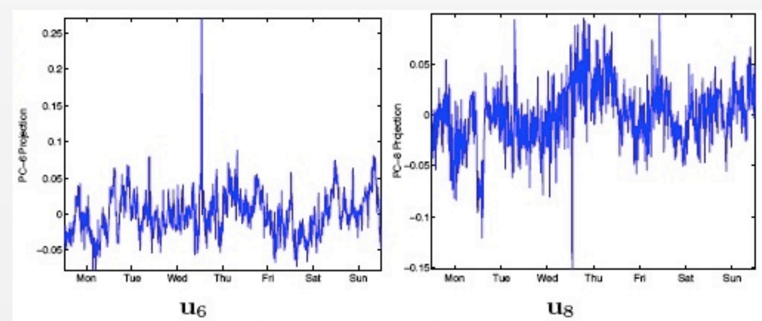
$\mathbf{x}_{ji}$ = amount of traffic on link j in the network during each time interval i



Model assumption: total traffic is sum of flows along a few "paths"
Apply PCA: each principal component intuitively represents a "path"
Anomaly when traffic deviates from first few principal components



Normal



Anomalous

Unsupervised POS tagging [Schütze 1995]

Part-of-speech (POS) tagging task:

Input:	I	like	reducing	the	dimensionality	of	data	.
Output:	NOUN	VERB	VERB(-ING)	DET	NOUN	PREP	NOUN	.

Unsupervised POS tagging [Schütze 1995]

Part-of-speech (POS) tagging task:

Input:	I	like	reducing	the	dimensionality	of	data	.
Output:	NOUN	VERB	VERB(-ING)	DET	NOUN	PREP	NOUN	.

Each $\mathbf{x}_i$ is (the context distribution of) a word.

$\mathbf{x}_{ji}$ is number of times word i appeared in context j

Unsupervised POS tagging [Schütze 1995]

Part-of-speech (POS) tagging task:

Input:	I	like	reducing	the	dimensionality	of	data	.
Output:	NOUN	VERB	VERB(-ING)	DET	NOUN	PREP	NOUN	.

Each $\mathbf{x}_i$ is (the context distribution of) a word.

$\mathbf{x}_{ji}$ is number of times word i appeared in context j

Key idea: words appearing in similar contexts
tend to have the same POS tags;
so cluster using the contexts of each word type

Problem: contexts are too sparse

Solution: run PCA first,
then cluster using new representation

PCA Summary

- Intuition: capture variance of data or minimize reconstruction error
- Algorithm: find eigendecomposition of covariance matrix or SVD
- Impact: reduce storage (from $O(nd)$ to $O(nk)$), reduce time complexity
- Advantages: simple, fast
- Applications: eigen-faces, eigen-documents, network anomaly detection, etc.

Dimensionality Reduction

Shantanu Jain

Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- Image retrieval: for each image, have the following:
  - $\mathbf{x}$: Pixels (or other visual features)
  - $\mathbf{y}$: Text around the image

Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- Image retrieval: for each image, have the following:
  - $\mathbf{x}$: Pixels (or other visual features)
  - $\mathbf{y}$: Text around the image
- Time series:
  - $\mathbf{x}$: Signal at time t
  - $\mathbf{y}$: Signal at time $t+1$

Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- Image retrieval: for each image, have the following:
  - $\mathbf{x}$: Pixels (or other visual features)
  - $\mathbf{y}$: Text around the image
- Time series:
  - $\mathbf{x}$: Signal at time t
  - $\mathbf{y}$: Signal at time $t+1$
- Two-view learning: divide features into two sets
  - $\mathbf{x}$: Features of a word/object, etc.
  - $\mathbf{y}$: Features of the context in which it appears

Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- Image retrieval: for each image, have the following:
  - $\mathbf{x}$: Pixels (or other visual features)
  - $\mathbf{y}$: Text around the image
- Time series:
  - $\mathbf{x}$: Signal at time t
  - $\mathbf{y}$: Signal at time $t+1$
- Two-view learning: divide features into two sets
  - $\mathbf{x}$: Features of a word/object, etc.
  - $\mathbf{y}$: Features of the context in which it appears

Goal: reduce the dimensionality of the two views jointly

CCA Example

Setup:

Input data: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ (matrices $\mathbf{X}$, $\mathbf{Y}$)

Goal: find pair of projections $(\mathbf{u}, \mathbf{v})$

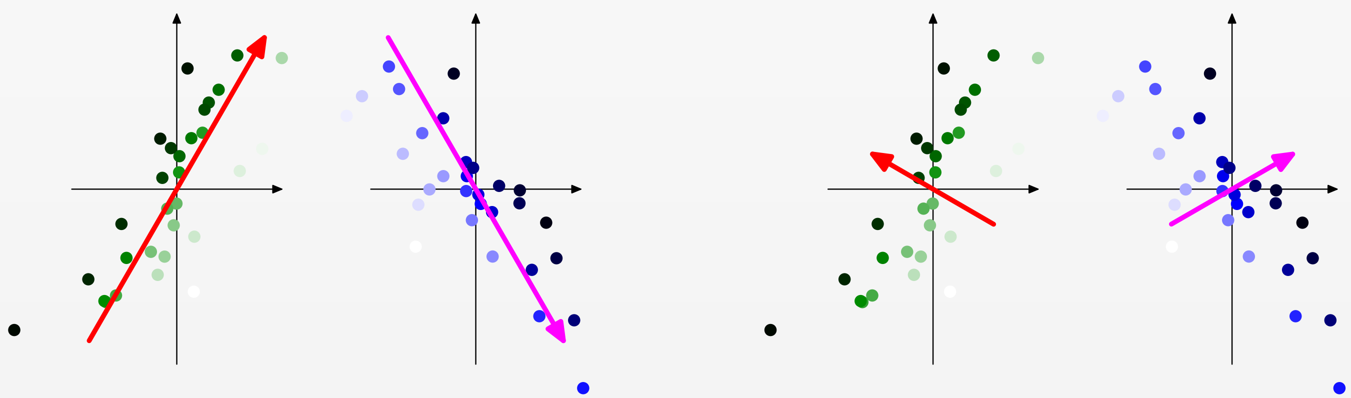
CCA Example

Setup:

Input data: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ (matrices $\mathbf{X}$, $\mathbf{Y}$)

Goal: find pair of projections $(\mathbf{u}, \mathbf{v})$

Dimensionality reduction solutions:



Independent Joint

$\mathbf{x}$ and $\mathbf{y}$ are paired by brightness

CCA Definition

Variance: $\widehat{\text{Var}}(z_x) = \widehat{\text{Var}}(u^T x) = \frac{1}{n} u^T X X^T u = u^T C_x u$

Variance: $\widehat{\text{Var}}(z_y) = \widehat{\text{Var}}(u^T x) = \frac{1}{n} v^T Y Y^T y = v^T C_y v$

Covariance: $\widehat{Cov}(z_x, z_y) = \widehat{Cov}(u^T x, v^T y) = \frac{1}{n} u^T X Y^T v = u^T C_{xy} v$

Covariance amongst the features of x view

$$C_x = \frac{1}{n} X X^T$$

Covariance amongst the features of y view

$$C_y = \frac{1}{n} Y Y^T$$

Covariance between the features of x and y view

$$C_{xy} = \frac{1}{n} X Y^T$$

Objective: Maximize correlation between the projected views.

$$\widehat{Cor}(z_x, z_y) = \frac{u^T C_{xy} v}{\sqrt{u^T C_x u}\sqrt{v^T C_y v}}$$

Properties

- Captures how the projected views are related and not how they vary
- Invariant to rotation and scaling of the data.

CCA Solution

$$\rho_1 = \max_{u,v} \frac{u^T C_{xy} v}{\sqrt{u^T C_x u}\sqrt{v^T C_y v}}$$

$$= \max_{\tilde{u},\tilde{v}} \frac{\tilde{u}^T C_x^{-\frac{1}{2}} C_{xy} C_y^{-\frac{1}{2}} \tilde{v}}{\sqrt{\tilde{u}^T \tilde{u}}\sqrt{\tilde{v}^T \tilde{v}}}$$

$$= \max_{\tilde{u},\tilde{v}} \frac{\tilde{u}^T M \tilde{v}}{\sqrt{\tilde{u}^T \tilde{u}}\sqrt{\tilde{v}^T \tilde{v}}}$$

$$\tilde{u} = C_x^{\frac{1}{2}} u$$

$$\tilde{v} = C_y^{\frac{1}{2}} v$$

$$M = C_x^{-\frac{1}{2}} C_{xy} C_y^{-\frac{1}{2}}$$

Covariance amongst the features of x view

$$C_x = \frac{1}{n} X X^T$$

Covariance amongst the features of y view

$$C_y = \frac{1}{n} Y Y^T$$

Covariance between the features of x and y view

$$C_{xy} = \frac{1}{n} X Y^T$$

CCA Solution

$$\rho_1^2 = \max_{\tilde{u},\tilde{v}} \frac{(\tilde{u}^T M \tilde{v})^2}{\tilde{u}^T \tilde{u} \times \tilde{v}^T \tilde{v}}$$

$$= \max_{\tilde{v}} \frac{(\tilde{v}^T M^T M \tilde{v})^2}{\tilde{v}^T M^T M \tilde{v} \times \tilde{v}^T \tilde{v}}$$

For a given $\tilde{v}$, the correlation w.r.t. $\tilde{u}$ is maximized by $\tilde{u} = cM\tilde{v}$

$$= \max_{\tilde{v}} \frac{\tilde{v}^T M^T M \tilde{v}}{\tilde{v}^T \tilde{v}}$$

$$= \lambda_1$$ First eigen-value of $M^T M$

$$C_x = \frac{1}{n} X X^T$$

$$C_y = \frac{1}{n} Y Y^T$$

$$C_{xy} = \frac{1}{n} X Y^T$$

$$\tilde{u} = C_x^{\frac{1}{2}} u$$

$$\tilde{v} = C_y^{\frac{1}{2}} v$$

$$M = C_x^{-\frac{1}{2}} C_{xy} C_y^{-\frac{1}{2}}$$

Same maximization equation as we saw in PCA

The correlation is maximized by the first eigen-vector of $M^T M$ as $\tilde{v}$ and the correlation is given by the square-root associated eigen-value.

CCA Solution

Solution for the first canonical components.

- $u = C_x^{-\frac{1}{2}}\tilde{u}$, $v = C_y^{-\frac{1}{2}}\tilde{v}$, where $\tilde{u}$ and $\tilde{v}$ are the first eigen-vectors of MM^T and M^TM, respectively.
- Correlation between the first canonical components is $\sqrt{\lambda_1}$

If $\tilde{v}$ is an eigen-vector of M^TM and $\tilde{u} = cM\tilde{v}$

$$\begin{aligned} MM^T\tilde{u} &= cMM^TM\tilde{v} \\ &= cM(M^TM\tilde{v}) \\ &= c\lambda_1 M\tilde{v} \\ &= \lambda_1\tilde{u} \end{aligned}$$

The correlation is maximized by the first eigen-vectors of M^TM and MM^T as $\tilde{u}$ and $\tilde{v}$, respectively.

$$C_x = \frac{1}{n}XX^T$$

$$C_y = \frac{1}{n}YY^T$$

$$C_{xy} = \frac{1}{n}XY^T$$

$$\tilde{u} = C_x^{\frac{1}{2}}u$$

$$\tilde{v} = C_y^{\frac{1}{2}}v$$

$$M = C_x^{-\frac{1}{2}}C_{xy}C_y^{-\frac{1}{2}}$$

CCA Solution

Solution for the first k canonical components.

- $U = C_x^{-\frac{1}{2}}\tilde{U}$, $V = C_y^{-\frac{1}{2}}\tilde{V}$, where $\tilde{U}$ and $\tilde{V}$ contains the top-k eigen-vectors of MM^T and M^TM, respectively.
- Correlation between the top-$k$canonical components is given by the top-k eigen-values of M^TM: $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_k}$.

$$C_x = \frac{1}{n}XX^T$$

$$C_y = \frac{1}{n}YY^T$$

$$C_{xy} = \frac{1}{n}XY^T$$

$$\tilde{u} = C_x^{\frac{1}{2}}u$$

$$\tilde{v} = C_y^{\frac{1}{2}}v$$

$$M = C_x^{-\frac{1}{2}}C_{xy}C_y^{-\frac{1}{2}}$$