Data Mining Techniques

CS 6220 - Section 3 - Fall 2016

Lecture 12

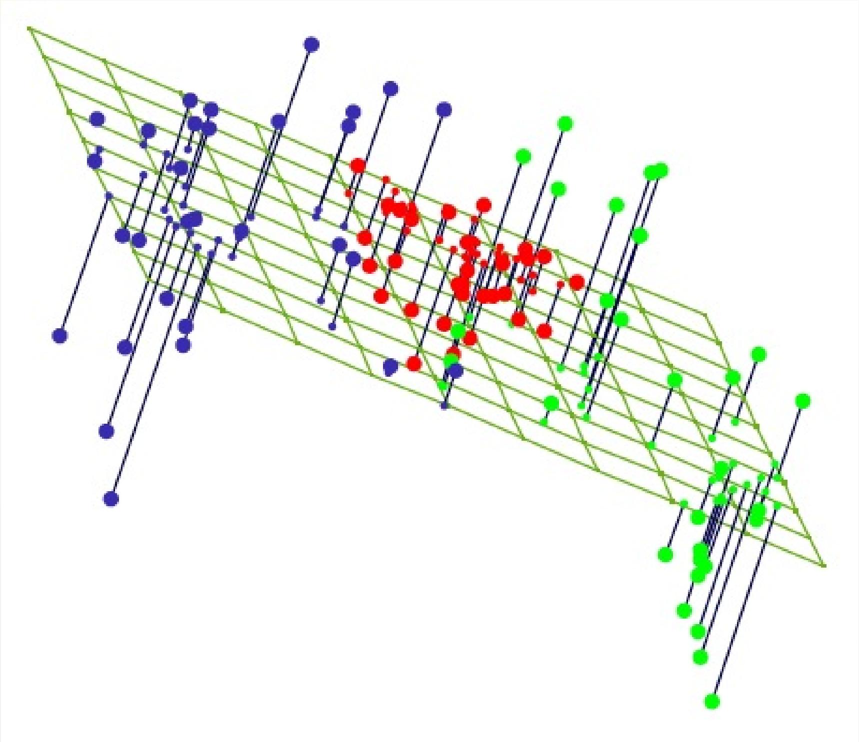
Jan-Willem van de Meent
(credit: Yijun Zhao, Percy Liang)

DIMENSIONALITY REDUCTION

Borrowing from:
Percy Liang
(Stanford)

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 $\mathbf{u}_1, \ldots, \mathbf{u}_k$

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

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 $\mathbf{u}_1, \ldots, \mathbf{u}_k$

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

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

Problem Setup

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

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

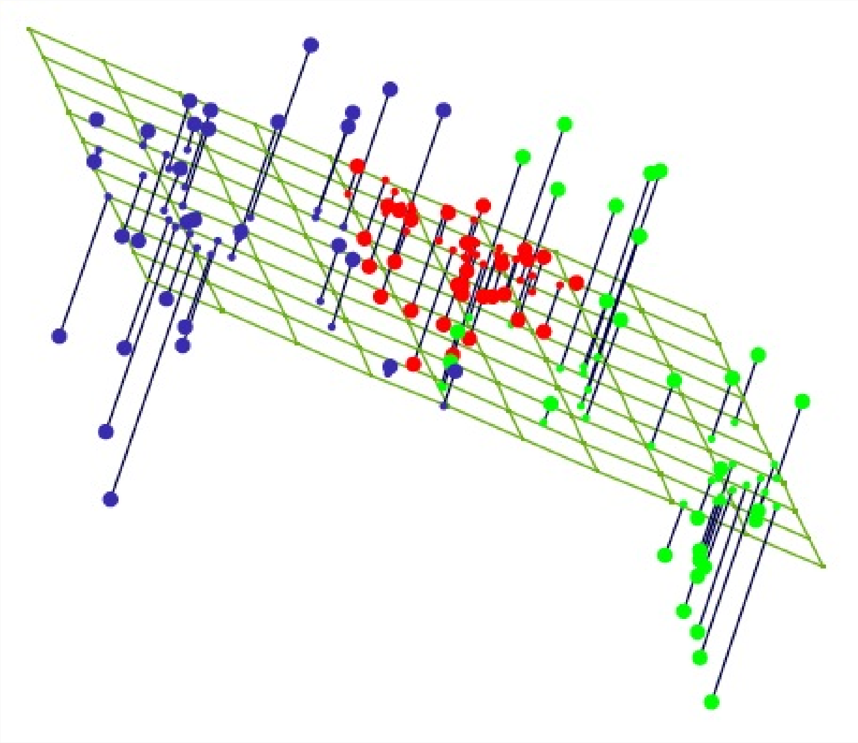
Want to reduce dimensionality from d to k

- 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 $\mathbf{U}$?

Principal Component Analysis



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

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

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

How do we choose $\mathbf{U}$?

Two Objectives

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

PCA Objective 1: Reconstruction Error

$\mathbf{U}$ serves two functions:

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

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

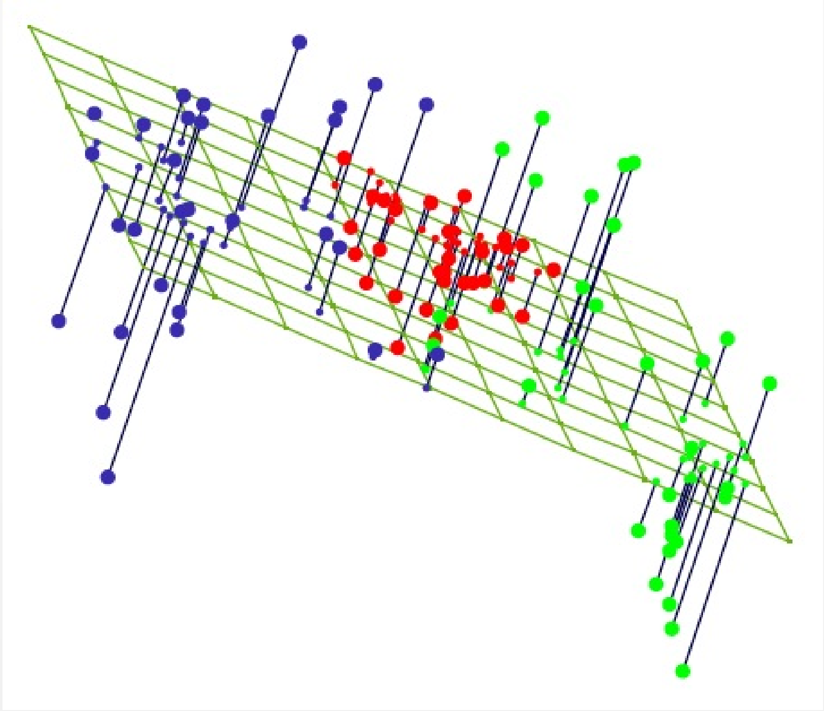
PCA Objective 1: Reconstruction Error

$\mathbf{U}$ 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



$$\min_{\mathbf{U} \in \mathbb{R}^{d \times k}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|^2$$

PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\widehat{\text{var}}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\widehat{\text{var}}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

Assume data is centered: $\hat{\mathbb{E}}[\mathbf{x}] = 0$

PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\widehat{\text{var}}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

Assume data is centered: $\hat{\mathbb{E}}[\mathbf{x}] = 0$ (what's $\hat{\mathbb{E}}[\mathbf{U}^\top \mathbf{x}]$?)

PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\widehat{\text{var}}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

Assume data is centered: $\hat{\mathbb{E}}[\mathbf{x}] = 0$ (what's $\hat{\mathbb{E}}[\mathbf{U}^\top \mathbf{x}]$?)

Objective: maximize variance of projected data

$$\max_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = I} \hat{\mathbb{E}}[\|\mathbf{U}^\top \mathbf{x}\|^2]$$

PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\widehat{\text{var}}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

Assume data is centered: $\hat{\mathbb{E}}[\mathbf{x}] = 0$ (what's $\hat{\mathbb{E}}[\mathbf{U}^\top \mathbf{x}]$?)

Objective: maximize variance of projected data

$$\max_{\mathbf{U} \in \mathbb{R}^{d \times k},\ \mathbf{U}^\top \mathbf{U} = I} \hat{\mathbb{E}}[\|\mathbf{U}^\top \mathbf{x}\|^2]$$

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

Equivalence of two objectives

Key intuition:

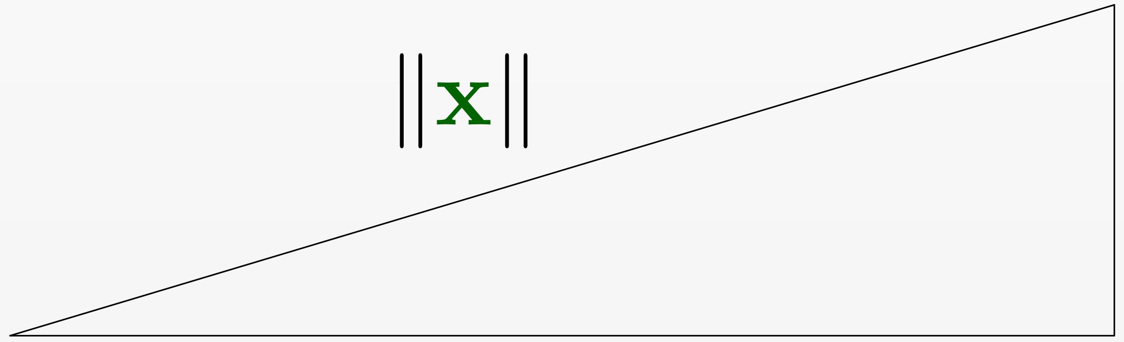
$$\underbrace{\text{variance of data}}_{\text{fixed}} = \underbrace{\text{captured variance}}_{\text{want large}} + \underbrace{\text{reconstruction error}}_{\text{want small}}$$

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



Take expectations; note rotation $\mathbf{U}$ doesn't affect length:

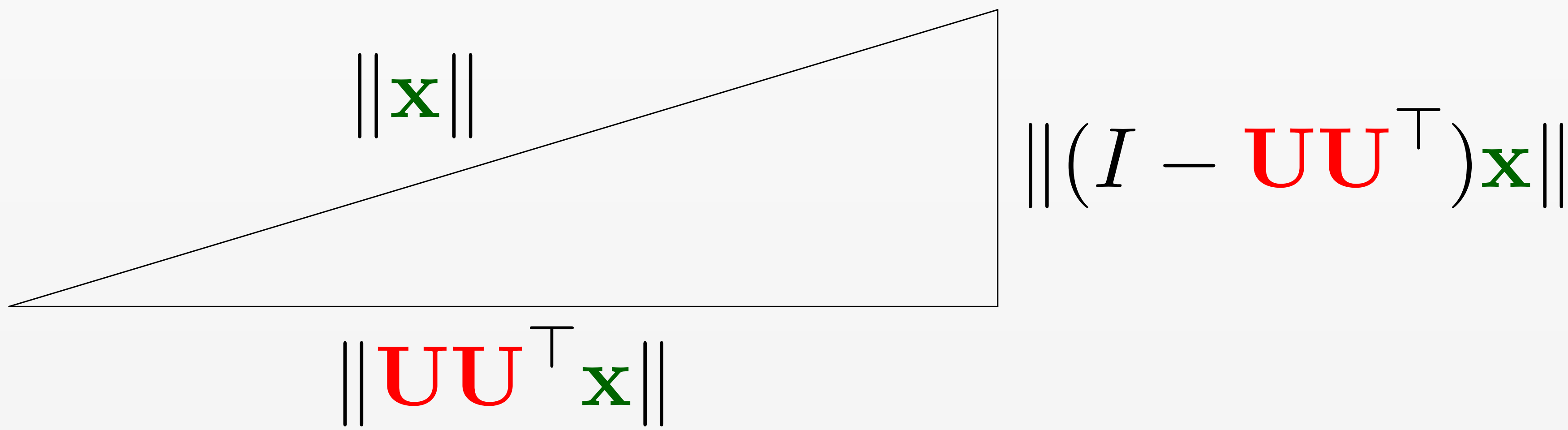
$$\hat{\mathbb{E}}[\|\mathbf{x}\|^2] = \hat{\mathbb{E}}[\|\mathbf{U}^\top \mathbf{x}\|^2] + \hat{\mathbb{E}}[\|\mathbf{x} - \mathbf{U}\mathbf{U}^\top \mathbf{x}\|^2]$$

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

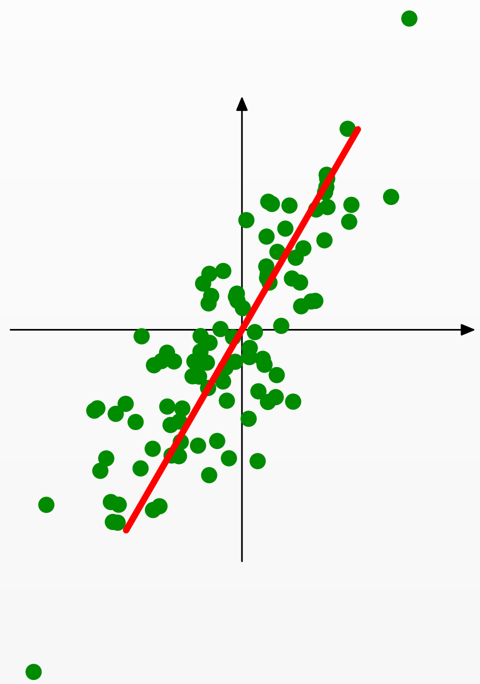


Take expectations; note rotation $\mathbf{U}$ doesn't affect length:

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

Minimize reconstruction error $\leftrightarrow$ Maximize captured variance

Finding one principal component

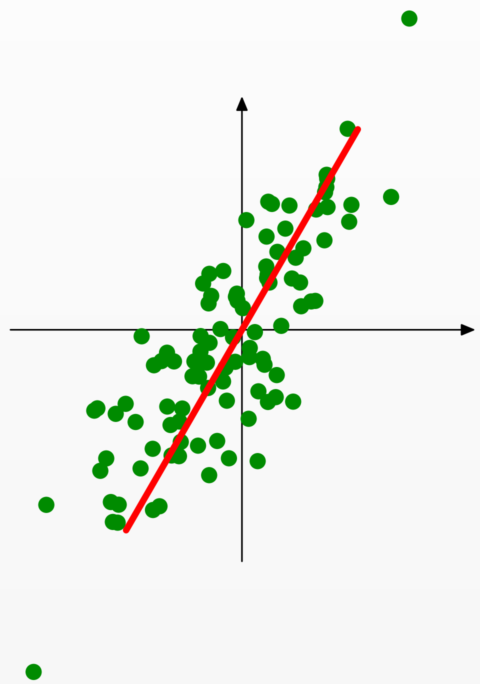


Objective: maximize variance of projected data

Input data:

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

Finding one principal component



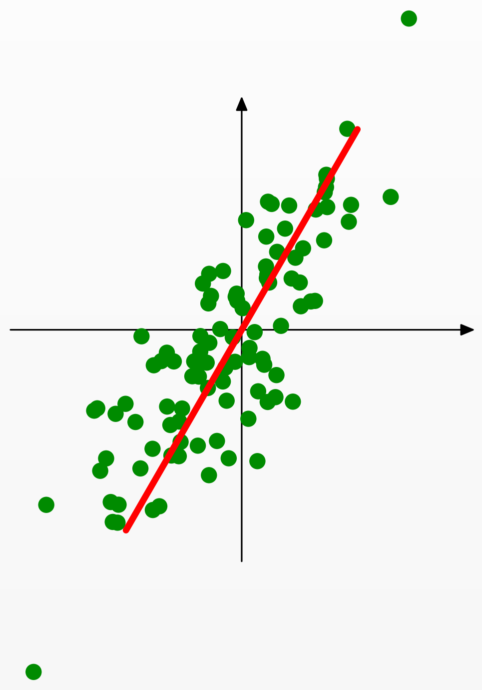
Objective: maximize variance of projected data

$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

Input data:

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

Finding one principal component



Objective: maximize variance of projected data

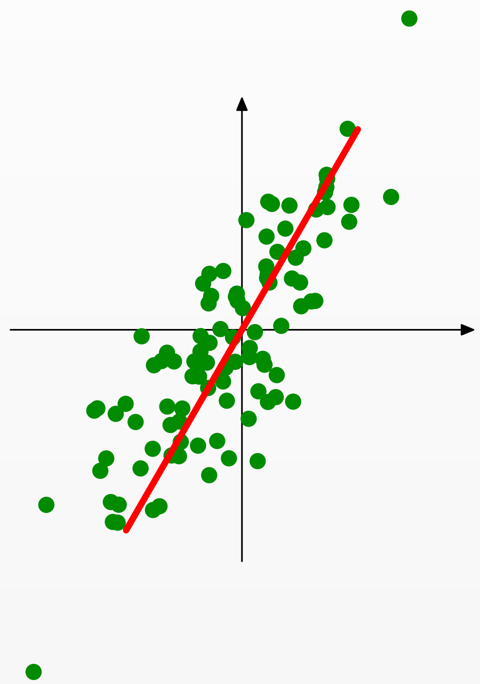
$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2$$

Input data:

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

Finding one principal component



Input data:

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

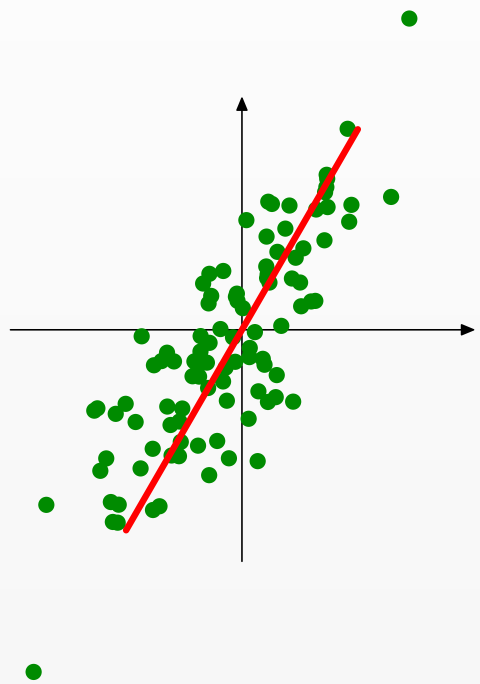
Objective: maximize variance of projected data

$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \|\mathbf{u}^\top \mathbf{X}\|^2$$

Finding one principal component



Input data:

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

Objective: maximize variance of projected data

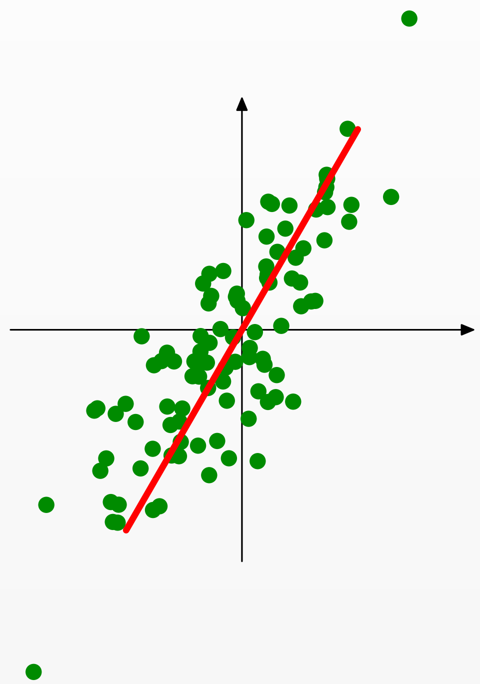
$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \|\mathbf{u}^\top \mathbf{X}\|^2$$

$$= \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \left(\frac{1}{n} \mathbf{X}\mathbf{X}^\top \right) \mathbf{u}$$

Finding one principal component



Input data:

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

Objective: maximize variance of projected data

$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

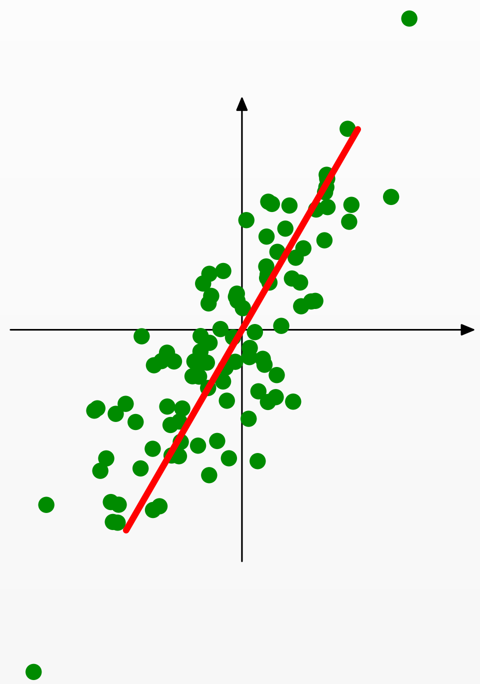
$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \|\mathbf{u}^\top \mathbf{X}\|^2$$

$$= \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \left(\frac{1}{n}\mathbf{X}\mathbf{X}^\top\right) \mathbf{u}$$

$$= \text{largest eigenvalue of } C \stackrel{\text{def}}{=} \frac{1}{n}\mathbf{X}\mathbf{X}^\top$$

Finding one principal component



Objective: maximize variance of projected data

$$= \max_{\|\mathbf{u}\|=1} \hat{\mathbb{E}}[(\mathbf{u}^\top \mathbf{x})^2]$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2$$

$$= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \|\mathbf{u}^\top \mathbf{X}\|^2$$

$$= \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \left(\frac{1}{n} \mathbf{X}\mathbf{X}^\top \right) \mathbf{u}$$

$= \text{largest eigenvalue of } C \stackrel{\text{def}}{=} \frac{1}{n}\mathbf{X}\mathbf{X}^\top$

(C is covariance matrix of data)

Input data:

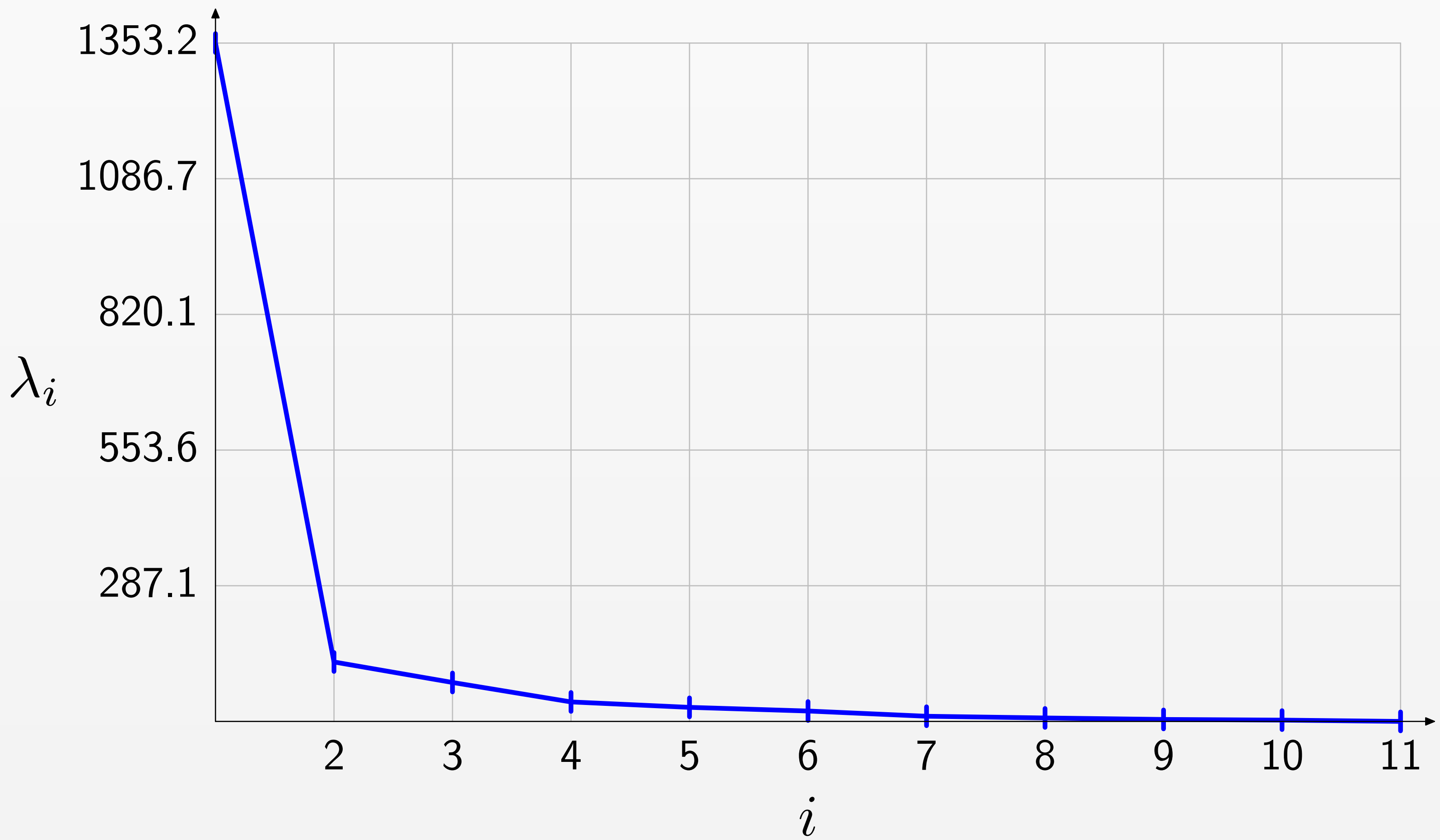
$$\mathbf{X} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{pmatrix}$$

How many components?

- Similar to question of “How many clusters?”
- Magnitude of eigenvalues indicate fraction of variance captured.

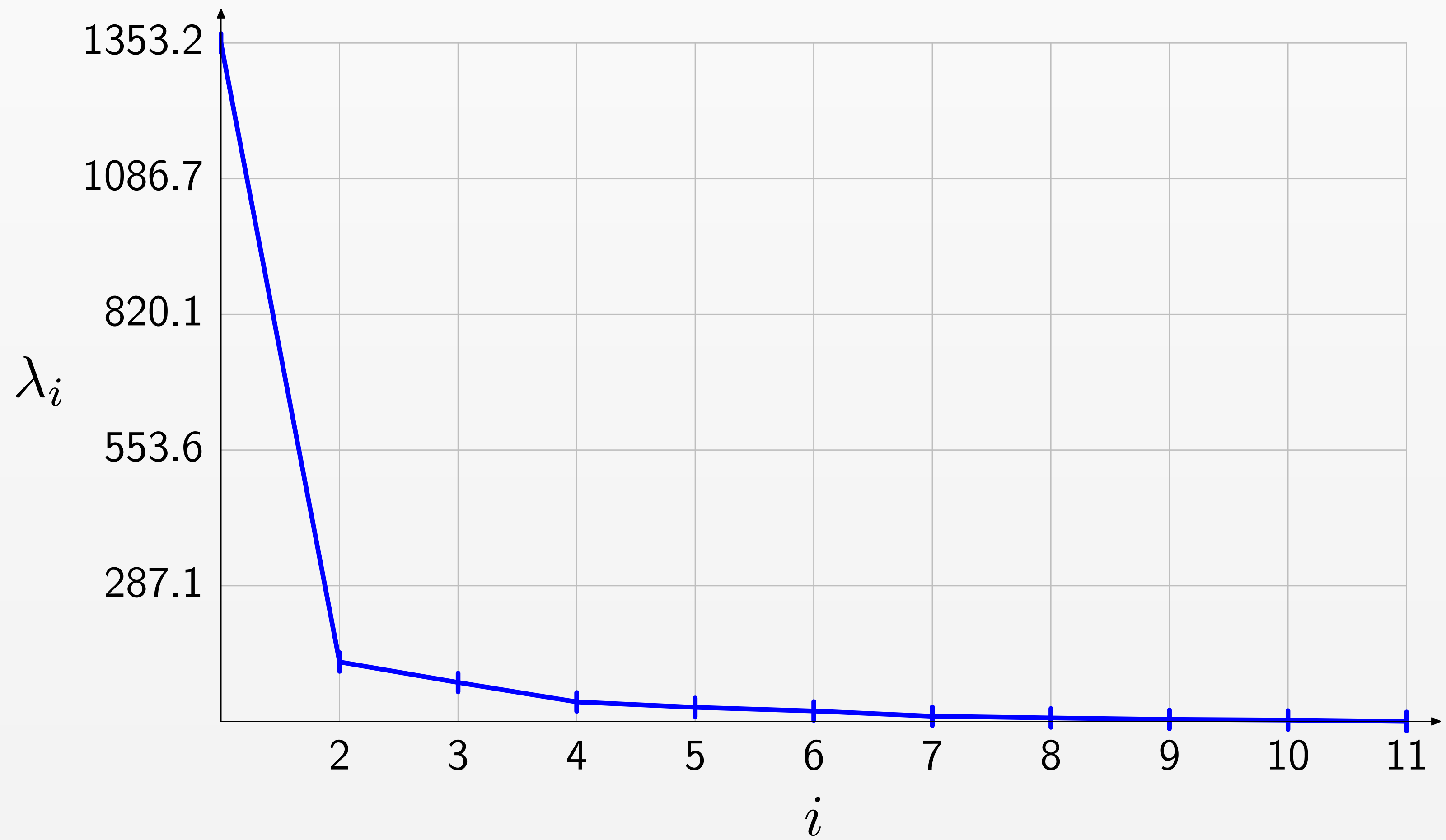
How many components?

- Similar to question of "How many clusters?"
- Magnitude of eigenvalues indicate fraction of variance captured.
- Eigenvalues on a face image dataset:



How many components?

- Similar to question of "How many clusters?"
- 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...

Computing PCA

Method 1: eigendecomposition

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

Computing PCA

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 PCA

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 PCA

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 PCA

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 ($C = \mathbf{U}\Sigma^2\mathbf{U}^\top$)

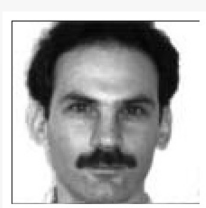
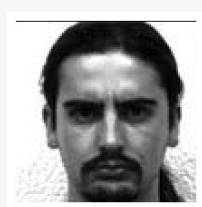
Eigen-faces [Turk & Pentland 1991]

- 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

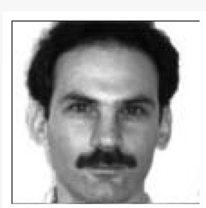
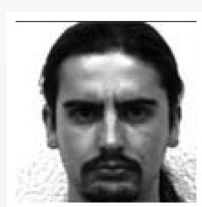
$$\mathbf{X}_{d\times n} \approxeq \mathbf{U}_{d\times k} \; \mathbf{Z}_{k\times n}$$

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

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

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

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

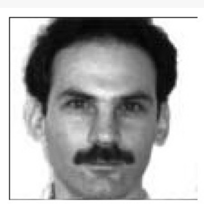
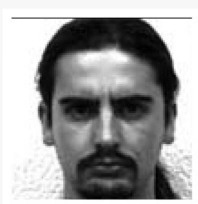
Idea: $\mathbf{z}_i$ more "meaningful" representation of i-th face than $\mathbf{x}_i$

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

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

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

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: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \gg k$

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

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

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: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \gg k$

Why no time savings for linear classifier?

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

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

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

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

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

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

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

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

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

How to measure similarity between two documents?

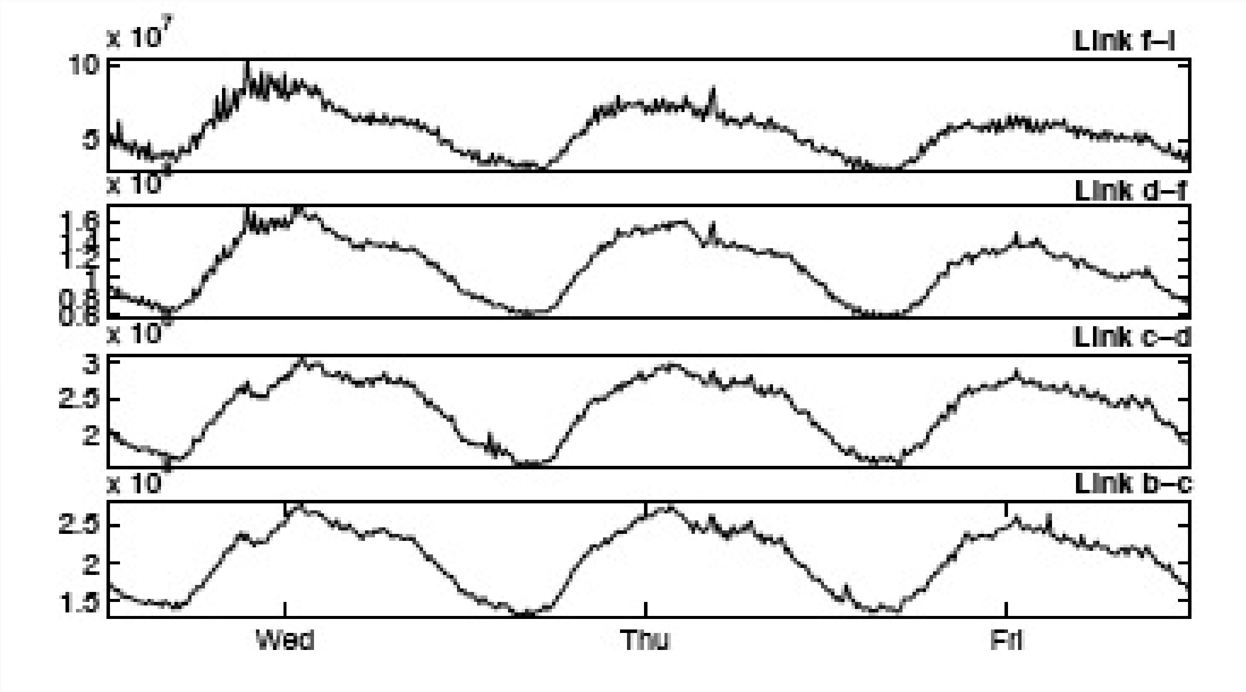
$\mathbf{z}_1^\top \mathbf{z}_2$ is probably better than $\mathbf{x}_1^\top \mathbf{x}_2$

Applications: information retrieval

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

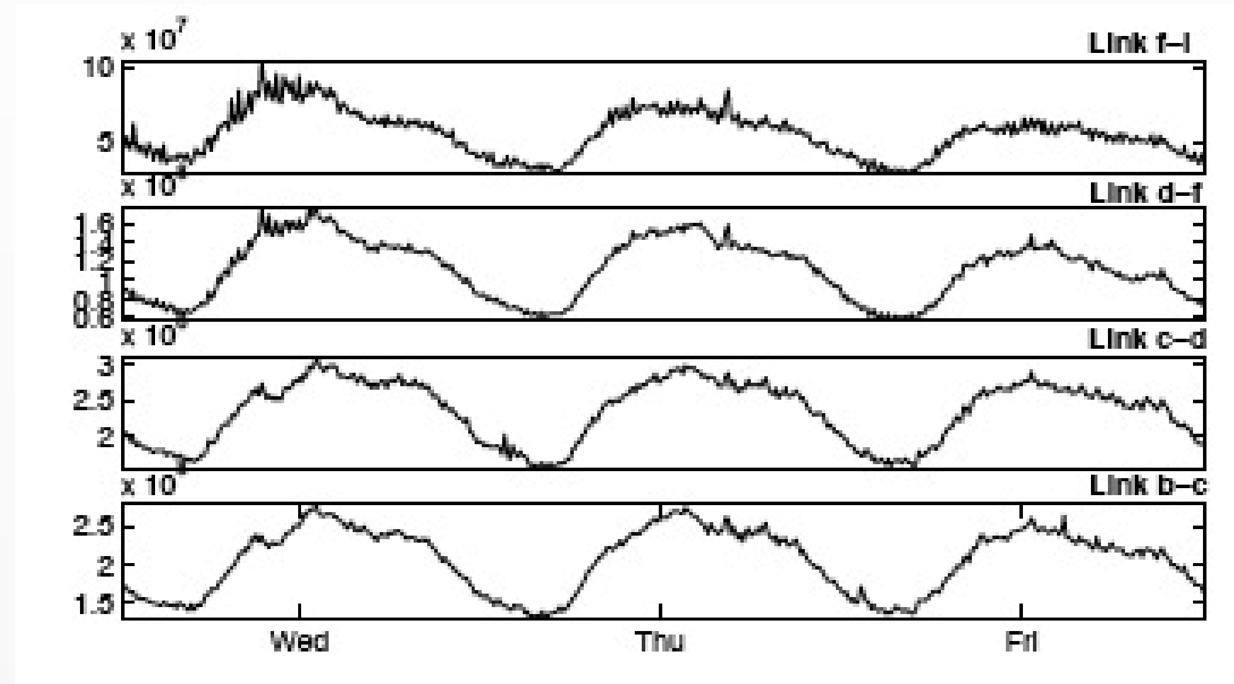
Network anomaly detection [Lakhina 2005]

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

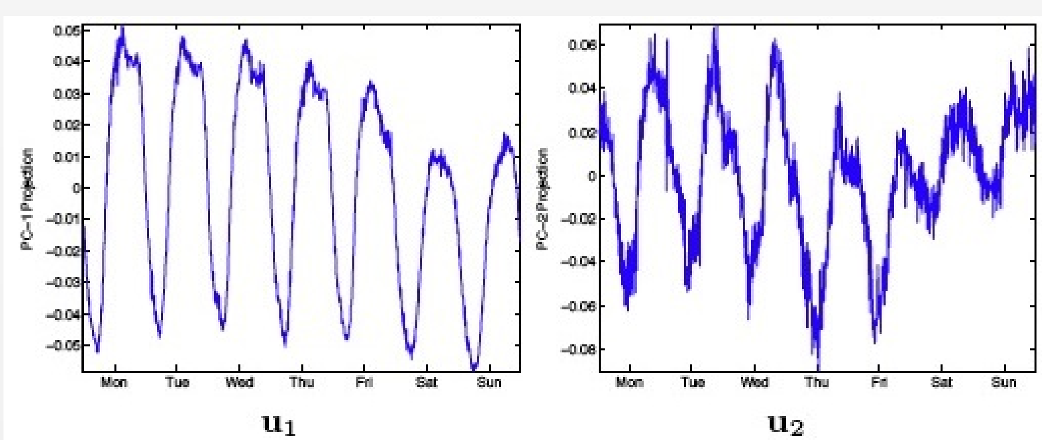


Network anomaly detection [Lakhina 2005]

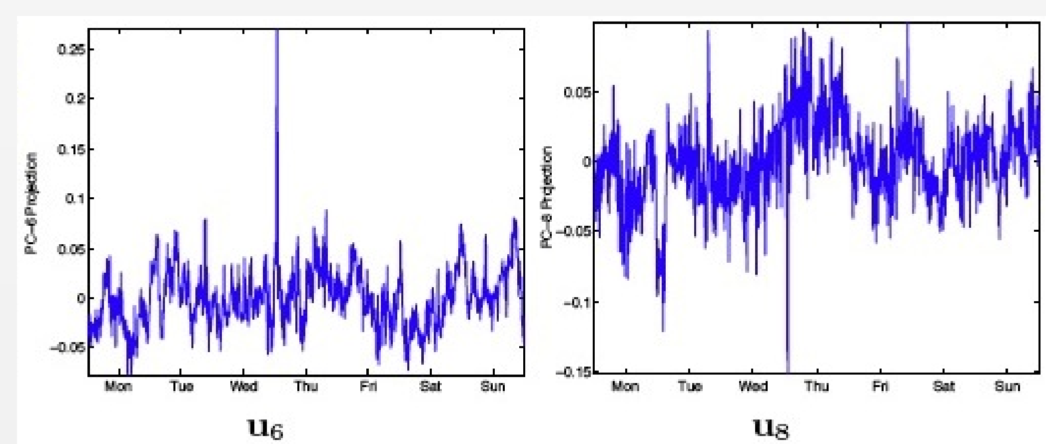
$\mathbf{x}_{ji}$ = amount of traffic on link j in the network during each time interval i



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

Multi-task learning [Ando & Zhang 2005]

- Have n related tasks (classify documents for various users)
- Each task has a linear classifier with weights $\mathbf{x}_i$
- Want to share structure between classifiers

Multi-task learning [Ando & Zhang 2005]

- Have n related tasks (classify documents for various users)
- Each task has a linear classifier with weights $\mathbf{x}_i$
- Want to share structure between classifiers

One step of their procedure:
given n linear classifiers $\mathbf{x}_1, \ldots, \mathbf{x}_n$,
run PCA to identify shared structure:

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

Each principal component is a eigen-classifier

Multi-task learning [Ando & Zhang 2005]

- Have n related tasks (classify documents for various users)
- Each task has a linear classifier with weights $\mathbf{x}_i$
- Want to share structure between classifiers

One step of their procedure:
given n linear classifiers $\mathbf{x}_1, \ldots, \mathbf{x}_n$,
run PCA to identify shared structure:

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

Each principal component is a eigen-classifier

Other step of their procedure:
Retrain classifiers, regularizing towards subspace $\mathbf{U}$

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.

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

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

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: $\mathbf{x}_i \sim \mathcal{N}(\mathbf{U}\mathbf{z}_i, \sigma^2 I_{d \times d})$

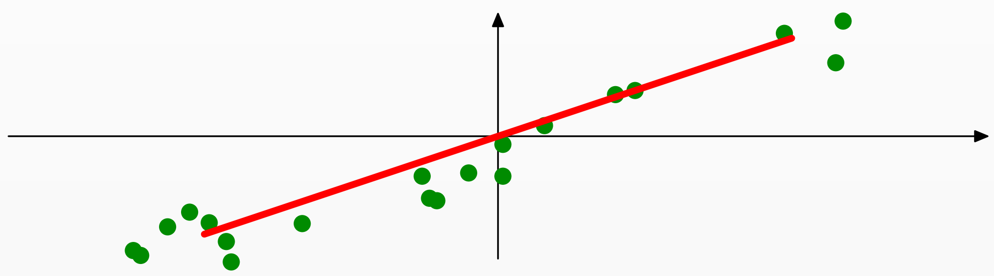
PCA finds the $\mathbf{U}$ that maximizes the likelihood of the data

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

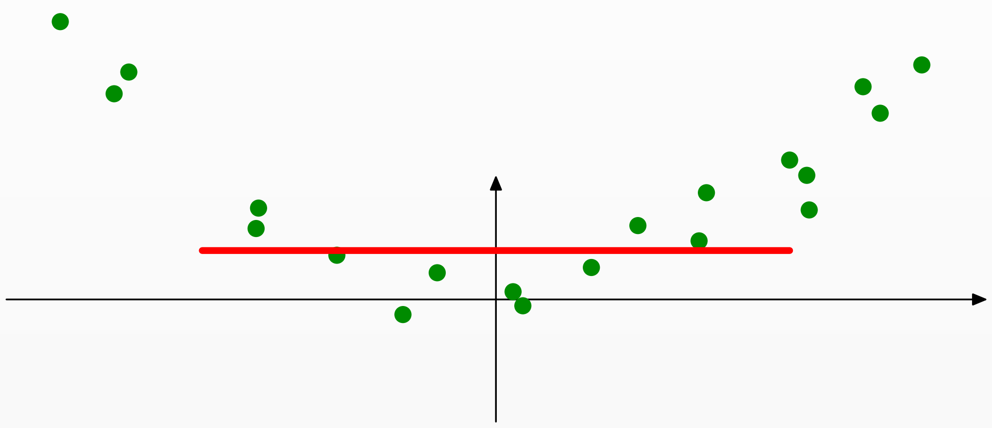
Advantages:

- Handles missing data (important for collaborative filtering)
- Extension to factor analysis: allow non-isotropic noise (replace $\sigma^2 I_{d \times d}$ with arbitrary diagonal matrix)

Limitations of Linearity

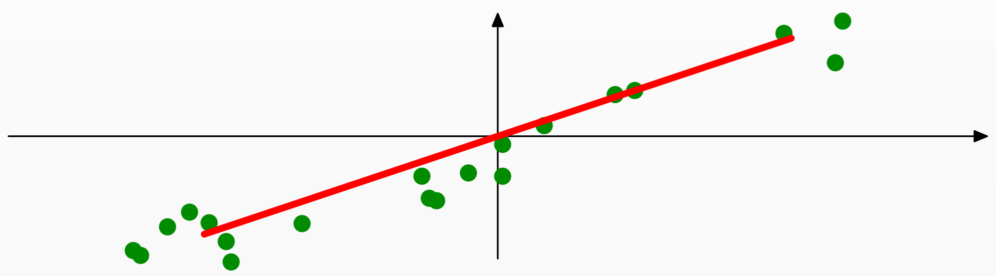


PCA is effective

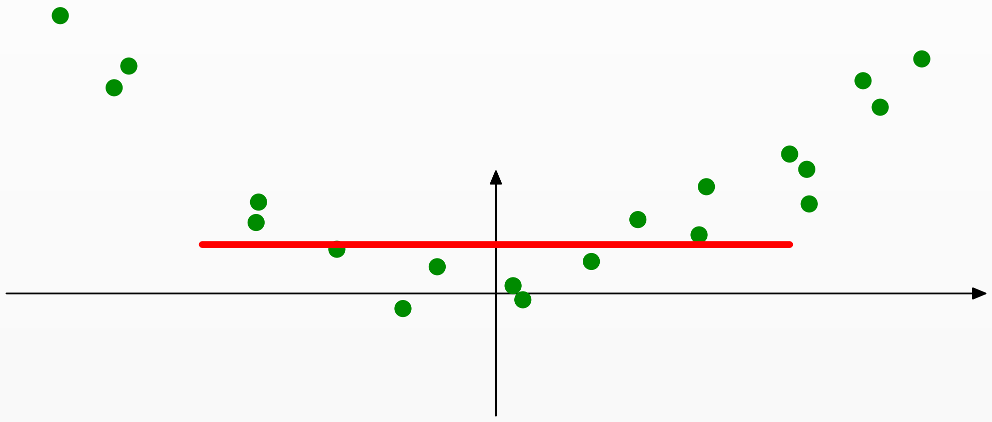


PCA is ineffective

Limitations of Linearity



PCA is effective



PCA is ineffective

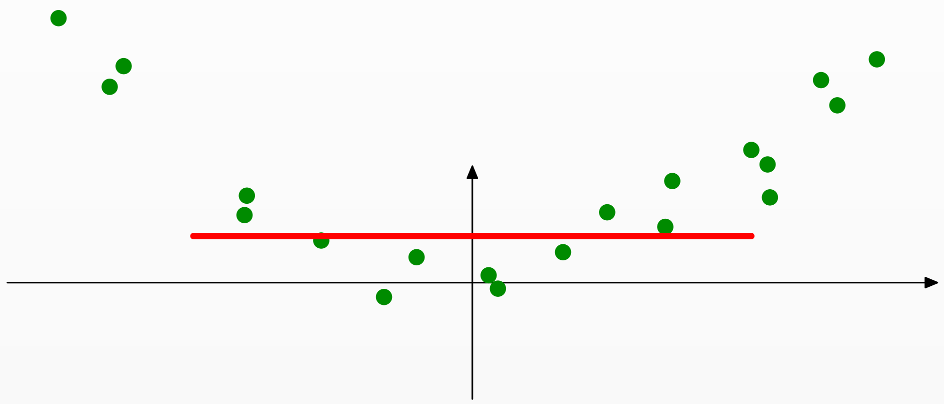
Problem is that PCA subspace is linear:

$$S = \{\mathbf{x} = \mathbf{U}\mathbf{z} : \mathbf{z} \in \mathbb{R}^k\}$$

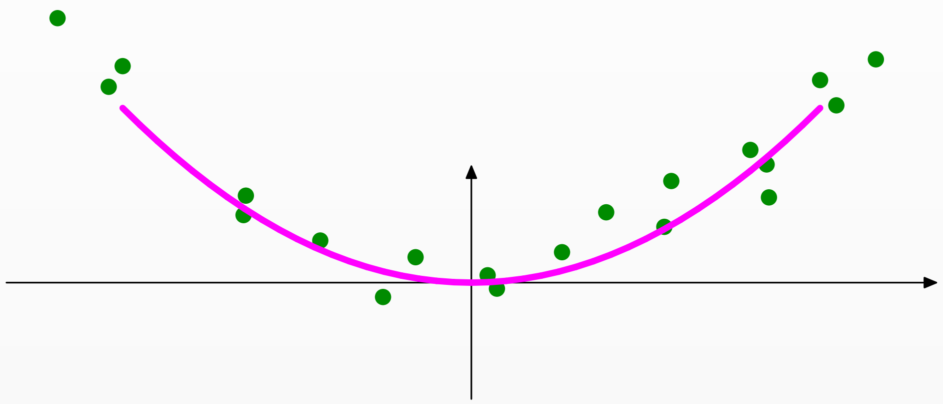
In this example:

$$S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1\}$$

Nonlinear PCA



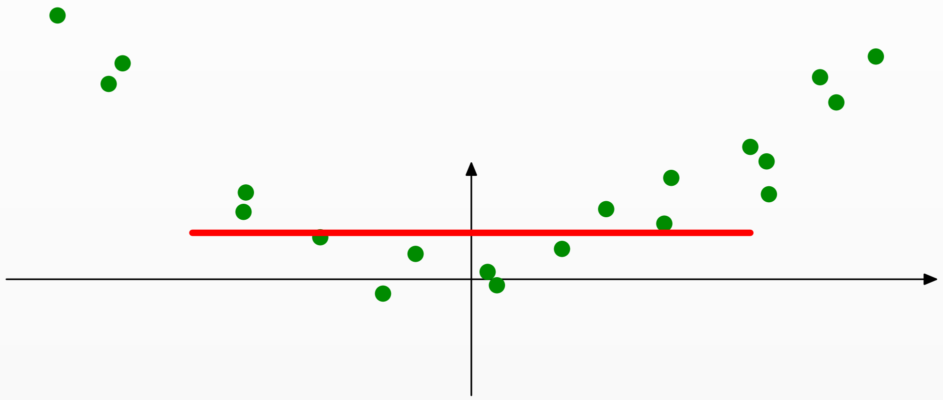
Broken solution



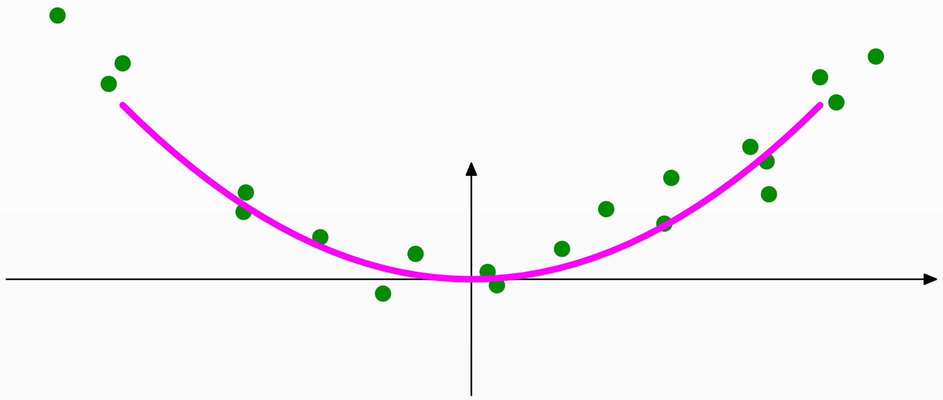
Desired solution

We want desired solution: $S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\}$

Nonlinear PCA



Broken solution

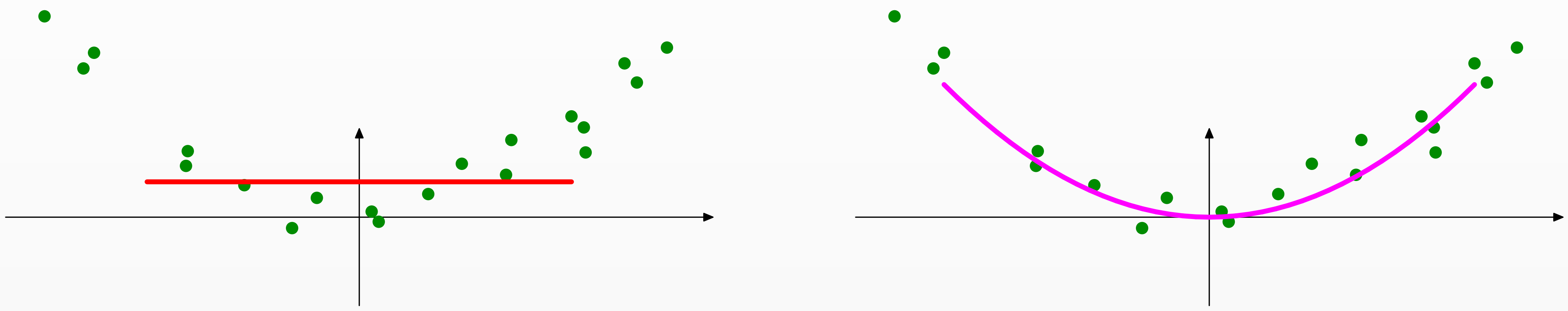


Desired solution

We want desired solution: $S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\}$

We can get this: $S = \{\phi(\mathbf{x}) = \mathbf{Uz}\}$ with $\phi(\mathbf{x}) = (x_1^2, x_2)^\top$

Nonlinear PCA



Broken solution   Desired solution

We want desired solution: $S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\}$

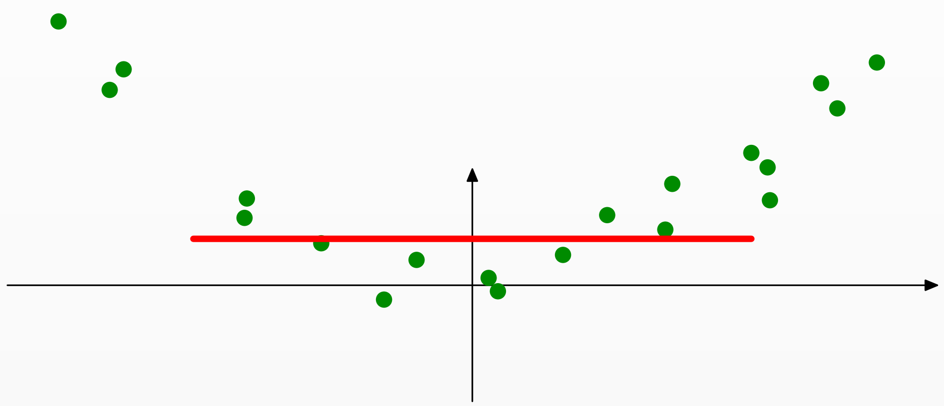
We can get this: $S = \{\phi(\mathbf{x}) = \mathbf{U}\mathbf{z}\}$ with $\phi(\mathbf{x}) = (x_1^2, x_2)^\top$

Linear dimensionality reduction in $\phi(\mathbf{x})$ space

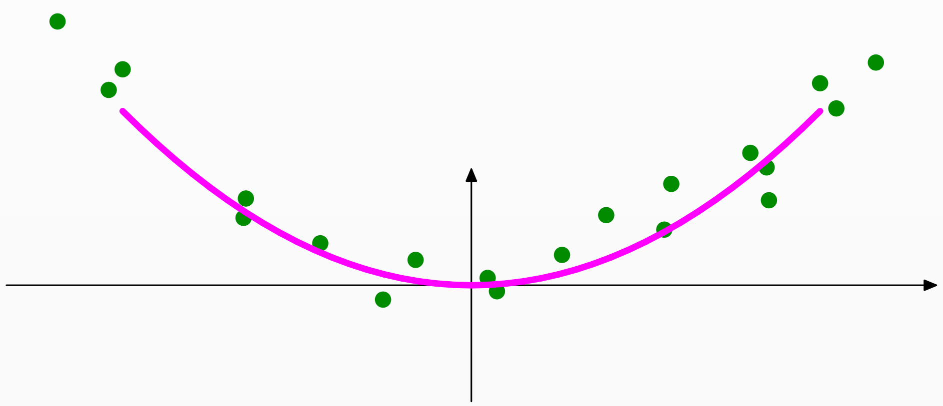
$\Updownarrow$

Nonlinear dimensionality reduction in $\mathbf{x}$ space

Nonlinear PCA



Broken solution



Desired solution

We want desired solution: $S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\}$

We can get this: $S = \{\phi(\mathbf{x}) = \mathbf{U}\mathbf{z}\}$ with $\phi(\mathbf{x}) = (x_1^2, x_2)^\top$

Linear dimensionality reduction in $\phi(\mathbf{x})$ space

$\Updownarrow$

Nonlinear dimensionality reduction in $\mathbf{x}$ space

Idea: Use kernels

Kernel PCA

Representer theorem:

$$\mathbf{X}\mathbf{X}^\top \mathbf{u} = \lambda \mathbf{u} \qquad \mathbf{u} = \mathbf{X}\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

Kernel PCA

Representer theorem:

$$\mathbf{X}\mathbf{X}^\top \mathbf{u} = \lambda \mathbf{u} \qquad\qquad \mathbf{u} = \mathbf{X}\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

Kernel function: $k(\mathbf{x}_1, \mathbf{x}_2)$ such that

K, the kernel matrix formed by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is positive semi-definite

Kernel PCA

Representer theorem:

$$\mathbf{X}\mathbf{X}^\top \mathbf{u} = \lambda \mathbf{u} \qquad\qquad \mathbf{u} = \mathbf{X}\boldsymbol{\alpha} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

Kernel function: $k(\mathbf{x}_1, \mathbf{x}_2)$ such that
K, the kernel matrix formed by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$,
is positive semi-definite

$$\max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u} = \max_{\boldsymbol{\alpha}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\alpha}=1} \boldsymbol{\alpha}^\top (\mathbf{X}^\top \mathbf{X})(\mathbf{X}^\top \mathbf{X})\boldsymbol{\alpha}$$

$$= \max_{\boldsymbol{\alpha}^\top K \boldsymbol{\alpha}=1} \boldsymbol{\alpha}^\top K^2 \boldsymbol{\alpha}$$

Kernel PCA

Direct method:

- Kernel PCA objective:

$$\max_{\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} = 1} \boldsymbol{\alpha}^\top K^2 \boldsymbol{\alpha}$$

- $\Rightarrow$ kernel PCA eigenvalue problem: $\mathbf{X}^\top \mathbf{X} \boldsymbol{\alpha} = \lambda' \boldsymbol{\alpha}$

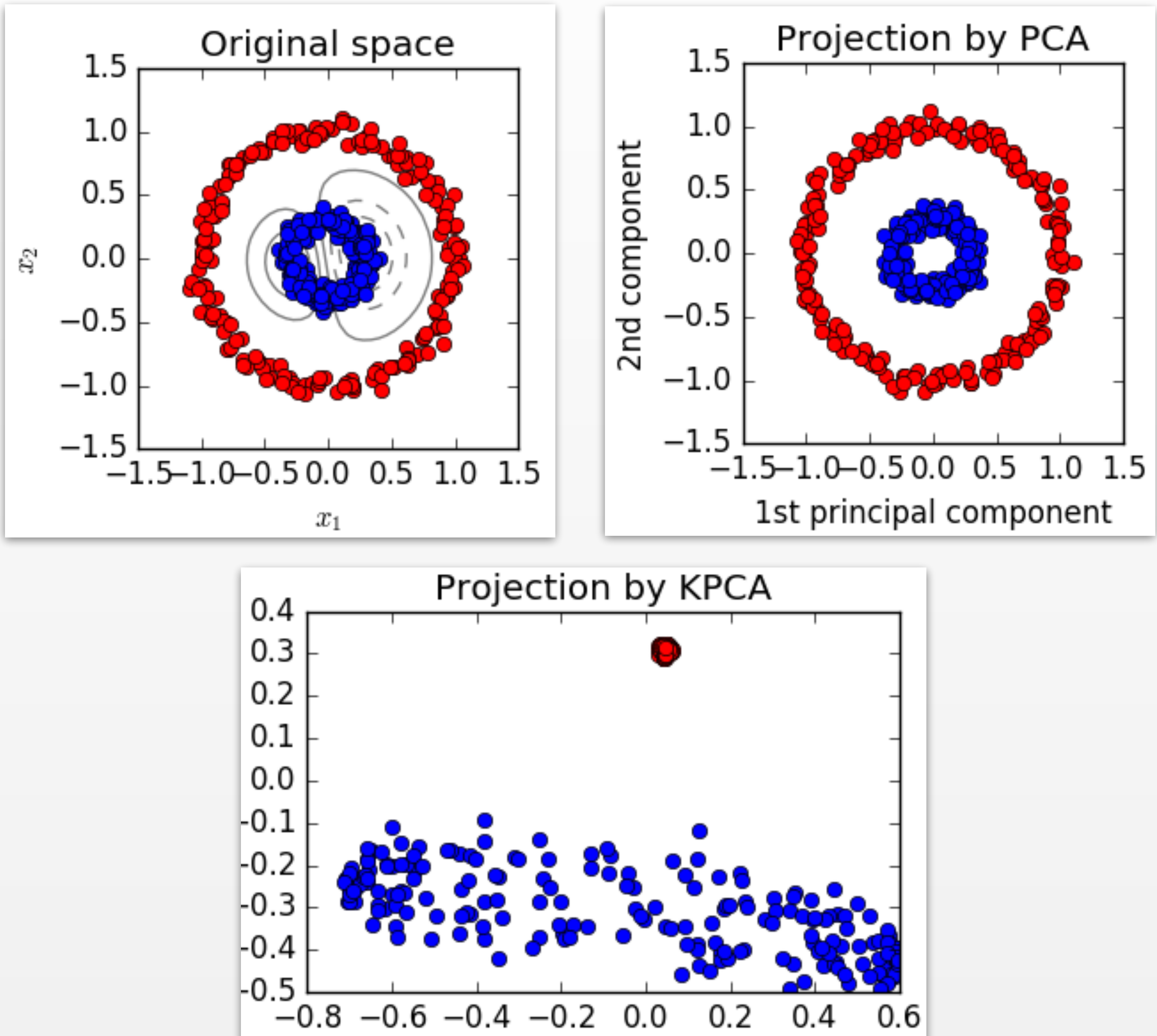
Modular method (if you don't want to think about kernels):

- Find vectors $\mathbf{x}'_1, \dots, \mathbf{x}'_n$ such that

$$\mathbf{x}'^\top_i \mathbf{x}'_j = K_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- Key: use any vectors that preserve inner products
- One possibility is Cholesky decomposition $K = \mathbf{X}'^\top \mathbf{X}'$

Kernel PCA


Original space
x_1
x_2
Projection by PCA
1st principal component
2nd component
Projection by KPCA

Canonical Correlation Analysis (CCA)

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:

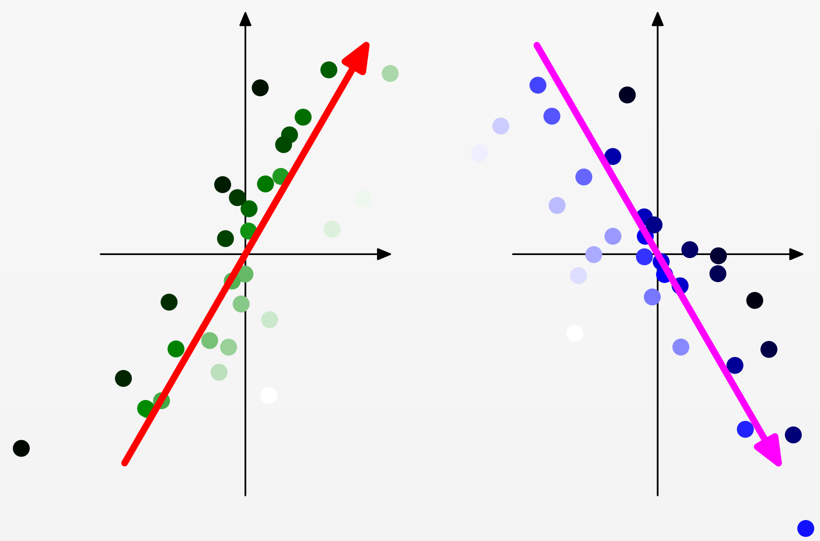
- Input data: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ (matrices $\mathbf{X}, \mathbf{Y}$)
- Goal: find pair of projections $(\mathbf{u}, \mathbf{v})$

CCA Example

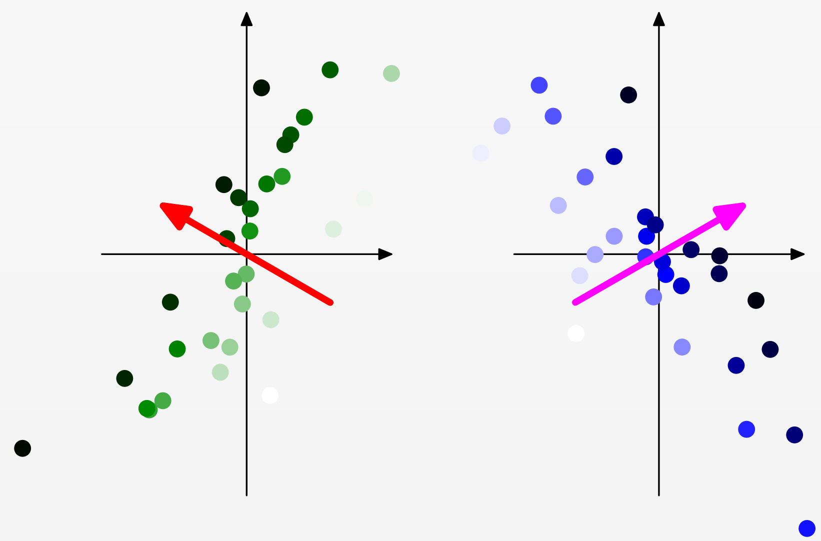
Setup:

- 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

Definitions:

Variance: $\widehat{\text{var}}(\mathbf{u}^\top \mathbf{x}) = \mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u}$

Covariance: $\widehat{\text{cov}}(\mathbf{u}^\top \mathbf{x}, \mathbf{v}^\top \mathbf{y}) = \mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}$

Correlation: $\frac{\widehat{\text{cov}}(\mathbf{u}^\top \mathbf{x}, \mathbf{v}^\top \mathbf{y})}{\sqrt{\widehat{\text{var}}(\mathbf{u}^\top \mathbf{x})}\sqrt{\widehat{\text{var}}(\mathbf{v}^\top \mathbf{y})}}$

Objective: maximize correlation between projected views

$$\max_{\mathbf{u},\mathbf{v}} \widehat{\text{corr}}(\mathbf{u}^\top \mathbf{x}, \mathbf{v}^\top \mathbf{y})$$

Properties:

- Focus on how variables are related, not how much they vary
- Invariant to any rotation and scaling of data

From PCA to CCA

PCA on views separately: no covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u}}{\mathbf{u}^\top \mathbf{u}} + \frac{\mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

PCA on concatenation $(\mathbf{X}^\top, \mathbf{Y}^\top)^\top$: includes covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u} + 2\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v} + \mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{u}^\top \mathbf{u} + \mathbf{v}^\top \mathbf{v}}$$

From PCA to CCA

PCA on views separately: no covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u}}{\mathbf{u}^\top \mathbf{u}} + \frac{\mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

PCA on concatenation $(\mathbf{X}^\top, \mathbf{Y}^\top)^\top$: includes covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u} + 2\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v} + \mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{u}^\top \mathbf{u} + \mathbf{v}^\top \mathbf{v}}$$

Maximum covariance: drop variance terms

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}}{\sqrt{\mathbf{u}^\top \mathbf{u}}\sqrt{\mathbf{v}^\top \mathbf{v}}}$$

From PCA to CCA

PCA on views separately: no covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u}}{\mathbf{u}^\top \mathbf{u}} + \frac{\mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

PCA on concatenation $(\mathbf{X}^\top, \mathbf{Y}^\top)^\top$: includes covariance term

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u} + 2\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v} + \mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}{\mathbf{u}^\top \mathbf{u} + \mathbf{v}^\top \mathbf{v}}$$

Maximum covariance: drop variance terms

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}}{\sqrt{\mathbf{u}^\top \mathbf{u}}\sqrt{\mathbf{v}^\top \mathbf{v}}}$$

Maximum correlation (CCA): divide out variance terms

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}}{\sqrt{\mathbf{u}^\top \mathbf{X}\mathbf{X}^\top \mathbf{u}}\sqrt{\mathbf{v}^\top \mathbf{Y}\mathbf{Y}^\top \mathbf{v}}}$$

Importance of Regularization

Extreme examples of degeneracy:

- If $\mathbf{x} = A\mathbf{y}$, then any $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} = A\mathbf{v}$ is optimal (correlation 1)
- If $\mathbf{x}$ and $\mathbf{y}$ are independent, then any $(\mathbf{u}, \mathbf{v})$ is optimal (correlation 0)

Importance of Regularization

Extreme examples of degeneracy:

- If $\mathbf{x} = A\mathbf{y}$, then any $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} = A\mathbf{v}$ is optimal (correlation 1)
- If $\mathbf{x}$ and $\mathbf{y}$ are independent, then any $(\mathbf{u}, \mathbf{v})$ is optimal (correlation 0)

Problem: if $\mathbf{X}$ or $\mathbf{Y}$ has rank n, then any $(\mathbf{u}, \mathbf{v})$ is optimal

(correlation 1) with $\mathbf{u} = \mathbf{X}^\top \mathbf{Y} \mathbf{v} \Rightarrow$ CCA is meaningless!

Importance of Regularization

Extreme examples of degeneracy:

- If $\mathbf{x} = A\mathbf{y}$, then any $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} = A\mathbf{v}$ is optimal (correlation 1)
- If $\mathbf{x}$ and $\mathbf{y}$ are independent, then any $(\mathbf{u}, \mathbf{v})$ is optimal (correlation 0)

Problem: if $\mathbf{X}$ or $\mathbf{Y}$ has rank n, then any $(\mathbf{u}, \mathbf{v})$ is optimal

(correlation 1) with $\mathbf{u} = \mathbf{X}^\top \mathbf{Y}\mathbf{v} \Rightarrow$ CCA is meaningless!

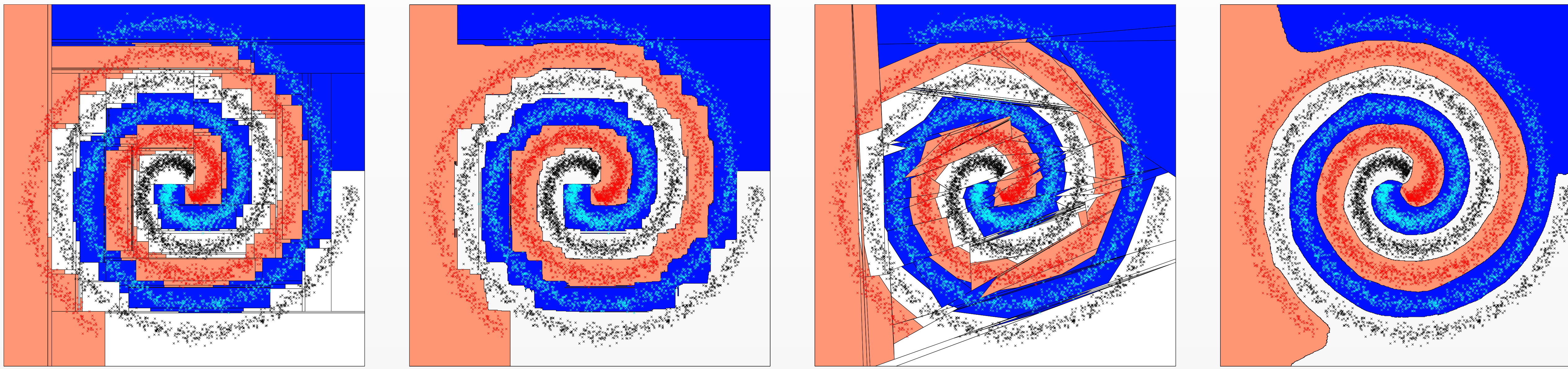
Solution: regularization (interpolate between

maximum covariance and maximum correlation)

$$\max_{\mathbf{u},\mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}}{\sqrt{\mathbf{u}^\top (\mathbf{X}\mathbf{X}^\top + \lambda I)\mathbf{u}}\sqrt{\mathbf{v}^\top (\mathbf{Y}\mathbf{Y}^\top + \lambda I)\mathbf{v}}}$$

Canonical Correlation Forests

(a) Single CART (unpruned) (b) RF with 200 Trees (c) Single CCT (unpruned) (d) CCF with 200 Trees



Canonical Correlation Forests

Tom Rainforth, Frank Wood

(Submitted on 20 Jul 2015 (v1), last revised 5 Dec 2015 (this version, v5))

We introduce canonical correlation forests (CCFs), a new decision tree ensemble method for classification. Individual canonical correlation trees are binary decision trees with hyperplane splits based on canonical correlation components. Unlike axis-aligned alternatives, the decision surfaces of CCFs are not restricted to the coordinate system of the input features and therefore more naturally represent data with correlation between the features. Additionally we introduce a novel alternative to bagging, the projection bootstrap, which maintains use of the full dataset in selecting split points. CCFs do not require parameter tuning and our experiments show that they out-perform axis-aligned random forests, other state-of-the-art tree ensemble methods and all of the 179 popular classifiers considered in a recent extensive survey.

Example: RF that uses CCA to determine axis for splits

Summary

Framework: $\mathbf{z} = \mathbf{U}^\top \mathbf{x}$, $\mathbf{x} \approxeq \mathbf{U}\mathbf{z}$

Criteria for choosing $\mathbf{U}$:

- PCA: maximize projected variance
- CCA: maximize projected correlation

Algorithm: generalized eigenvalue problem

Extensions:

non-linear using kernels (using same linear framework)

probabilistic, sparse, robust (hard optimization)