

PCA and Kernel PCA

Learning Representations.
Dimensionality Reduction.

Maria-Florina Balcan

04/08/2015

Big & High-Dimensional Data

- High-Dimensions = Lot of Features

Document classification

Features per document =
thousands of words/unigrams
millions of bigrams, contextual
information



Surveys - Netflix

480189 users x 17770 movies

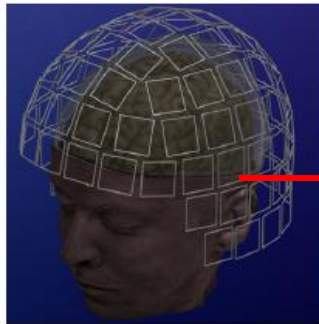
	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

Big & High-Dimensional Data

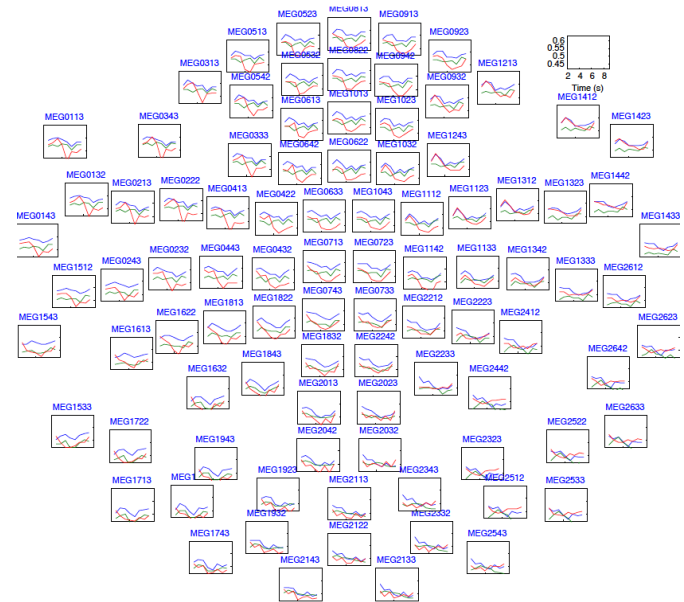
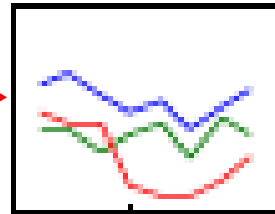
- High-Dimensions = Lot of Features

MEG Brain Imaging

120 locations x 500 time points
x 20 objects



MEG0633



Or any high-dimensional image data



- Big & High-Dimensional Data.
- Useful to learn lower dimensional representations of the data.

Learning Representations

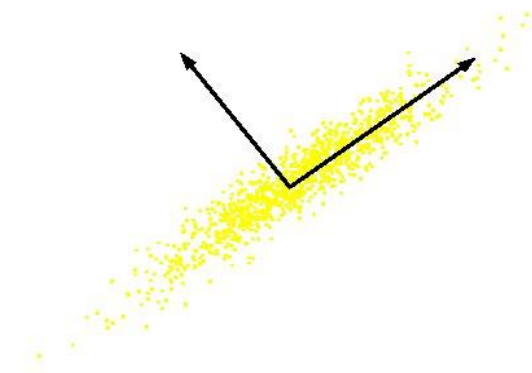
PCA, Kernel PCA, ICA: Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

Useful for:

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)
- Further processing by machine learning algorithms

Principal Component Analysis (PCA)

What is PCA: Unsupervised technique for extracting variance structure from high dimensional datasets.

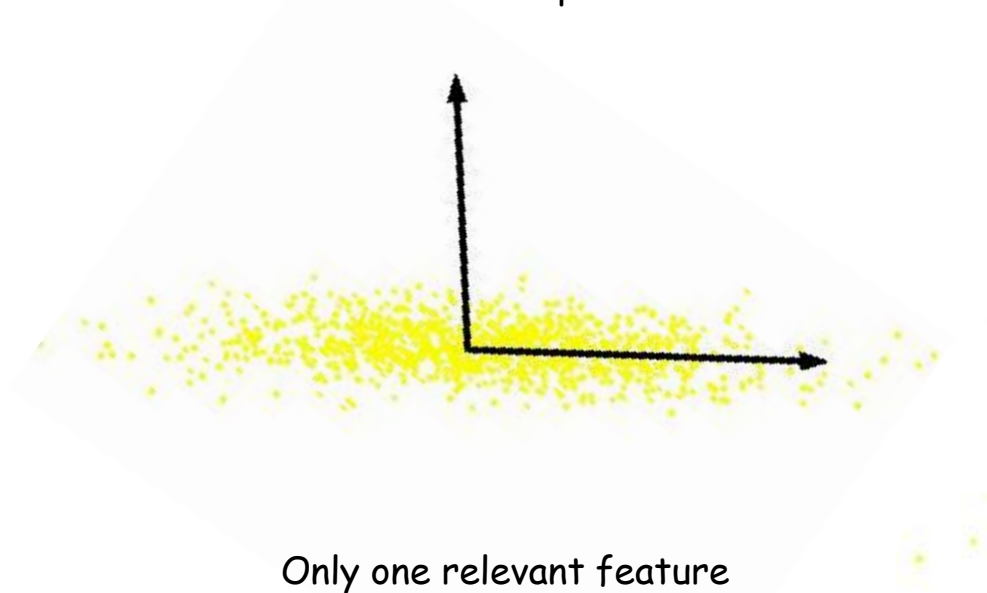


- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

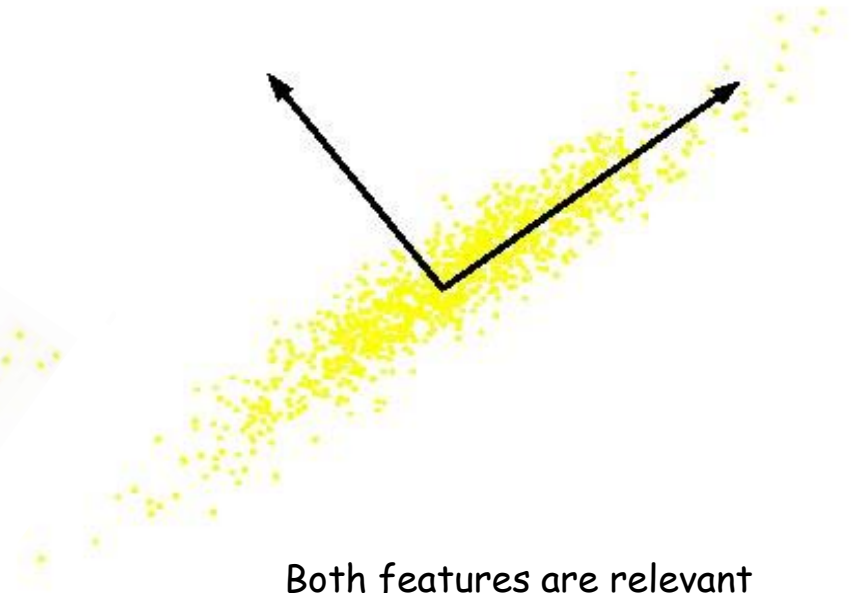
Principal Component Analysis (PCA)

Intrinsically lower dimensional than the dimension of the ambient space.

If we rotate data, again only one coordinate is more important.



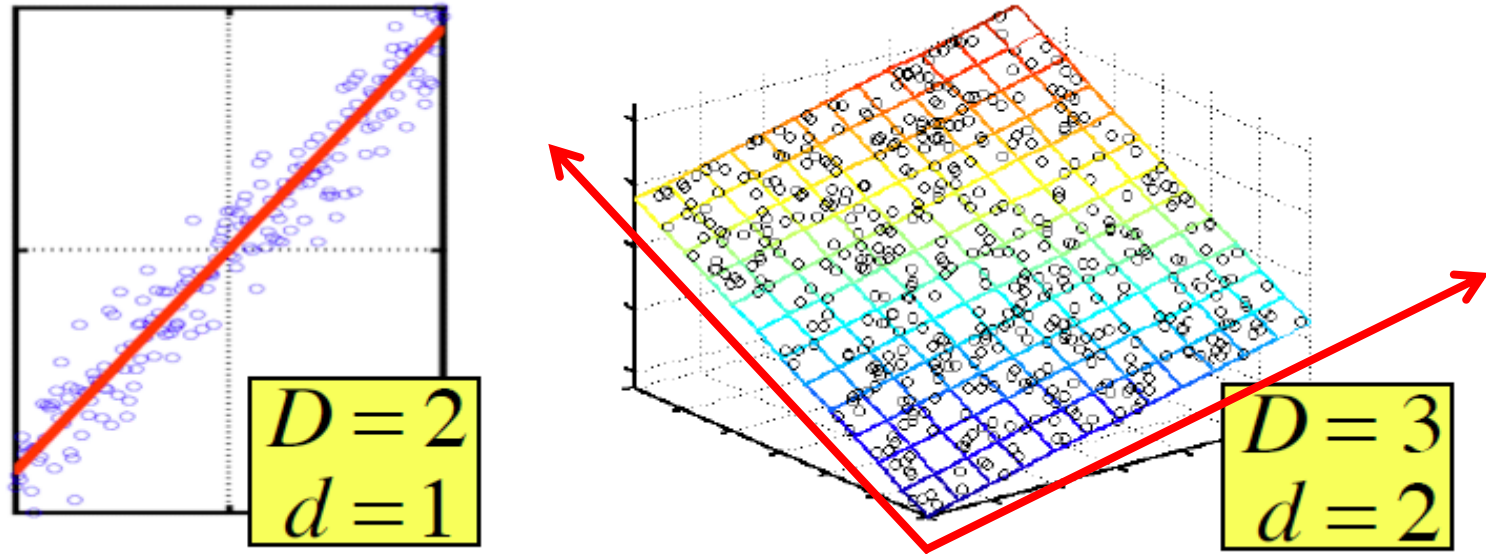
Only one relevant feature



Both features are relevant

Question: Can we transform the features so that we only need to preserve one latent feature?

Principal Component Analysis (PCA)



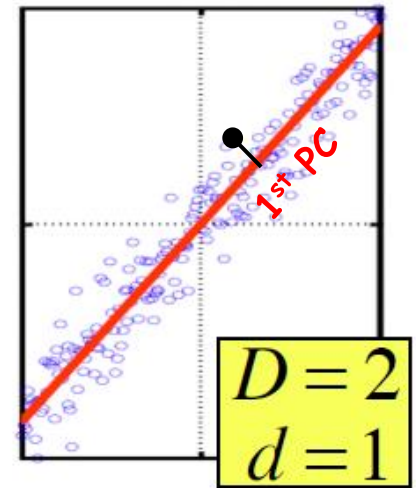
In case where data lies on or near a low d -dimensional linear subspace, axes of this subspace are an effective representation of the data.

Identifying the axes is known as **Principal Components Analysis**, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

Principal Component Analysis (PCA)

Principal Components (PC) are orthogonal directions that capture most of the variance in the data.

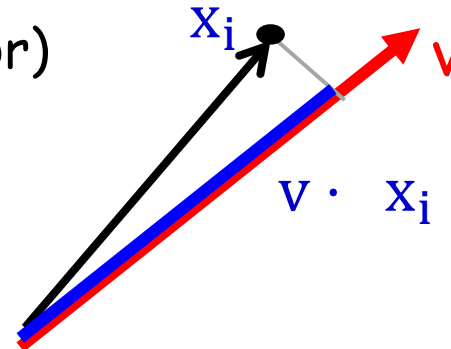
- First PC - direction of greatest variability in data.
- Projection of data points along first PC discriminates data most along any one direction (pts are the most spread out when we project the data on that direction compared to any other directions).



Quick reminder:

$\|v\|=1$, Point x_i (D-dimensional vector)

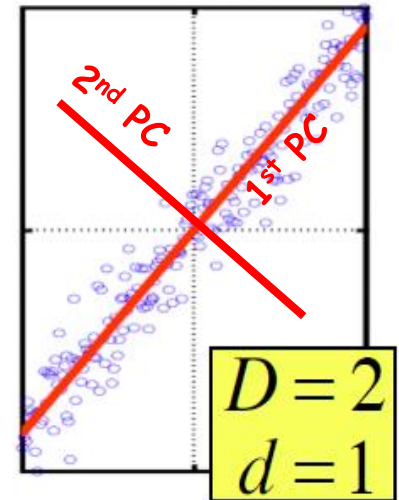
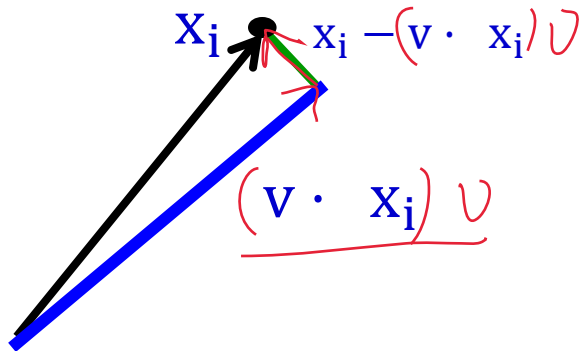
Projection of x_i onto v is $v \cdot x_i$



Principal Component Analysis (PCA)

Principal Components (PC) are orthogonal directions that capture most of the variance in the data.

- 1st PC - direction of greatest variability in data.



- 2nd PC - Next orthogonal (uncorrelated) direction of greatest variability

(remove all variability in first direction, then find next direction of greatest variability)

- And so on ...

Principal Component Analysis (PCA)

$$V = [v_1, v_2, \dots, v_d] \in \mathbb{R}^{D \times d}$$

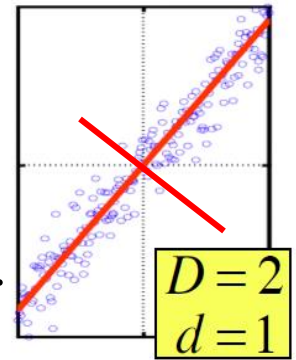
Let $\underline{v_1, v_2, \dots, v_d}$ denote the \underline{d} principal components. $\underline{d} < \underline{D}$

$$v_i \cdot v_j = 0, i \neq j \text{ and } v_i \cdot v_i = 1, i = j$$

Assume data is centered (we extracted the sample mean).

$$X \leftarrow X - C_n$$

Let $X = [x_1, x_2, \dots, x_n]$ (columns are the datapoints)



Find vector that maximizes sample variance of projected data

$$\frac{1}{n} \sum_{i=1}^n (v^T x_i)^2 = v^T \underline{XX^T} v$$

$$\max_v \underline{v^T XX^T v} \quad \text{s.t.} \quad \underline{v^T v = 1}$$

$$\text{Lagrangian: } \max_v v^T XX^T v - \lambda v^T v$$

$$\partial / \partial v = 0$$

$$(XX^T - \lambda I)v = 0$$

$$\Rightarrow \boxed{(XX^T)v = \lambda v}$$

$$(XX^T)v = \lambda v$$

$$C_m = I_m - \frac{1}{n} \mathbb{1}_m \mathbb{1}_m^T$$

$$C_m^k = C_m$$

$$\frac{1}{n} \sum_{i=1}^n (x_i - \mu_i) (x_i - \mu_i)^T = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$$

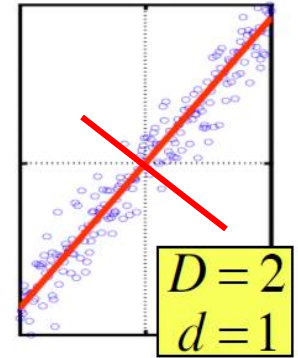
Wrap constraints into the objective function $= \frac{1}{n} X \cdot X^T$

Principal Component Analysis (PCA)

$(X X^T) \mathbf{v} = \underline{\lambda \mathbf{v}}$, so \mathbf{v} (the first PC) is the eigenvector of sample correlation/covariance matrix $X X^T$

Sample variance of projection $(\mathbf{v}^T X X^T) \mathbf{v} = \lambda \mathbf{v}^T \mathbf{v} = \lambda$
 $(\mathbf{v}^T (X X^T) \mathbf{v})^T$

Thus, the eigenvalue λ denotes the amount of variability captured along that dimension (aka amount of energy along that dimension).

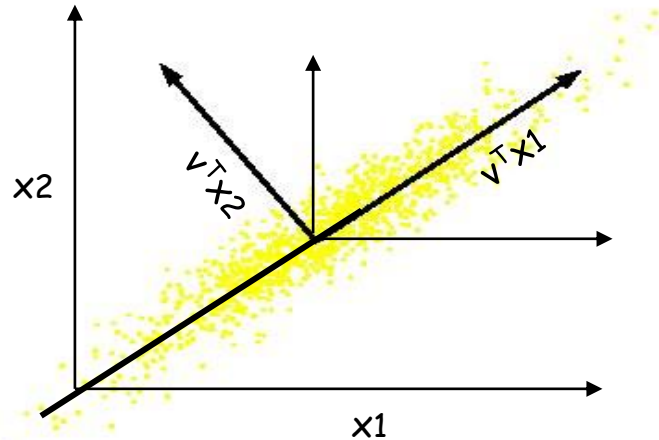


Eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots$

- The 1st PC \mathbf{v}_1 is the the eigenvector of the sample covariance matrix $X X^T$ associated with the largest eigenvalue
- The 2nd PC \mathbf{v}_2 is the the eigenvector of the sample covariance matrix $X X^T$ associated with the second largest eigenvalue
- And so on ...

Principal Component Analysis (PCA)

- So, the new axes are the eigenvectors of the matrix of sample correlations $\mathbf{X}\mathbf{X}^T$ of the data.
- Transformed features are uncorrelated.



- Geometrically: centering followed by rotation.
 - Linear transformation

Key computation: eigendecomposition of $\mathbf{X}\mathbf{X}^T$ (closely related to SVD of \mathbf{X}).

Algorithms of PCA

Alg. 1. $X \in \mathbb{R}^{D \times n}$ $(X \cdot \mathbb{1}_n = 0_d)$

1. Data centering

$$X \leftarrow X \cdot C_n$$

2. Sample var.

$$\frac{1}{n} X X^T \leftarrow \mathcal{O}(nD^2)$$

$$V = \{v_1, v_2, \dots, v_d\} \in \mathbb{R}^{D \times d}$$

3. $\left(\frac{1}{n} X X^T \right) V = V \Lambda \quad \mathcal{O}(D^3)$

4. d eigenvectors w.r.t. top-d eigenvalues

V

$$X^T = U D V^T$$

$$\begin{aligned} X X^T &= V D U^T U D U^T \\ &= V \underline{D^2} \bar{U}^T = V \cdot \Lambda V^T \end{aligned}$$

$$(X X^T) V = V \cdot \Lambda$$

Alg 2.

1. —

$$X \leftarrow X \cdot C_n$$

2. SVD on X

$$X^T = U D V^T$$

3.

$$X \in \mathbb{R}^{D \times n}$$

$$\mathcal{O}(\min\{nD^2, Dn^2\})$$

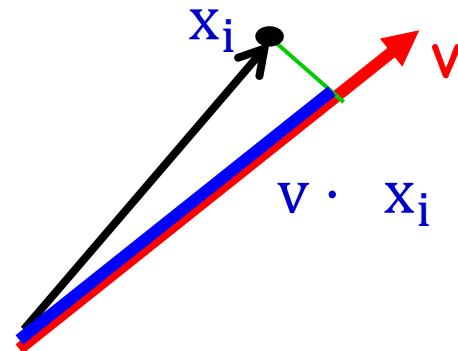
Two Interpretations

So far: **Maximum Variance Subspace**. PCA finds vectors \mathbf{v} such that projections on to the vectors capture maximum variance in the data

$$\frac{1}{n} \sum_{i=1}^n (\mathbf{v}^T \mathbf{x}_i)^2 = \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v}$$

Alternative viewpoint: **Minimum Reconstruction Error**. PCA finds vectors \mathbf{v} such that projection on to the vectors yields minimum MSE reconstruction

$$\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - (\mathbf{v}^T \mathbf{x}_i) \mathbf{v}\|^2$$



Two Interpretations

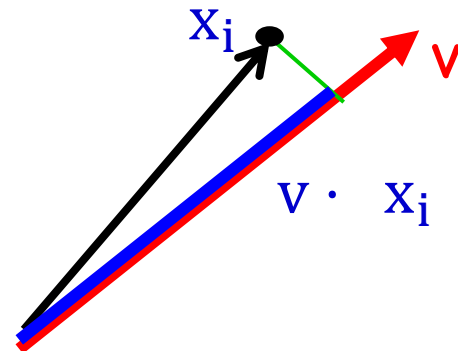
E.g., for the first component.

Maximum Variance Direction: 1st PC a vector v such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections)

$$\frac{1}{n} \sum_{i=1}^n (\mathbf{v}^T \mathbf{x}_i)^2 = \mathbf{v}^T \mathbf{X} \mathbf{X}^T \mathbf{v}$$

Minimum Reconstruction Error: 1st PC a vector v such that projection on to this vector yields minimum MSE reconstruction

$$\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - (\mathbf{v}^T \mathbf{x}_i) \mathbf{v}\|^2$$



Why? Pythagorean Theorem

E.g., for the first component.

Maximum Variance Direction: 1st PC a vector v such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections)

$$\frac{1}{n} \sum_{i=1}^n (v^T x_i)^2 = v^T X X^T v$$

$\underline{v^T X X^T v}$

$$X = \mathbb{R}^{D \times n}$$

$$V = [v_1, v_2, \dots, v_d]$$

$$\frac{1}{n} \sum_{i=1}^n \|x_i - (v^T x_i)v\|^2$$

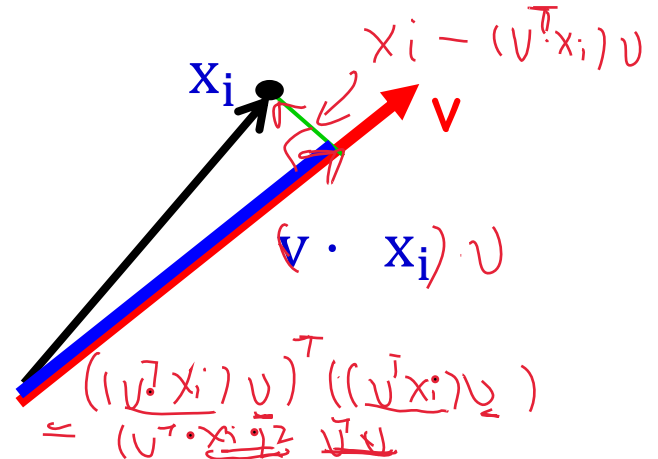
$$= \|X^T - X^T V \cdot V^T\|_F^2$$

Minimum Reconstruction Error: 1st PC a vector v such that projection on to this vector yields minimum MSE reconstruction

$$\underbrace{(v^T x_i)^2}_{\text{blue}^2} + \underbrace{x_i^T x_i}_{\text{green}^2} = \underbrace{\|x_i - (v^T x_i)v\|_2^2}_{\text{black}^2}$$

black² is fixed (it's just the data)

So, maximizing blue² is equivalent to minimizing green²

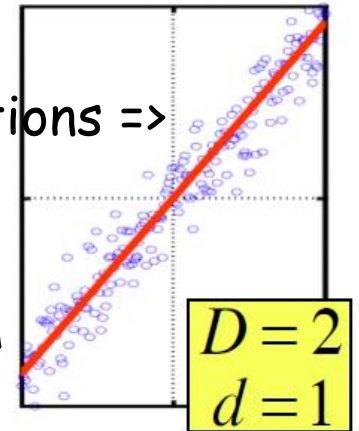


Dimensionality Reduction using PCA

The eigenvalue λ denotes the amount of variability captured along that dimension (aka amount of energy along that dimension).

Zero eigenvalues indicate no variability along those directions => data lies exactly on a linear subspace

Only keep data projections onto principal components with non-zero eigenvalues, say v_1, \dots, v_d , where $d = \text{rank}(X X^T)$



$$V = [v_1, \dots, v_d] \in \mathbb{R}^{p \times d}$$

Original representation

Transformed representation

Data point

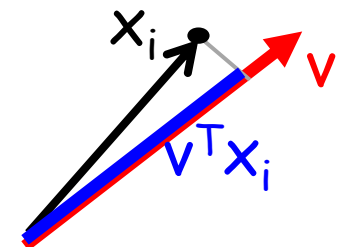
$$x_i = (x_i^1, \dots, x_i^D)^T$$

D-dimensional vector

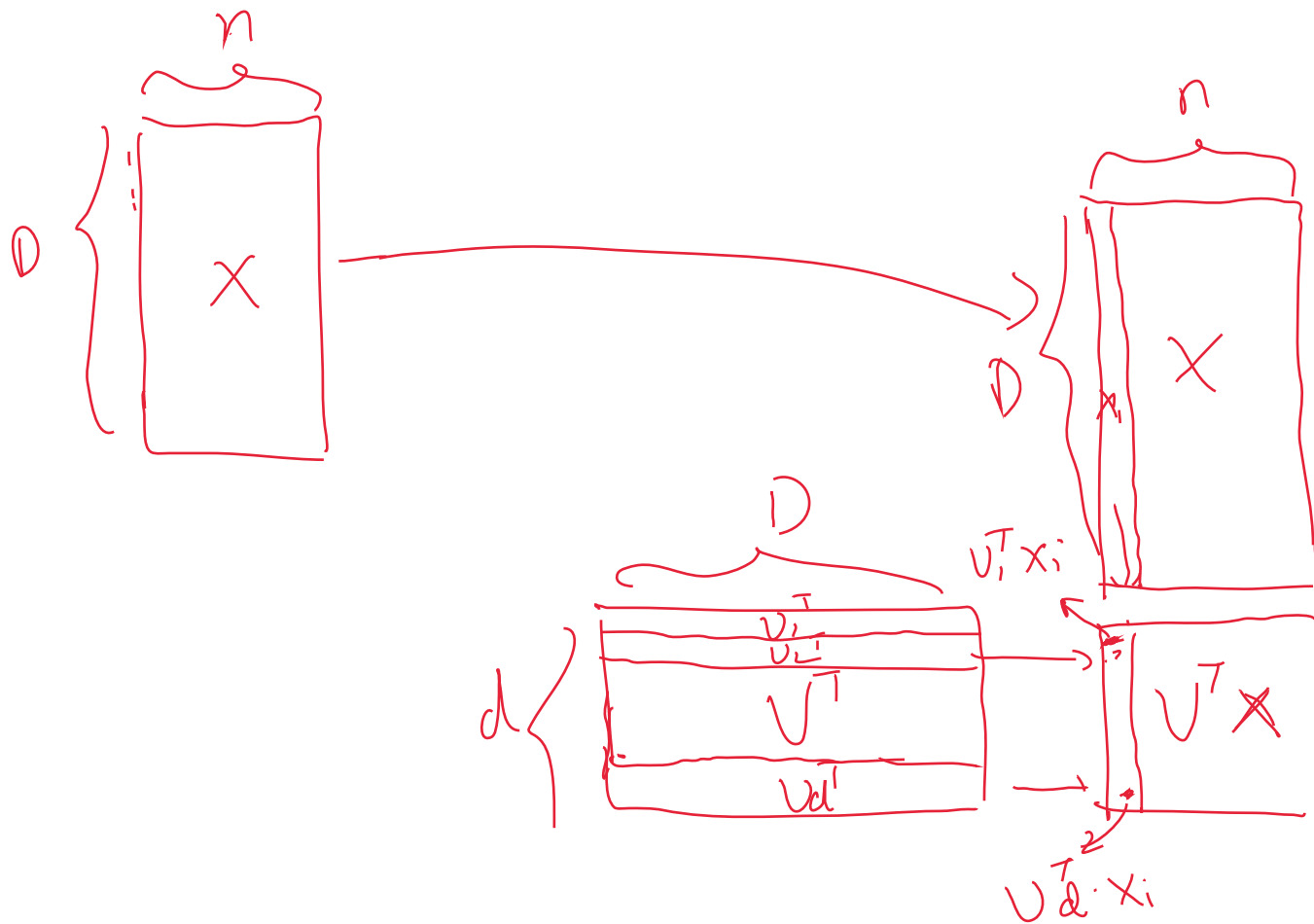
projection

$$V^T x_i = (v_1 \cdot x_i^1, \dots, v_d \cdot x_i^d)^T$$

d-dimensional vector



$$X \in \mathbb{R}^{D \times n} \xrightarrow[V]{\text{PLA}} V^T X \in \mathbb{R}^{d \times n}$$

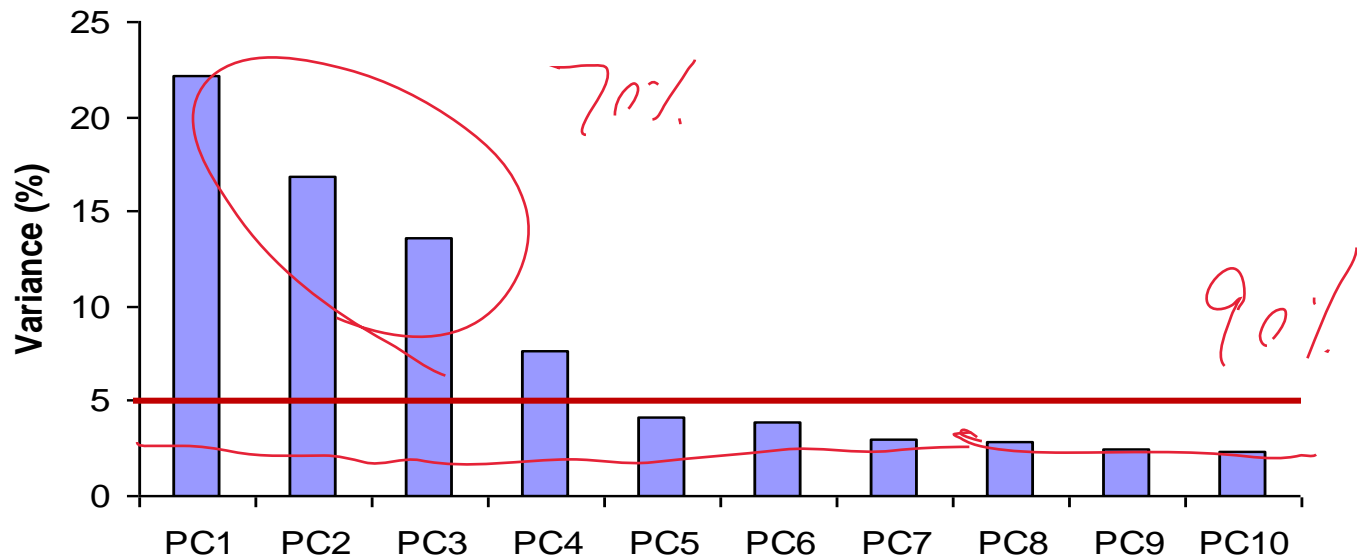


Dimensionality Reduction using PCA

In high-dimensional problems, data sometimes lies near a linear subspace, as noise introduces small variability

Only keep data projections onto principal components with **large** eigenvalues

Can *ignore* the components of smaller significance.



Might **lose some info**, but if eigenvalues are small, do not lose much

Defined only in this slide.

Low-Rank Approximation

$$X \in \mathbb{R}^{n \times d}$$

$$\begin{aligned} \|A\|_F^2 &= \text{Tr}(A^T A) \\ &= \text{Tr}(A A^T) \\ &= \langle A, A \rangle \end{aligned}$$

PCA

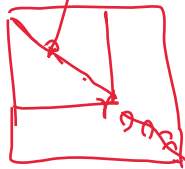
$$\min_U \left\| X - \underbrace{X U U^T}_{\text{rank}(X)} \right\|_F^2$$

s.t. $U^T U = I$ rank $\leq d$

$$\begin{aligned} \min_{\hat{X}} & \|X - \hat{X}\|_F^2 \\ \text{s.t. } & \text{rank}(\hat{X}) \leq r \end{aligned}$$

$$X = U D V^T$$

$$\hat{X} = U \underbrace{D_{1:r}}_{\Delta D} V^T$$



k-means

$$\begin{aligned} \min_{R, C} & \|X - RC\|_F^2 \\ \text{s.t. } & R \in \mathbb{R}^{n \times k}, C \in \mathbb{R}^{k \times d} \end{aligned}$$

$$\begin{aligned} \|X - \hat{X}\|_F^2 &= \|U(\underbrace{D - D_{1:r}}_{\Delta D})V^T\|_F^2 \\ &= \text{Tr}(U \Delta D^2 U^T) = \text{Tr}(\Delta D^2) \\ &= \sum_{i=r+1}^D \sigma_i^2 \end{aligned}$$

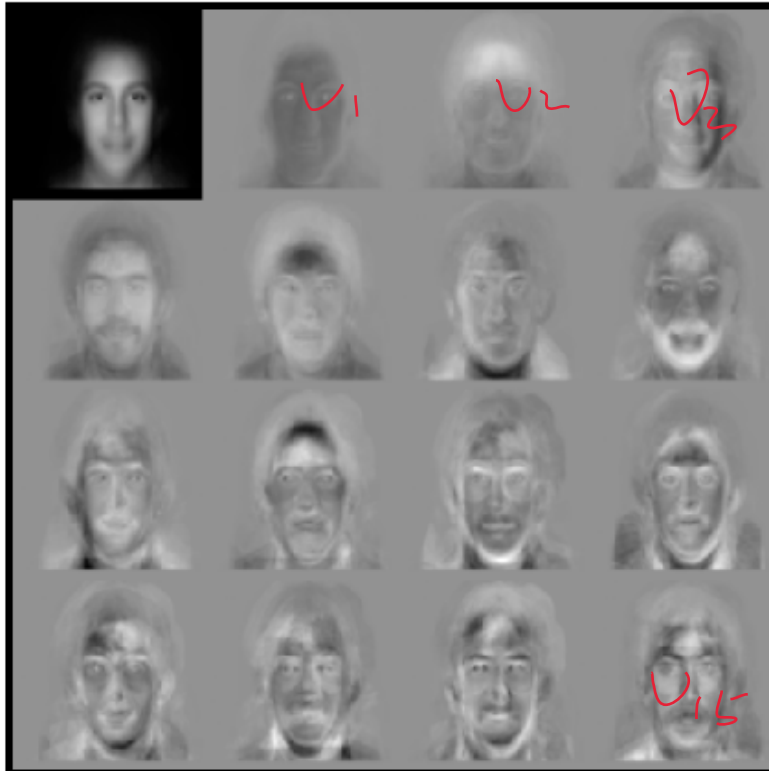
$$\min_{\hat{X}} \|X - \hat{X}\|_F^2 + \lambda (\|\hat{X}\|_*)$$

Matrix completion

$$\|A\|_* = \text{Tr}(\sqrt{A^T A})$$

ℓ_1 -norm on singular values of X

Example: faces



Eigenfaces
from 7562
images:

**top left image
is linear
combination
of rest.**

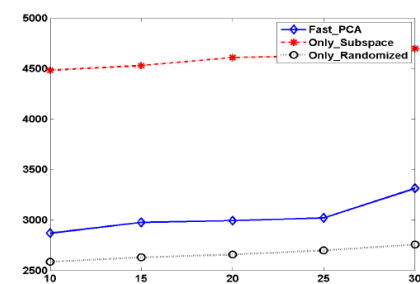
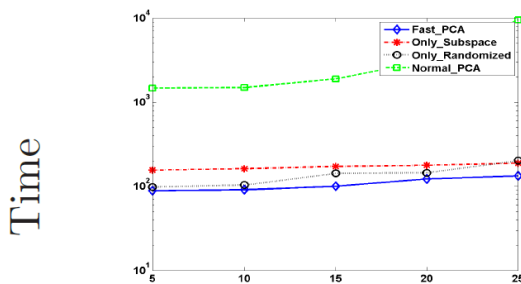
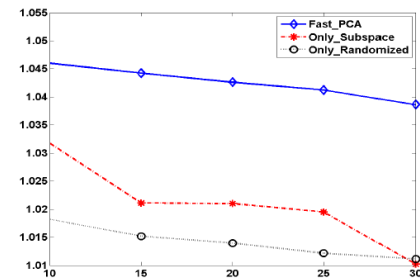
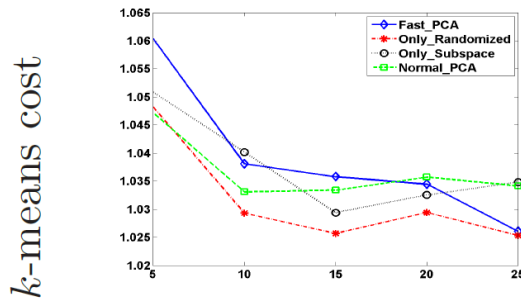
Sirovich & Kirby (1987)
Turk & Pentland (1991)

Can represent a face image using just 15 numbers!

Quiz 

- PCA provably useful before doing k-means clustering and also empirically useful. E.g.,

- ▷ **Performance:** cost increase $< 5\%$; $\times 10$ to $\times 100$ speedup
- ▷ ***k*-Means Clustering:** *k*-means cost/time vs dimension



NewsGroups

BOWpubmed

PCA Discussion

Strengths

Eigenvector method

No tuning of the parameters

No local optima

Weaknesses

Limited to second order statistics

Limited to linear projections

n th-order 矩: $\mu_n =$

$$\mu_n = \int (x - c)^n f(x) dx$$

$n = \begin{cases} 1 \leftarrow \text{期望} \\ 2 \leftarrow \text{方差} \\ 3 \leftarrow \text{偏态} \\ 4 \leftarrow \text{峰态} \end{cases}$

$c = \begin{cases} 0, & n=1 \\ E[x], & n \geq 2 \end{cases}$

kernel trick:

$$k(x_i, x_j) = \langle \underline{\phi(x_i)}, \underline{\phi(x_j)} \rangle$$

Kernel PCA (Kernel Principal Component Analysis)

Useful when data lies on or near a low d-dimensional linear subspace of the ϕ -space associated with a kernel

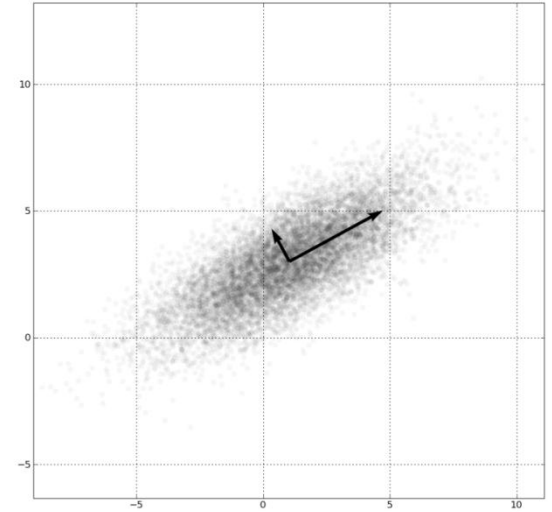
Properties of PCA

- Given a set of n centered observations $x_i \in \mathbb{R}^D$, 1st PC is the direction that maximizes the variance

- $X = (x_1, x_2, \dots, x_n)$

- $v_1 = \operatorname{argmax}_{\|v\|=1} \frac{1}{n} \sum_i (v^\top x_i)^2$
 $= \operatorname{argmax}_{\|v\|=1} \frac{1}{n} v^\top X X^\top v$

- Covariance matrix $C = \frac{1}{n} X X^\top$
- v_1 can be found by solving the eigenvalue problem:
 - $C v_1 = \lambda v_1$ (of maximum λ)



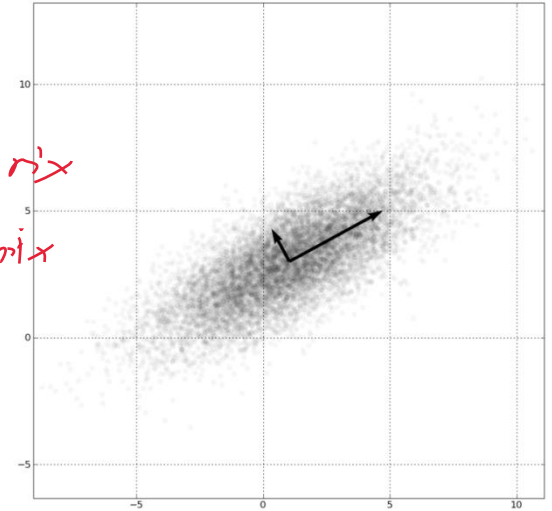
Properties of PCA

- Given a set of n centered observations $x_i \in \mathbb{R}^D$, 1st PC is the direction that maximizes the variance

- $X = (x_1, x_2, \dots, x_n)$

- $$v_1 = \operatorname{argmax}_{\|v\|=1} \frac{1}{n} \sum_i (v^\top x_i)^2$$
$$= \operatorname{argmax}_{\|v\|=1} \frac{1}{n} v^\top X X^\top v$$

XX^\top : scatter matrix
 $X^\top X$: gram matrix



- Covariance matrix $C = \frac{1}{n} X X^\top$ is a $D \times D$ matrix
the (i,j) entry of XX^\top is the correlation of the i -th coordinate of examples with j th coordinate of examples
- To use kernels, need to use the inner-product matrix $X^\top X$.

$$X^T = U D U^T$$

V 's cols lie in the span of column space of X \mathbb{R}^d

Alternative expression for PCA

- The principal component lies in the span of the data

$$V = [v_1, v_2, \dots, v_d]$$

$$V = X A \quad A = [\alpha_1, \alpha_2, \dots, \alpha_d] \quad v_1 = \sum_{i=1}^n \alpha_i x_i = X \alpha$$

$$C v_1 = \lambda v_1$$

Why? 1st PC is direction of largest variance, and for any direction outside of the span of the data, only get more variance if we project that direction into the span.

$$\text{span}(S) = \left\{ \sum_i \lambda_i v_i \mid v_i \in S, \lambda_i \in \mathbb{R} \right\}$$

线性生成空间.

- Plug this in we have

$$C v_1 = \frac{1}{n} X X^T X \alpha = \lambda X \alpha$$

- Now, left-multiply the LHS and RHS by X^T .

$$\frac{1}{n} X^T X X^T X \alpha = \lambda X^T X \alpha$$

Only depends on the inner product matrix

Kernel PCA

- **Key Idea:** Replace inner product matrix by kernel matrix

- PCA: $\frac{1}{n} \underbrace{X^T X}_{K} \underbrace{X^T X}_{K} \alpha = \lambda \underbrace{X^T X}_{K} \alpha$

- Let $K = [K(x^i, x^j)]_{ij}$ be the matrix of all dot-products in the ϕ -space.

- Kernel PCA: replace " $X^T X$ " with K .

$\frac{1}{n} K K \alpha = \lambda K \alpha$, or equivalently, $\frac{1}{n} K \alpha = \lambda \alpha$

$\frac{1}{n} K A = A \Lambda$

- **Key computation:** form an n by n kernel matrix K , and then perform eigen-decomposition on K .

kernel k -means

similar

$\max_A A^T K A$
s.t. $A^T A = I, A = R D^{\frac{1}{2}}, R \in \mathbb{R}^{n \times k}$

$\max_A A^T (\frac{1}{n} K) A$
s.t. $A^T A = I$

Data centering

Kernel PCA

Training:

$$X \leftarrow \underline{X - \frac{1}{n} \mathbb{1} \mathbb{1}^T} = \underline{X \left(\mathbb{I}_n - \frac{1}{n} \mathbb{1} \mathbb{1}^T \right)}$$
$$K_c = X^T X$$
$$X \in \mathbb{R}^{D \times n}$$

$$= \underline{\left(\mathbb{I}_n - \frac{1}{n} \mathbb{1} \mathbb{1}^T \right)^T X^T X \left(\mathbb{I}_n - \frac{1}{n} \mathbb{1} \mathbb{1}^T \right)}$$
$$= K - K \frac{\mathbb{1} \mathbb{1}^T}{n} - \frac{\mathbb{1} \mathbb{1}^T}{n} K + \frac{\mathbb{1} \mathbb{1}^T}{n} K \frac{\mathbb{1} \mathbb{1}^T}{n}$$

Test:

Given a test data point $\hat{x}_0 \in \mathbb{R}^D$

$$k_c = \left(X - X \frac{\mathbb{1} \mathbb{1}^T}{n} \right) \left(\hat{x} - X \frac{\mathbb{1}}{n} \right)$$
$$= \underbrace{X^T \hat{x}}_k - \underbrace{X^T X}_{K} \frac{\mathbb{1}}{n} - \frac{\mathbb{1} \mathbb{1}^T}{n} \hat{x} \hat{x} + \frac{\mathbb{1} \mathbb{1}^T}{n} X^T X \frac{\mathbb{1}}{n} = k - K \frac{\mathbb{1}}{n} - \frac{\mathbb{1} \mathbb{1}^T}{n} k + \frac{\mathbb{1} \mathbb{1}^T}{n} K \frac{\mathbb{1}}{n}$$

Kernel PCA Example

- Gaussian RBF kernel $\exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ over 2 dimensional space

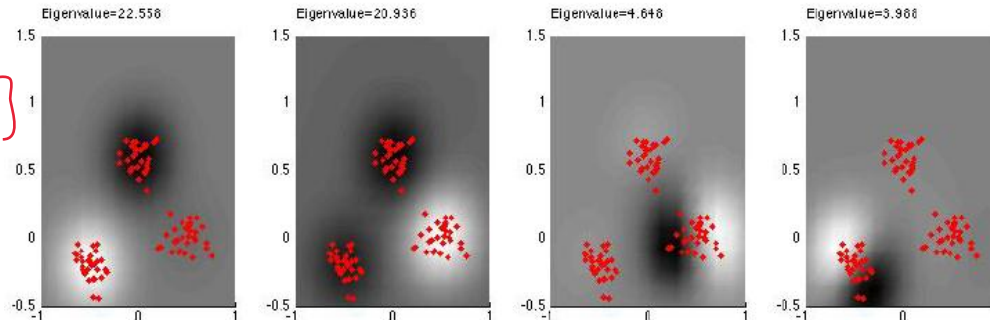
- Eigenvector evaluated at a test point x is a function

$$w^T \phi(x) = \sum_i \alpha_i \langle \phi(x^i), \phi(x) \rangle = \sum_i \alpha_i k(x^i, x)$$

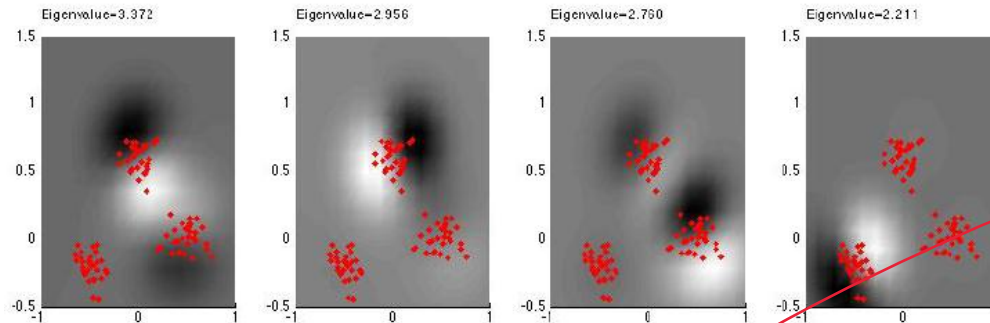
1st PC

$$V = [v_1, v_2, \dots, v_d]$$

$$V = X \cdot A$$



⇒ Prediction on x



$$V^T x = A^T \underbrace{X x}_{\downarrow} \\ = A^T k$$

$k \in \mathbb{R}^n$

$$k_i = k(x_i, x) = \langle \phi(x_i), \phi(x) \rangle, (i = 1, 2, \dots, n)$$

i -th training data

test data

What You Should Know

- Principal Component Analysis (PCA)
 - What PCA is, what is useful for.
 - Both the maximum variance subspace and the minimum reconstruction error viewpoint.
- Kernel PCA

Additional material on computing the principal components and ICA

Power method for computing PCs

Given matrix $X \in R^{D \times n}$, compute the top eigenvector of XX^T

Initialize with random $\hat{v} \in R^D$

Repeat

$$\hat{v} \leftarrow XX^T \hat{v}$$

$$\hat{v} \leftarrow \hat{v} / \|\hat{v}\|$$

Claim

For any $\epsilon > 0$, whp over choice of initial vector, after $O\left(\frac{1}{\epsilon} \log \frac{d}{\epsilon}\right)$ iterations, we have $\hat{v}^T XX^T \hat{v} \geq (1 - \epsilon)\lambda_1$.

Then can subtract the \hat{v} component off of each example and repeat to get the next.

Eigendecomposition

Any symmetric matrix $A = XX^T$ is guaranteed to have an eigendecomposition with real eigenvalues: $A = V \Lambda V^T$.

$$\begin{array}{c} \boxed{} \\ A \\ (D \times D) \end{array} = \begin{array}{c} \boxed{} \\ V \\ (D \times D) \end{array} \begin{array}{c} \boxed{\begin{array}{ccc} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \\ & & \dots \end{array}} \\ \Lambda \\ (D \times D) \end{array} \begin{array}{c} \boxed{} \\ V^T \\ (D \times D) \end{array} = \sum_i \lambda_i v_i v_i^T$$

Matrix Λ is diagonal with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots$ on the diagonal. Matrix V has the eigenvectors as the columns.

Singular Value Decomposition (SVD)

Eigendecomposition of XX^T is closely related to SVD of X .

Given a matrix $X \in \mathbb{R}^{D \times n}$, the SVD is a decomposition: $X^T = USV^T$

$$\begin{array}{c} \boxed{} \\ X^T \\ (n \times D) \end{array} = \begin{array}{c} \boxed{} \\ U \\ (n \times d) \end{array} \begin{array}{c} \boxed{\begin{array}{cc} \sigma_1 & 0 \\ 0 & \sigma_2 & \dots \end{array}} \\ S \\ (d \times d) \end{array} \begin{array}{c} \boxed{} \\ V^T \\ (d \times D) \end{array} = \sum_i \sigma_i u_i v_i^T$$

- S is a diagonal matrix with the singular values $\sigma_1, \dots, \sigma_d$ of X .
- Columns of U, V are orthogonal, unit length.
- So, $XX^T = VSU^TUSV^T = VS^2V^T =$ eigendecomposition of XX^T .

So, $\lambda_i = \sigma_i^2$ and can read off the solution from the SVD.

Singular Value Decomposition (SVD)

Eigendecomposition of XX^T is closely related to SVD of X .

Given a matrix $X \in \mathbb{R}^{D \times n}$, the SVD is a decomposition: $X^T = USV^T$

$$\begin{array}{c} \boxed{} \\ X^T \\ (n \times D) \end{array} = \begin{array}{c} \boxed{} \\ U \\ (n \times d) \end{array} \begin{array}{c} \boxed{\begin{array}{cc} \sigma_1 & 0 \\ 0 & \sigma_2 & \dots \end{array}} \\ S \\ (d \times d) \end{array} \begin{array}{c} \boxed{} \\ V^T \\ (d \times D) \end{array} = \sum_i \sigma_i u_i v_i^T$$

- In fact, can view the rows of US as the coordinates of each example along the axes given by the d eigenvectors.

So, $\lambda_i = \sigma_i^2$ and can read off the solution from the SVD.

Independent Component Analysis (ICA)

Find a linear transformation

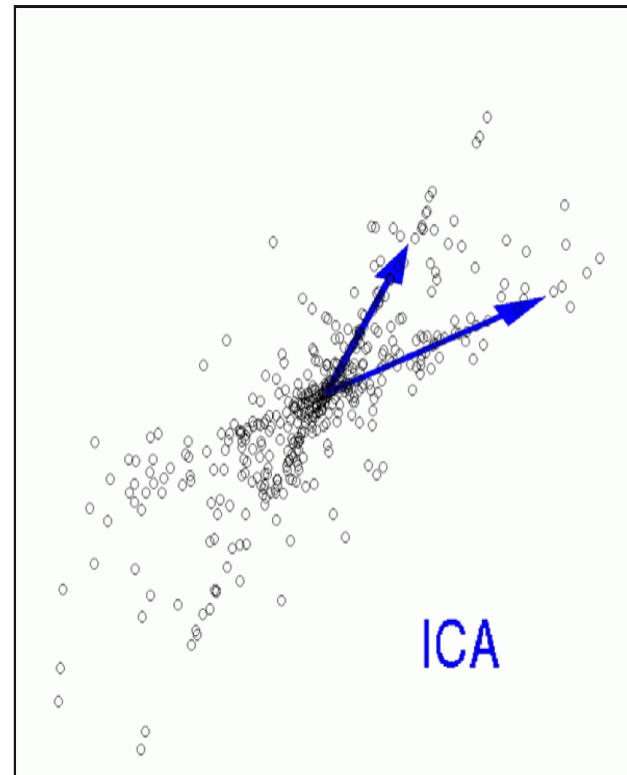
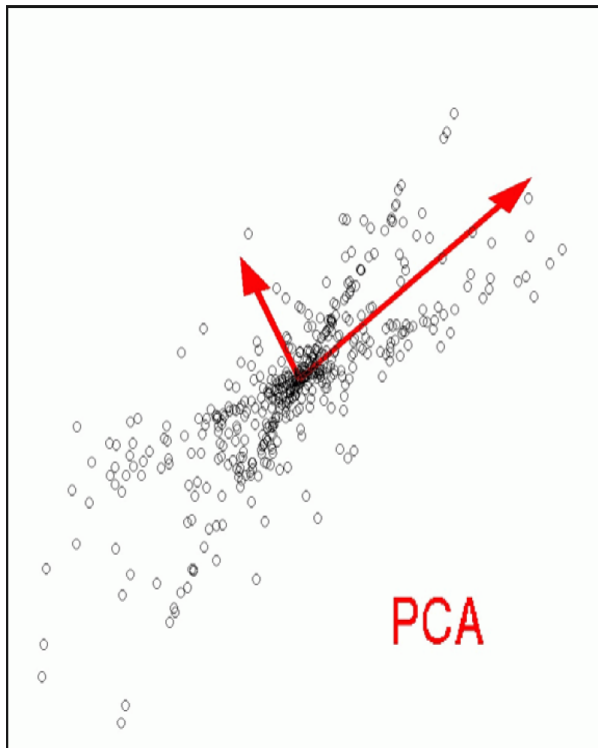
$$\mathbf{x} = \mathbf{V} \cdot \mathbf{s}$$

for which coefficients $\mathbf{s} = (s_1, s_2, \dots, s_D)^T$ are **statistically independent**

$$p(s_1, s_2, \dots, s_D) = p_1(s_1)p_2(s_2) \dots p_n(s_D)$$

Algorithmically, we need to identify matrix \mathbf{V} and coefficients \mathbf{s} , s.t. under the condition $\mathbf{x} = \mathbf{V}^T \cdot \mathbf{s}$ the **mutual information** between s_1, s_2, \dots, s_D is minimized:

$$I(s_1, s_2, \dots, s_D) = \sum_{i=1}^D H(s_i) - H(s_1, s_2, \dots, s_D)$$



PCA finds directions of maximum variation,
ICA would find directions most "aligned" with data.