Indian Institute of Information Technology, Allahabad





Introduction to Machine Learning

Principal Component Analysis

By

Dr. Shiv Ram Dubey
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

DISCLAIMER

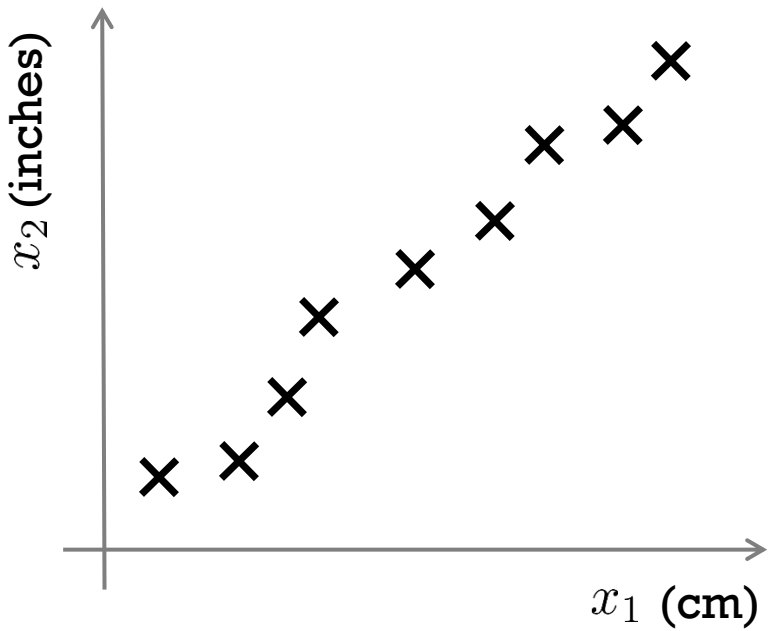
The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

Dimensionality Reduction

- **Goal:** Learn a mapping from $x \in \mathbb{R}^d$ to $x \in \mathbb{R}^{d'}$, with $d' \ll d$

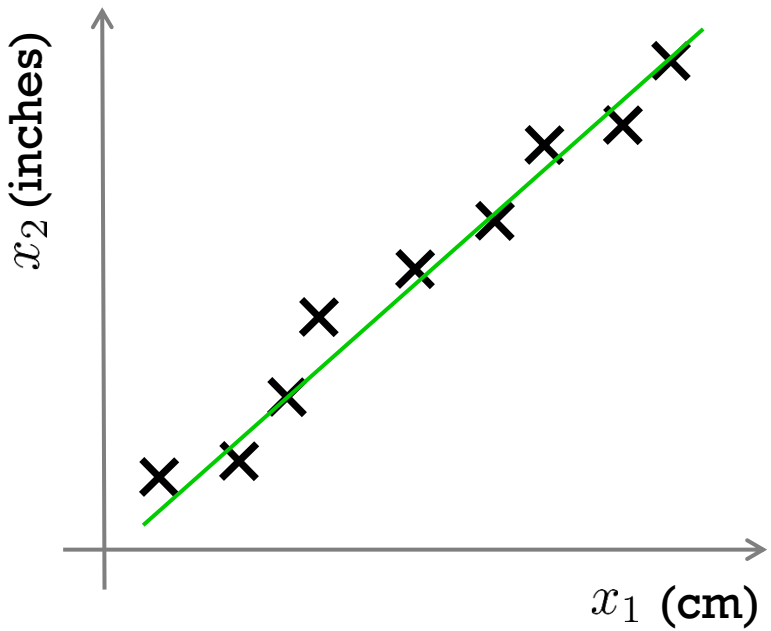
- We may want to reduce the number of features for several reasons:
  - Reduce the complexity of our learning problem
  - Remove colinear/correlated features
  - Visualize the features

Dimensionality Reduction: Data Compression



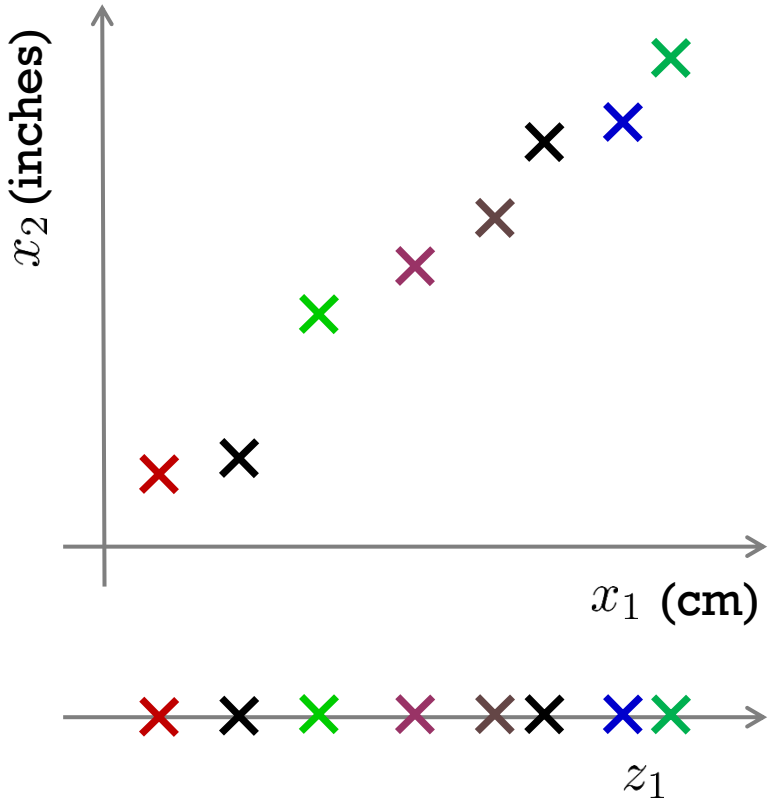
Reduce data from 2D to 1D

Dimensionality Reduction: Data Compression



Reduce data from 2D to 1D

Dimensionality Reduction: Data Compression



Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

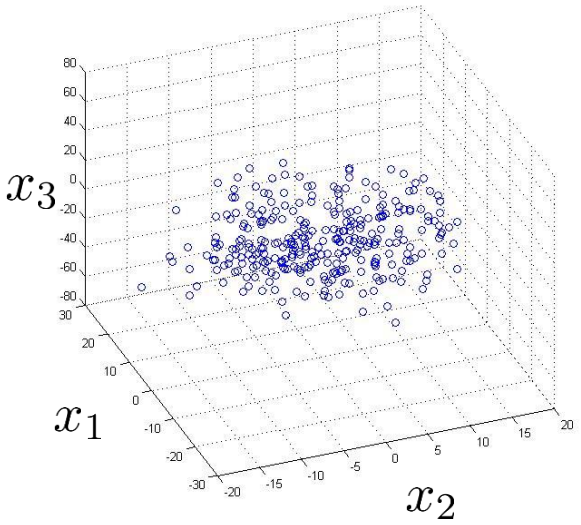
$$x^{(2)} \rightarrow z^{(2)}$$

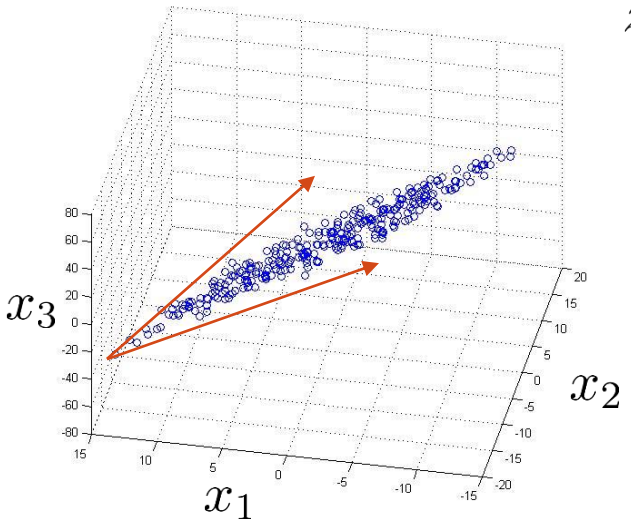
$$\vdots$$

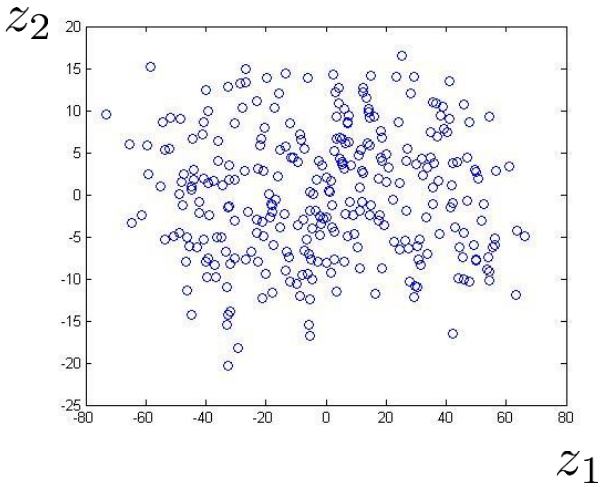
$$x^{(m)} \rightarrow z^{(m)}$$

Dimensionality Reduction: Data Compression

Reduce data from 3D to 2D







Dimensionality Reduction: Data Visualization

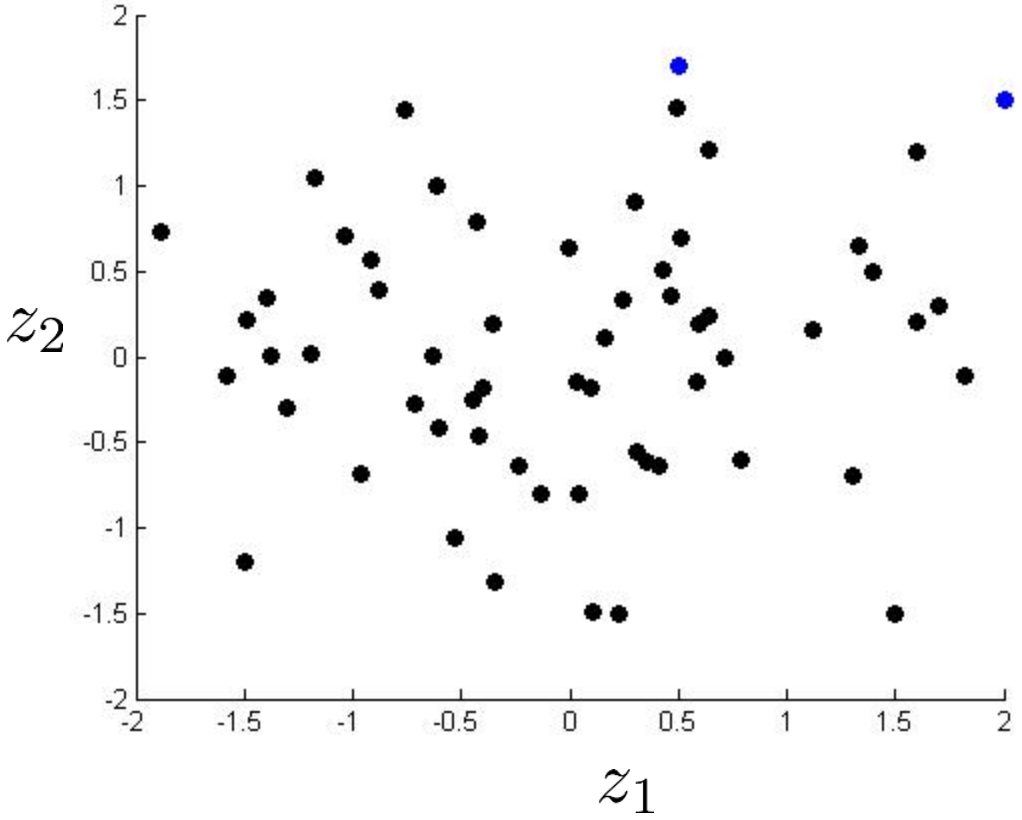
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

[resources from en.wikipedia.org]

Dimensionality Reduction: Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...	...	...

Dimensionality Reduction: Data Visualization



Dimensionality Reduction: Data Visualization

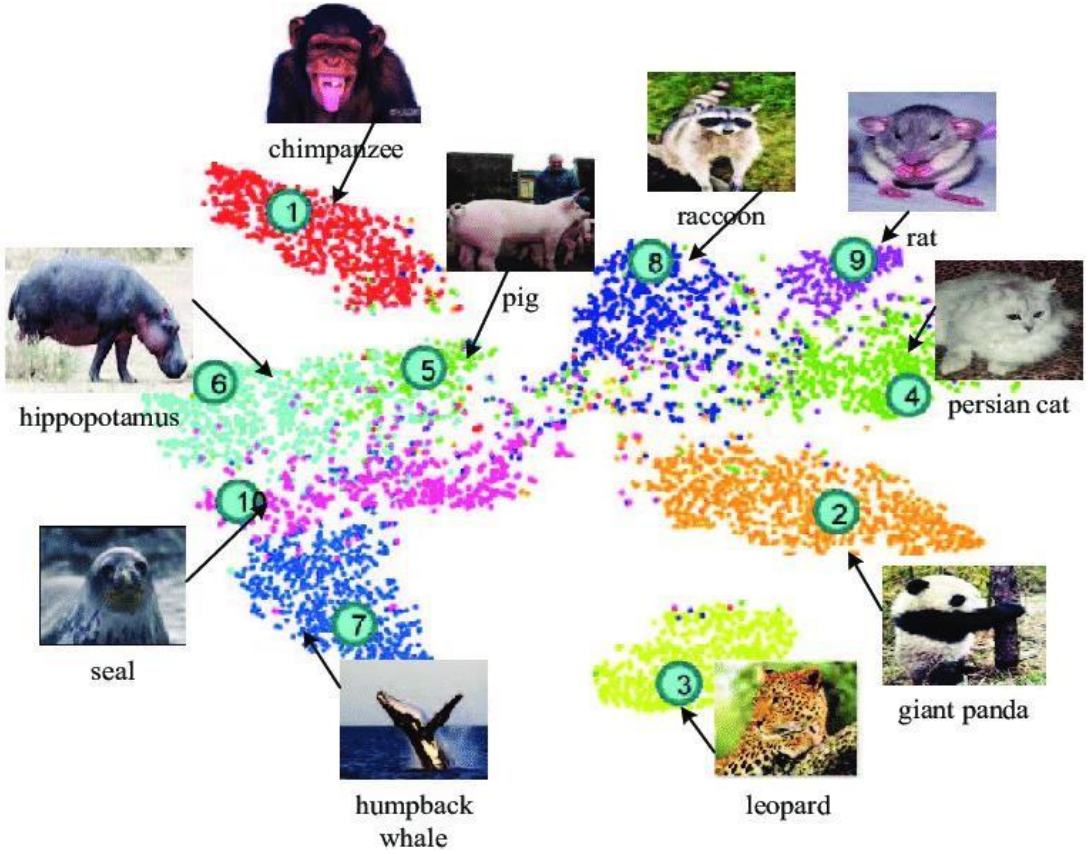
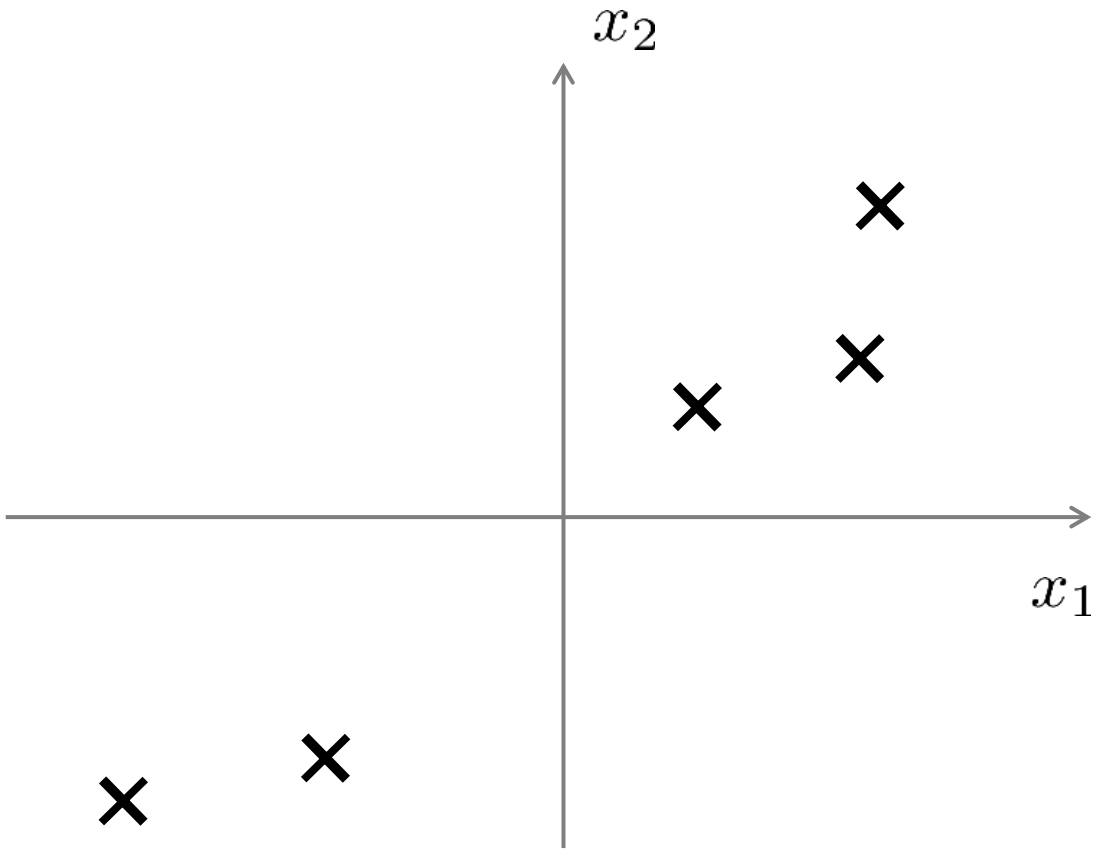
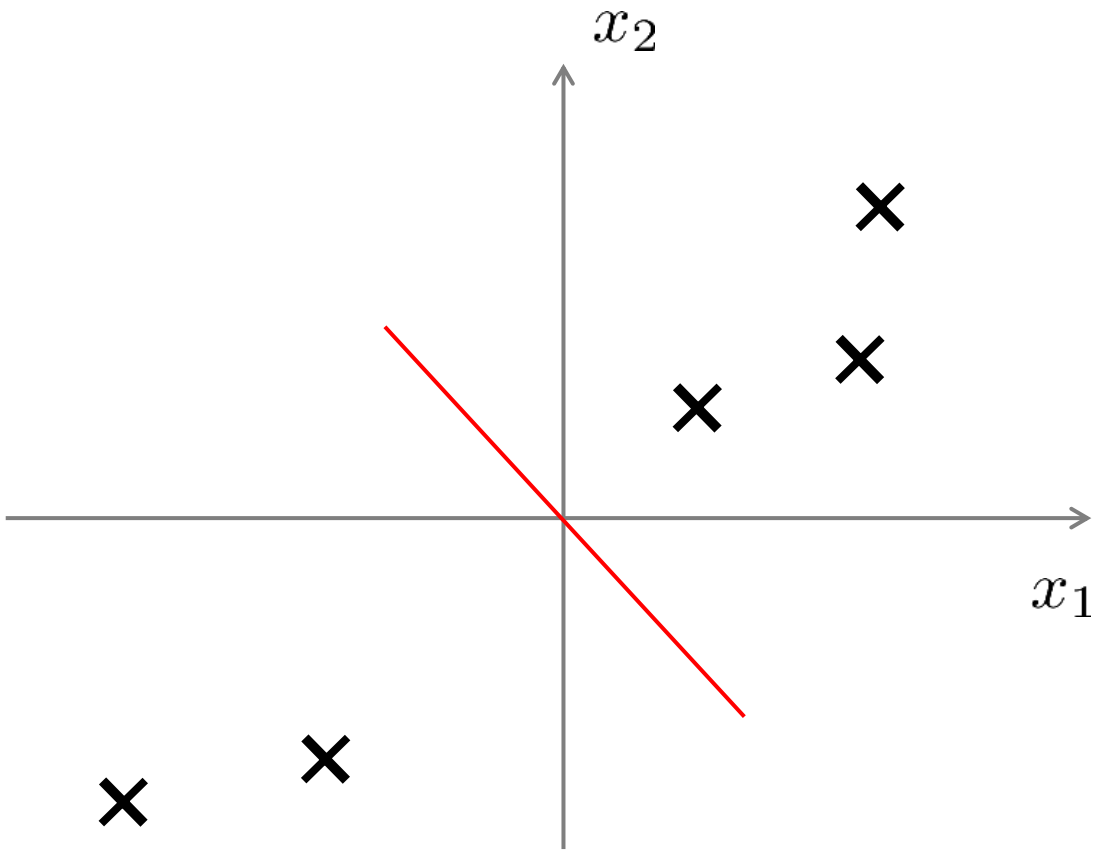


Image: https://arxiv.org/pdf/1703.08893.pdf

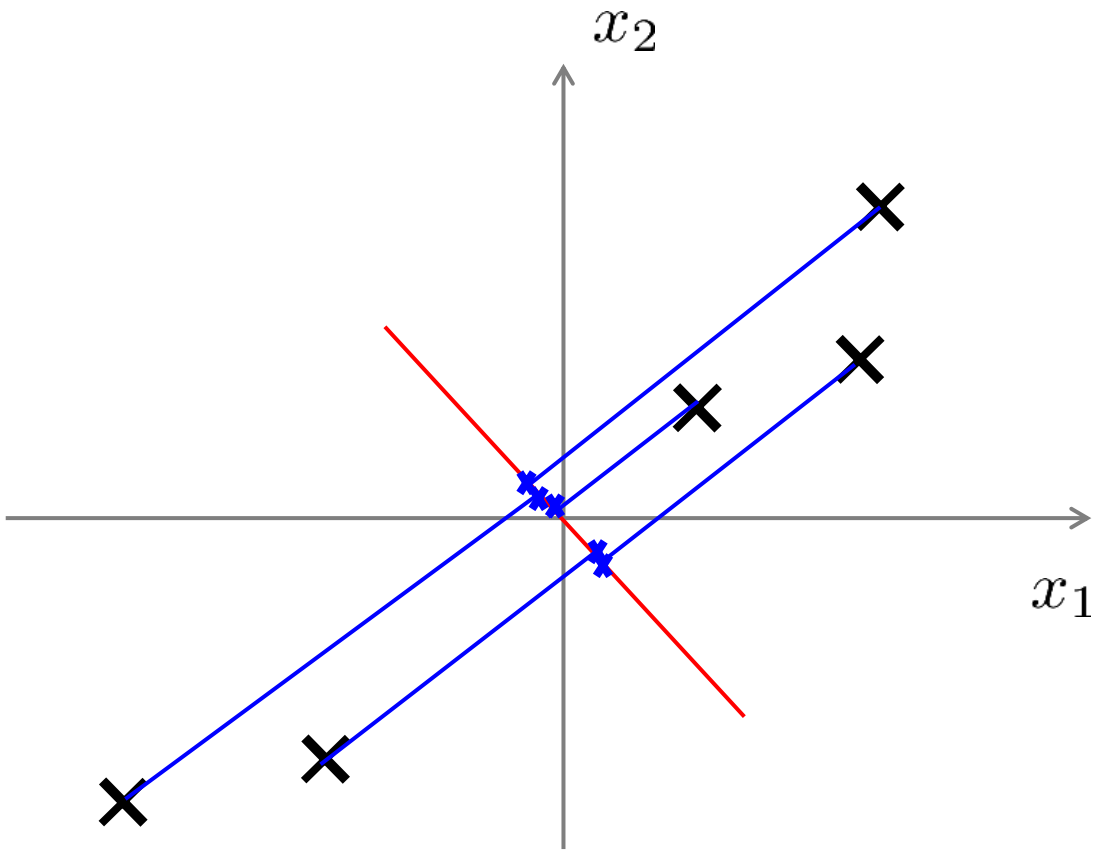
Principal Component Analysis: Problem Formulation



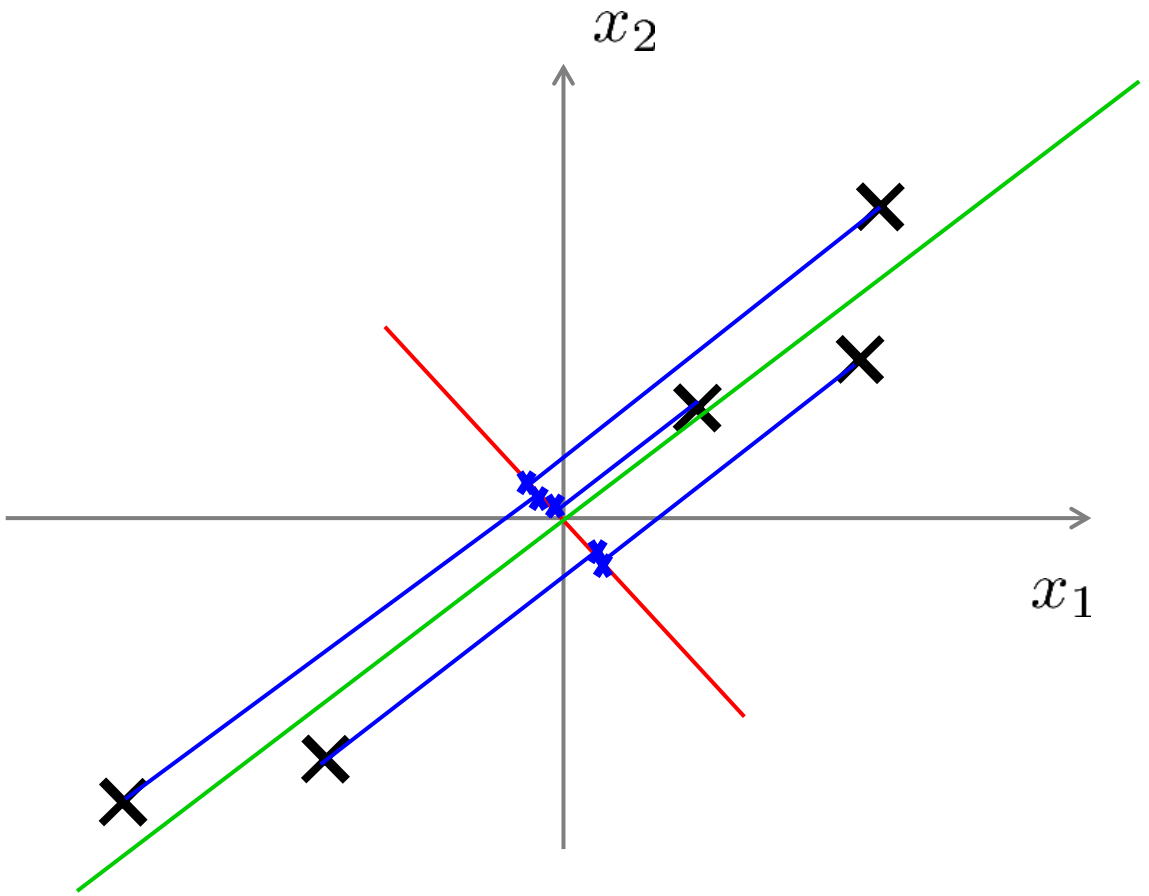
Principal Component Analysis: Problem Formulation



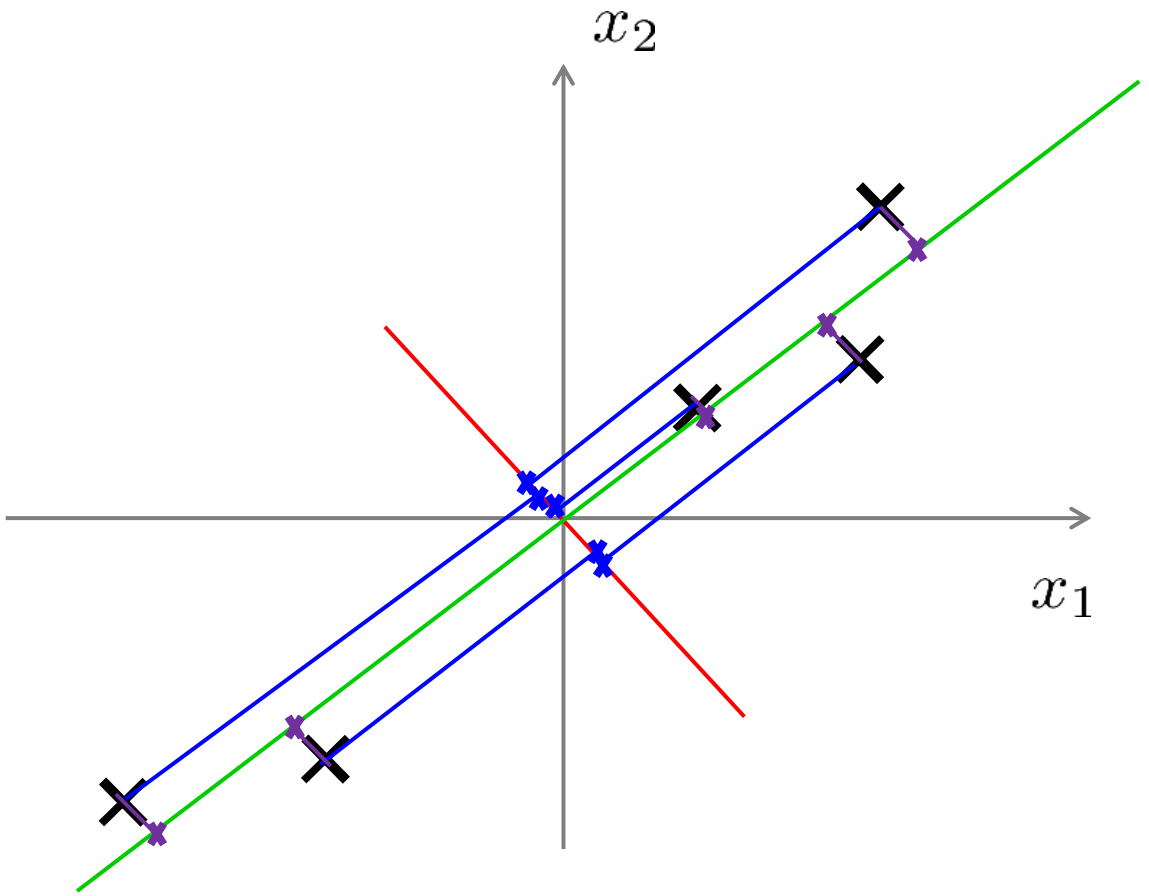
Principal Component Analysis: Problem Formulation



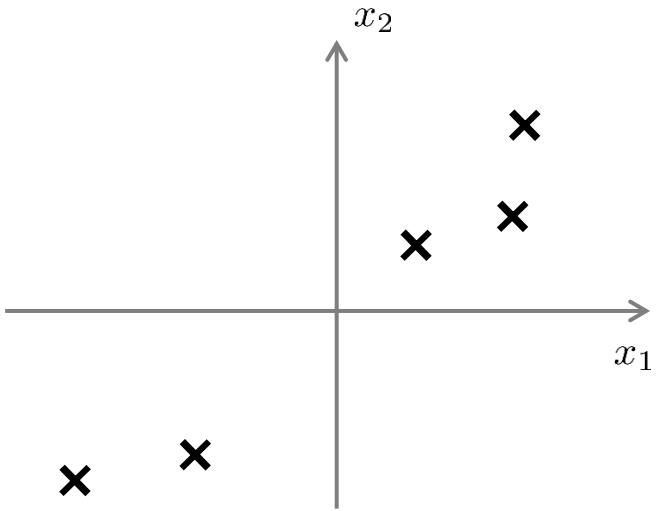
Principal Component Analysis: Problem Formulation



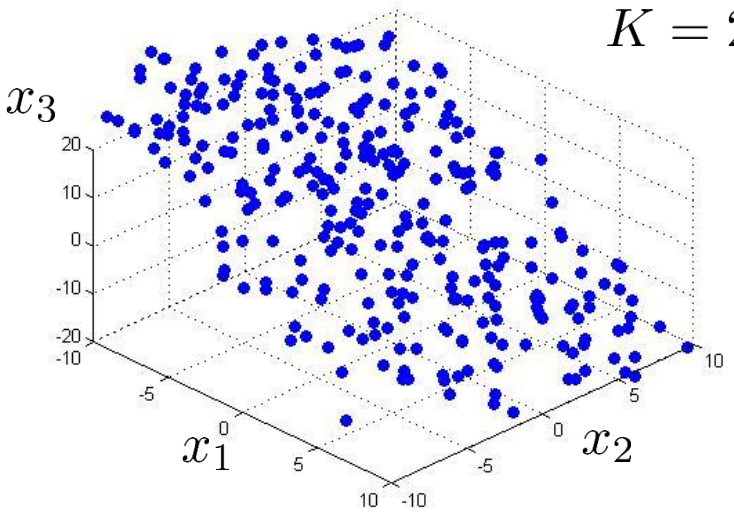
Principal Component Analysis: Problem Formulation



Principal Component Analysis: Problem Formulation



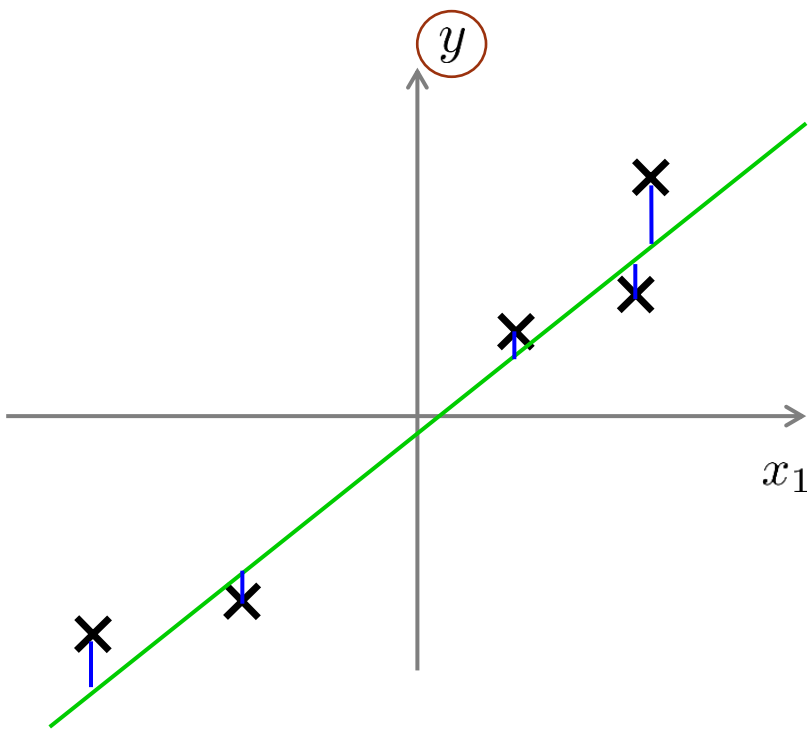
$3D \rightarrow 2D$
$K = 2$

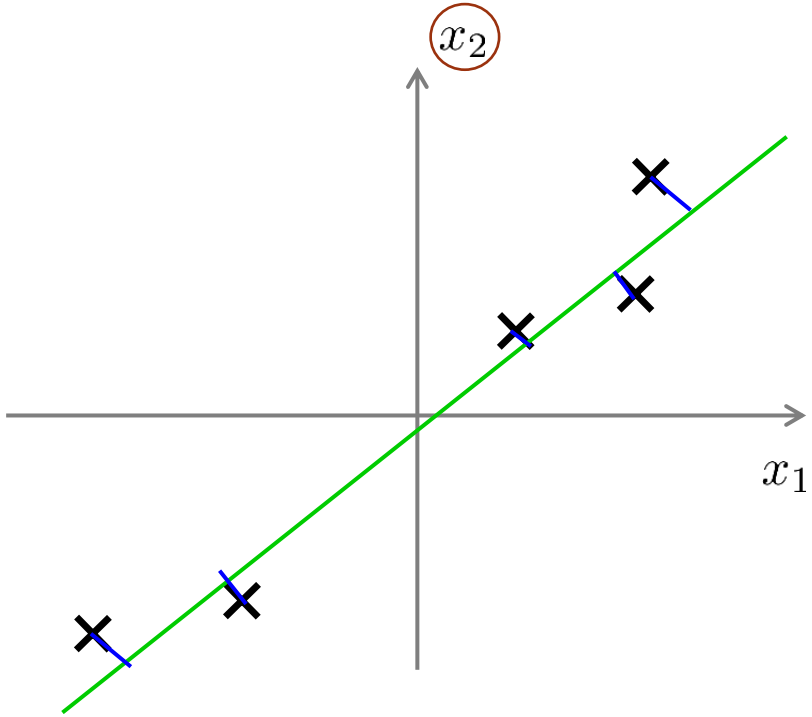


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k directions $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is not Linear Regression





Dimensionality Reduction

- We can write each input x as

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + x_d \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

projections original axes

- We aim to approximate x using a new basis $\{v_i\}_i$ (of unit norm):

$$x \approx \tilde{f}(x) = f(x)_1 v_1 + f(x)_2 v_2 + \cdots + f(x)_{d'} v_{d'}$$

Representation vs. Approximation

- We **approximate** x as follows:

$$x \approx \tilde{f}(x) = f(x)_1 v_1 + f(x)_2 v_2 + \cdots + f(x)_{d'} v_{d'} \in \mathbb{R}^d$$

- The corresponding **representation** is

$$f(x) = [f(x)_1 \quad f(x)_2 \quad \cdots \quad f(x)_{d'}] \in \mathbb{R}^{d'}$$

Dimensionality Reduction

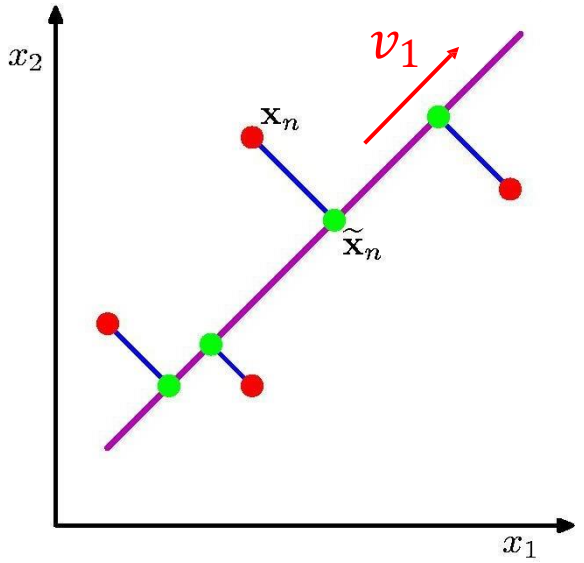
- **Loss function:** Minimize MSE of projected vectors

$$L(f; Z) = \frac{1}{n}\sum_{i=1}^{n} \left\| x_i - \tilde{f}(x_i) \right\|_2^2$$

1D Case

- **Simplest case:** If $d' = 1$, then we want $x \approx f(x)_1 v_1$

- Given v_1, we can take $f(x)_1 = x^\top v_1$
  - Minimizes $\|x - f(x)_1 v_1\|_2^2$
  - Then, we have $\tilde{f}(x) = (x^\top v_1) v_1$
  - i.e., orthogonal projection
  - Assuming $\|v_1\|_2 = 1$

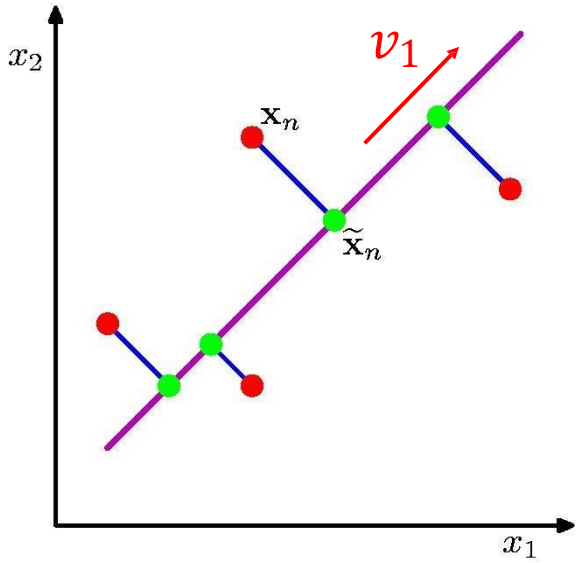


1D Case

- **Simplest case:** If $d' = 1$, then we want $x \approx f(x)_1 v_1$

- Given v_1, we can take $f(x)_1 = x^\top v_1$
  - Minimizes $\|x - f(x)_1 v_1\|_2^2$
  - Then, we have $\tilde{f}(x) = (x^\top v_1) v_1$
  - i.e., orthogonal projection
  - Assuming $\|v_1\|_2 = 1$

- How do we pick v_1 ?



1D Case

- In this case, the loss is

$$L(v_1; Z) = \frac{1}{n}\sum_{i=1}^{n} \left\| x_i - \left(x_i^\top v_1\right) v_1 \right\|_2^2$$

- Can be shown to be equivalent to maximizing variance:

$$L(v_1; Z) = -\mathrm{Var}\left(\left\{x_i^\top v_1\right\}_i\right)$$

- If variance of projection on v_1 is low, v_1 is not informative about x_i

1D Case

- Need a way to minimize $L(v_1; Z)$
- The **covariance matrix** is

$$C = \mathbb{E}[xx^\top] = \mathbb{E}\begin{bmatrix} x_1x_1 & \cdots & x_1x_d \\ \vdots & \ddots & \vdots \\ x_dx_1 & \cdots & x_dx_d \end{bmatrix}$$

- Given v_1, we have $\mathrm{Var}(x^\top v_1) = v_1^\top C v_1$
- Thus, $L(v_1; Z) = -\mathrm{Var}(x^\top v_1) = -v_1^\top C v_1$

1D Case

- The principal components analysis (PCA) algorithm computes

$$v_1^* = \min_{v_1} L(v_1; Z) = \max_{v_1} v_1^\top C v_1$$

- **Theorem:** Solution is $v_1^* = \text{TopEigenvector}\,(C)$
  - That is, eigenvector corresponding to the largest eigenvalue
  - **Recall:** If $Cv = \lambda v$, then v is an eigenvector corresponding to eigenvalue λ

1D Case

- In practice, use **empirical covariance matrix**

$$\hat{C} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n}\sum_{i=1}^{n} \begin{bmatrix} x_{i,1}x_{i,1} & \cdots & x_{i,1}x_{i,d} \\ \vdots & \ddots & \vdots \\ x_{i,d}x_{i,d} & \cdots & x_{i,d}x_{i,d} \end{bmatrix}$$

- **Algorithm:** Compute eigenvectors + eigenvalues of $\hat{C}$ and return the (unit) eigenvector corresponding to the largest eigenvalue
  - Sign of eigenvector doesn't matter

General Case

PCA(Z):

$Z \leftarrow \{ x - \text{Mean}(Z) \mid x \in Z \}$

$C \leftarrow \frac{1}{n} \sum_{i=1}^{n} x_i x_i^\top$

for $j \in \{1, \ldots, d'\}$:

$v_j \leftarrow \text{Eigenvector}(C, j)$

return $f : x \mapsto [x^\top v_1 \quad \cdots \quad x^\top v_{d'}]^\top$

General Case

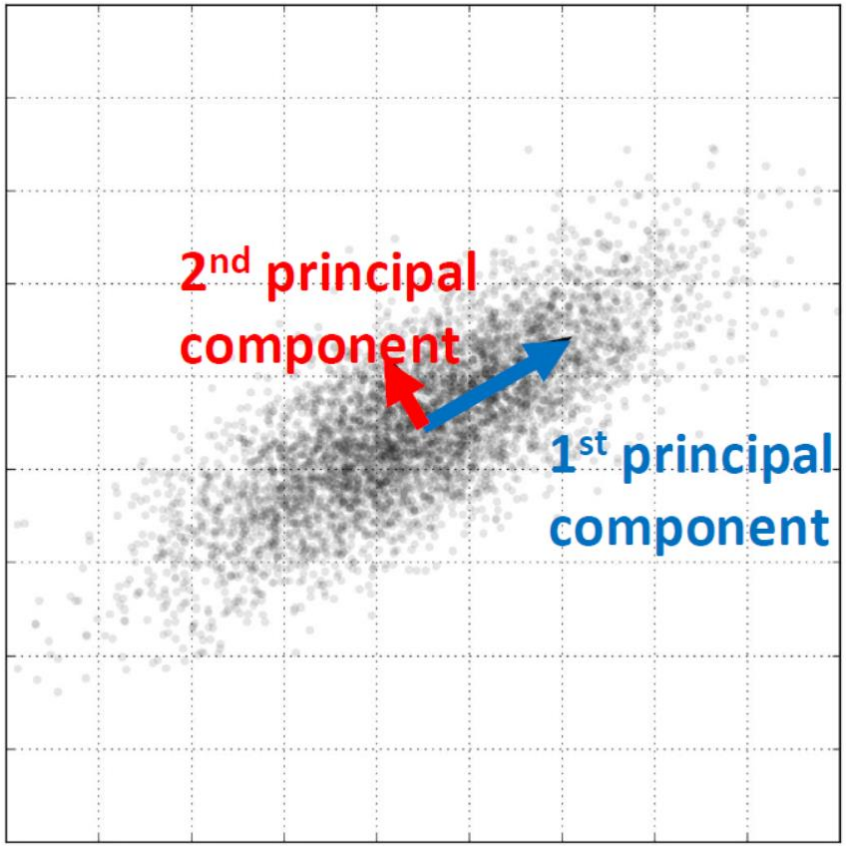
- Resulting function is

$$f(x) = \begin{bmatrix} x^\top v_1 \\ \vdots \\ x^\top v_{d'} \end{bmatrix} = \begin{bmatrix} v_1^\top \\ \vdots \\ v_{d'}^\top \end{bmatrix} x = Vx$$

PCA on a 2D Gaussian Dataset

- The vectors v_j are called **principal components**
  - Mutually orthogonal
  - Largest directions of variation

- Subtract mean to ensure vectors originate from the mean



By Nicoguaro - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=46871195

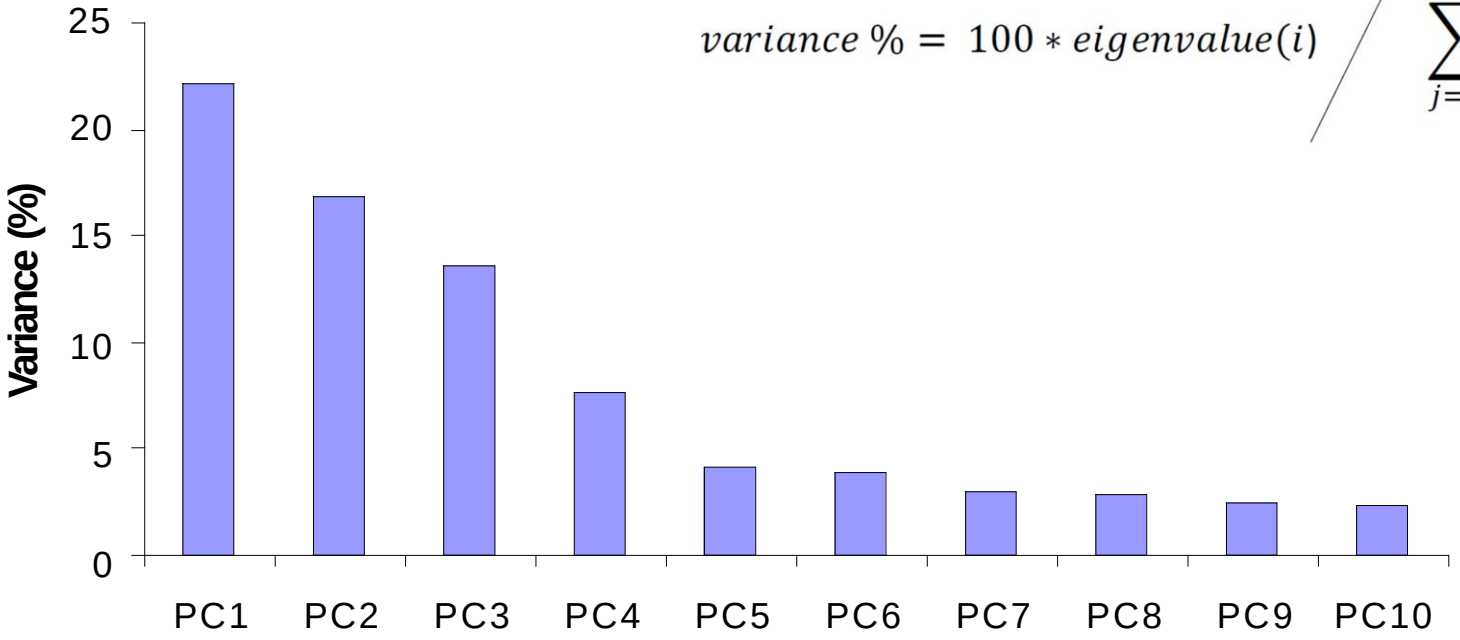
Dimensionality Reduction

- Taking $d' = d$ is just a change of basis
  - Linear regression does not change, but other algorithms may be affected

- Taking $d' \ll d$ reduce dimensionality of data while removing the smallest possible amount of information
  - In a linear sense

Dimensionality Reduction

$$variance\ \% = 100 * eigenvalue(i) \Big/ \sum_{j=1}^{d} eigenvalue(j)$$



Based on slide by Barnabás Póczos, UAlberta

PCA Algorithm using SVD

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$$

Compute "eigenvectors" of matrix Σ using Singular Value Decomposition:

```
[U,S,V] = svd(Sigma);
```

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

PCA Algorithm using SVD

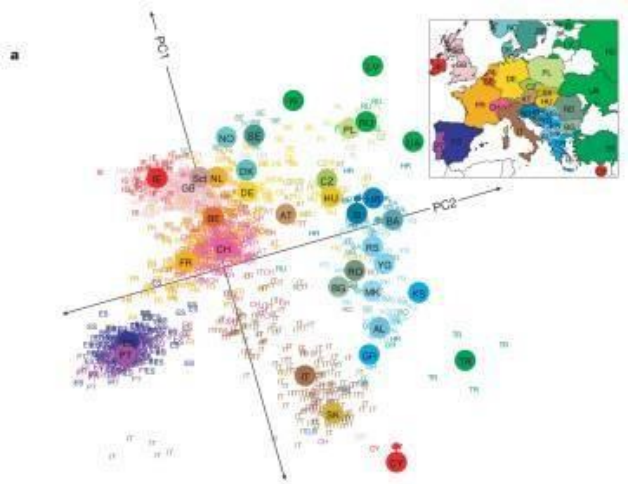
After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

Sigma = $\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$

```
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
z = Ureduce'*x;
```

Applications

- Can use $f(x)$ as the feature map
  - First examples of “learned features”
  - Form of regularization
  - Forms the basis for important modern deep learning algorithms
- Can be used to visualize high-dimensional data



Novembre et al., Genes mirror geography within Europe. Nature 2009.

Eigenfaces



(1000 64×64 images)

https://medium.com/data-science/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenfaces



mean face



https://medium.com/data-science/eigenfaces-recovering-humans-from-ghosts-17606c328184

Eigenface Projections



$d = 4096$



$d' = 1000$

Eigenface Projections



$d = 4096$



$d' = 250$

Eigenface Projections



$d = 4096$



$d' = 100$

Eigenface Projections



$d = 4096$



$d' = 50$

Nonlinear Dimensionality Reduction

MNIST Digit Dataset



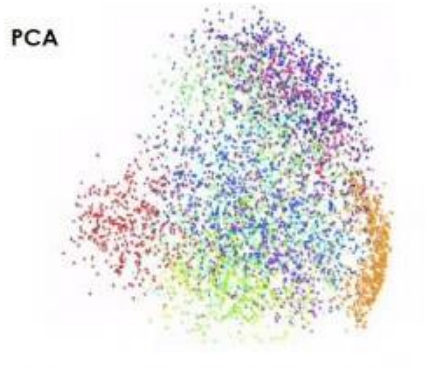


Fig: Laurens van der Maaten

Nonlinear Dimensionality Reduction

MNIST Digit Dataset



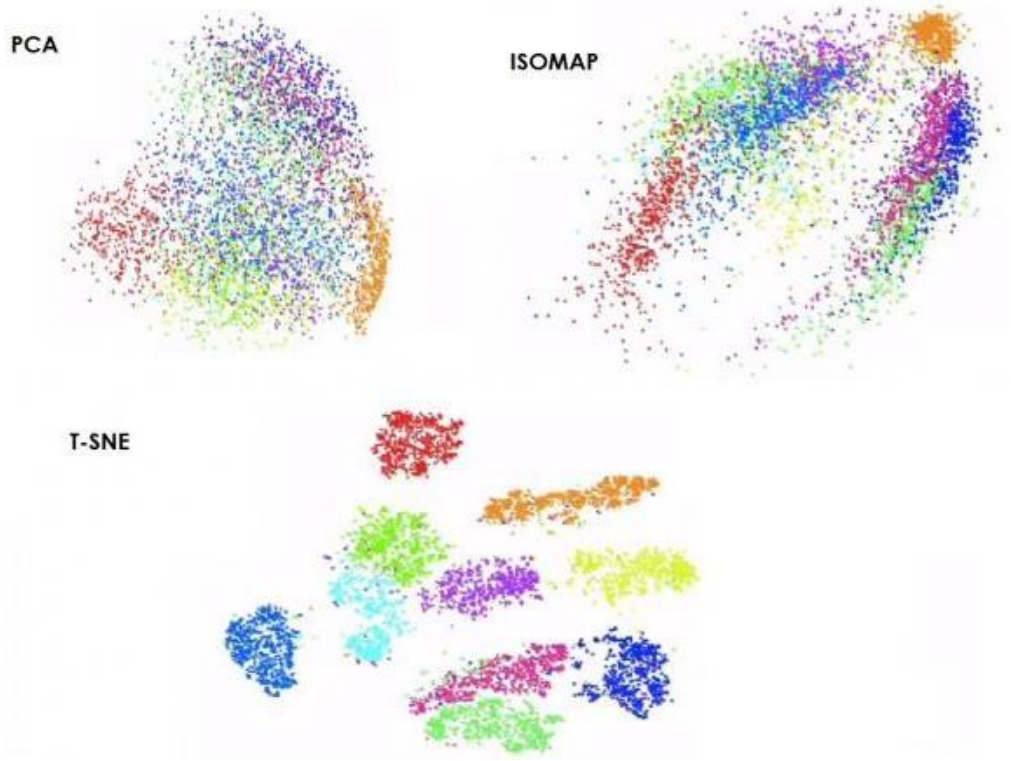


Fig: Laurens van der Maaten

Nonlinear Dimensionality Reduction

- **PCA benefits**
  - Projected representation of data can be approximate data in original space
  - Easy to optimize
  - No hyperparameters (except d')

- **Deep learning based approaches**
  - Nonlinear PCA is the basis of the autoencoder
  - Fundamental algorithm for feature learning that is still widely used

Acknowledgement

Most of the slides are based on the ML course of:

- Andrew Ng, Stanford University
- Mingmin Zhao and Jiatao Gu, University of Pennsylvania

THANK YOU