Machine Learning for Networking
ML4N

Luca Vassio
Gabriele Ciravegna
Zhihao Wang
Tailai Song

Recap – key concepts

- Random variables and data distributions
- Correlations
- Data types and properties
- Similarity and dissimilarity/distance metrics
- Data preprocessing
  - Data reduction (aggregation, sampling,...)
  - Feature transformation (nonlinear map function, one hot encoding,...)
  - Feature normalization

Recap - Feature Engineering

- A feature is a numeric representation of an aspect of raw data
- Feature engineering is the act of extracting features from raw data and transforming them
- Formats that are suitable for the machine learning model
- Feature learning: automate the choice of finding good features

Outline

- Dimensionality reduction problem
- Principal Component Analysis (PCA)
- Non-linear dimensionality reduction (Kernel-PCA, tSNE)

Matrix remarks

- If A and B are matrices whose sizes are such that the given operations are defined and c is any scalar then:

$$(A^t)^t = A$$

$$(A \pm B)^t = A^t \pm B^t$$

$$(cA)^t = cA^t$$

$$(AB)^t = B^t A^t$$

Variance - Single attribute (n=1)

- How much spread is in the data **x** along the (only) axis? (distance to the mean)
- Variance=Standard deviation^2

$$s = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2}$$

$$s^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2$$

Temperature (°C)
42
40
24
30
15
18
15
30
15
30
35

Covariance - Two attributes (n=2)

- Covariance: measures the correlation between **x** and **y**

$$cov(\mathbf{x}, \mathbf{y}) = \frac{1}{m-1} \sum_{i=1}^{m} (x_i - \bar{x})(y_i - \bar{y})$$

- cov(**x,y**)=0: independent
- cov(**x,y**)>0: move same direction
- cov(**x,y**)<0: move opposite directions

x=Temperature	y=Humidity
40	90
40	90
40	90
30	90
15	70
15	70
15	70
30	90
15	70
30	70
30	70
30	90
40	70
30	90

Covariance matrix

- Contains covariance values between all possible dimensions (n attributes)
- n=3 in the example below:

$$S = \begin{bmatrix} cov(\mathbf{x},\mathbf{x}) & cov(\mathbf{x},\mathbf{y}) & cov(\mathbf{x},\mathbf{z}) \\ cov(\mathbf{y},\mathbf{x}) & cov(\mathbf{y},\mathbf{y}) & cov(\mathbf{y},\mathbf{z}) \\ cov(\mathbf{z},\mathbf{x}) & cov(\mathbf{z},\mathbf{y}) & cov(\mathbf{z},\mathbf{z}) \end{bmatrix}$$

Covariance matrix

- Contains covariance values between all possible dimensions (n attributes)
- Sample covariance matrix S (n x n) of X (m x n) can be written as:

$$S = \frac{1}{m-1}(X - \overline{X})^t(X - \overline{X})$$

- Where $\overline{X}$ is the matrix (m x n) of the mean attributes (all m rows are equal, corresponding to the mean of all m rows of X)

Orthogonality/Orthonormality

- Two vectors $\mathbf{u_1}$ and $\mathbf{u_2}$ (n x 1) for which $\mathbf{u_1}^t\mathbf{u_2}=0$ holds are said to be **orthogonal**
- Unit vectors ($\mathbf{u_1}^t\mathbf{u_1}=1$) which are orthogonal are said to be **orthonormal**
- The inverse of an **orthogonal matrix** (columns and rows are orthogonal vectors) is its transpose (due to its definition)

Eigenvalues and eigenvectors

- Vectors **u** having same direction as A**u** are called eigenvectors of A (A is an n by n matrix)
- In the equation $A\boldsymbol{u}=\lambda\boldsymbol{u}$, λ is called an eigenvalue of A

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} = 4 \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Eigenvalues and eigenvectors

- $A\mathbf{u}=\lambda\mathbf{u} \Leftrightarrow (A-\lambda I)\mathbf{u}=0$
- How to calculate **u** and λ:
  - Calculate det(A-λI), yields a polynomial (degree n)
  - Determine roots to det(A-λI)=0, roots are eigenvalues λ
  - Solve (A- λI)**u**=0 for each λ to obtain eigenvectors **u**

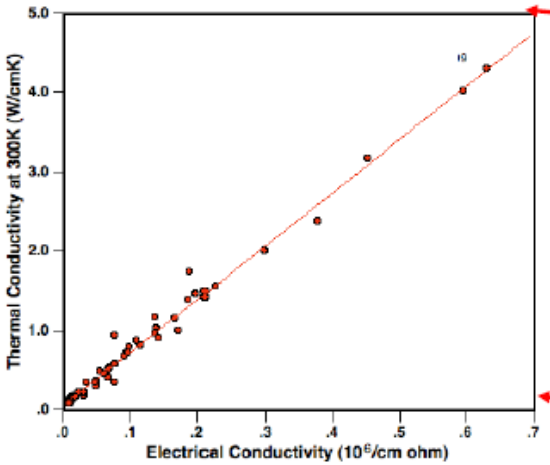
Dimensionality reduction

Visualization problem

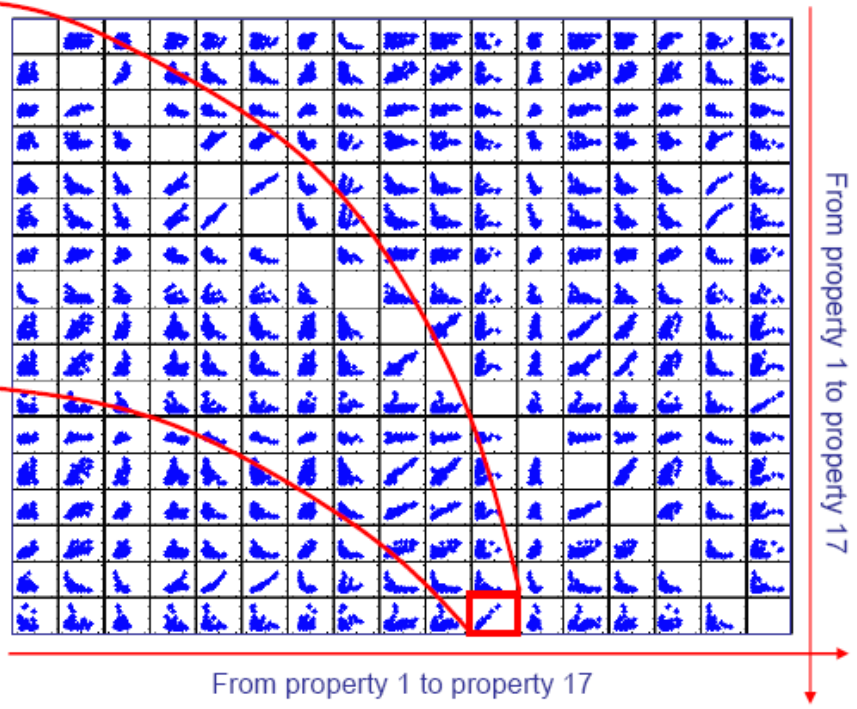
- Not easy to visualize multivariate data
  - 1D: dot
  - 2D: bivariate plot (i.e. X-Y plane)
  - 3D: X-Y-Z plot
  - 4D: ternary plot with a color code
  - 5D, 6D, etc. : ???

Problems with many features

- Not easy to extract useful information from multivariate data
- Bivariate plots represent correlations between couples of variables



Some relationships can be found from the bivariate plots of raw data
Ex. **Wiedemann-Franz Law**



Dimensionality reduction

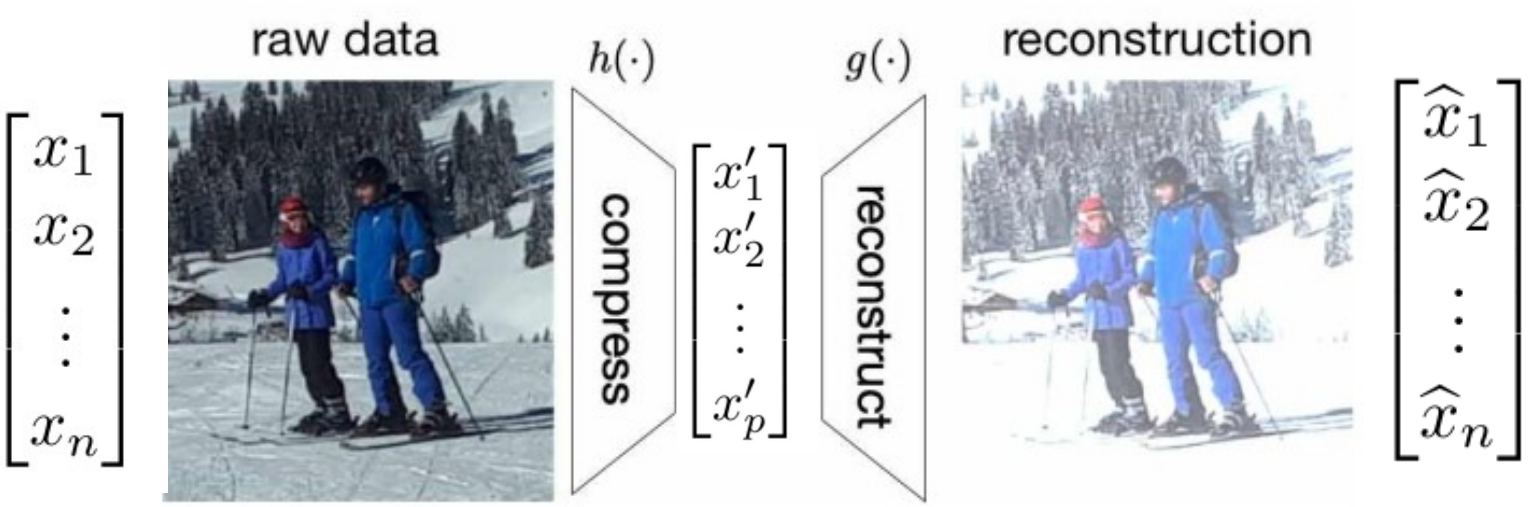
- Solving problems by changing the viewpoint
- Finding good features automatically
- Create a compressed representation of data
- Learn hypothesis map that reads representation of data point and transforms it to a set of features

Dimensionality reduction

- Feature engineering/feature learning also means feature reduction
  - Why reduce the number of features?
    - Capture the important information in a data set more efficiently than the original attributes
    - Allow data to be more easily visualized
    - Reduce overfitting – i.e., improve the generalization of models
    - Avoid curse of dimensionality
    - Reduce amount of time and memory required by algorithms
    - Help to eliminate irrelevant features or reduce noise

Dimensionality reduction

- Finding a map h which maximally compresses the raw data while still allowing to accurately reconstruct the original datapoint from a small number of features
- From n raw features to p features (p<n)
- Find a compressed representation



Dimensionality reduction

Techniques

- Singular Value Decomposition
- **Principal component analysis (PCA)**
- Linear Discriminant Analysis (LDA)
- **Non linear techniques (tSNE)**
- **…**

Principal Component Analysis

PCA

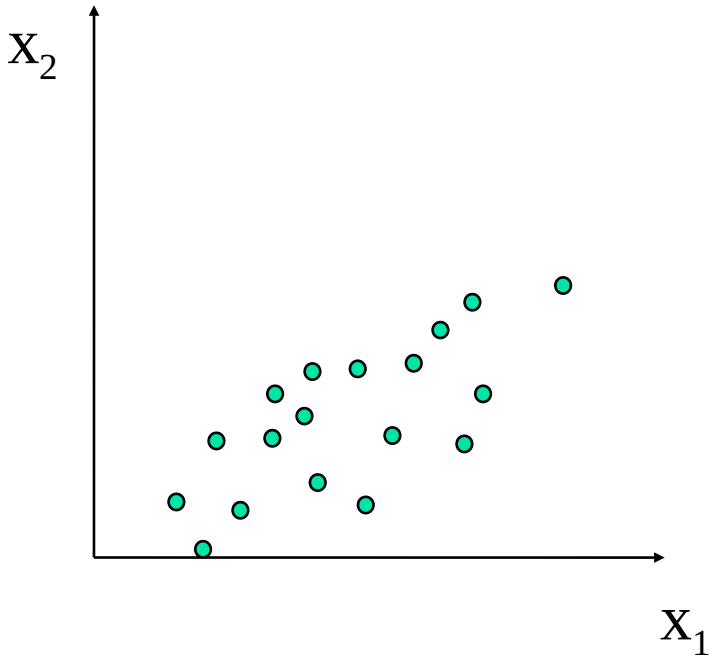
- Can be used to:
  - Reduce number of dimensions in data
  - Find patterns in high-dimensional data
  - Visualize data of high dimensionality

Idea of PCA

- Introduced by Pearson (1901) and Hotelling (1933) to describe the variation in a set of multivariate data in terms of a set of uncorrelated variables
- We typically have a data matrix **X** of m observations on n correlated variables $x_1, x_2, \ldots, x_n$
- A new basis such that the **first component maximizes information of data**
- For PCA that means: **captures variance in the data**
- PCA looks for a transformation of the x_i into p (up to n) new variables x'_i that are **uncorrelated**

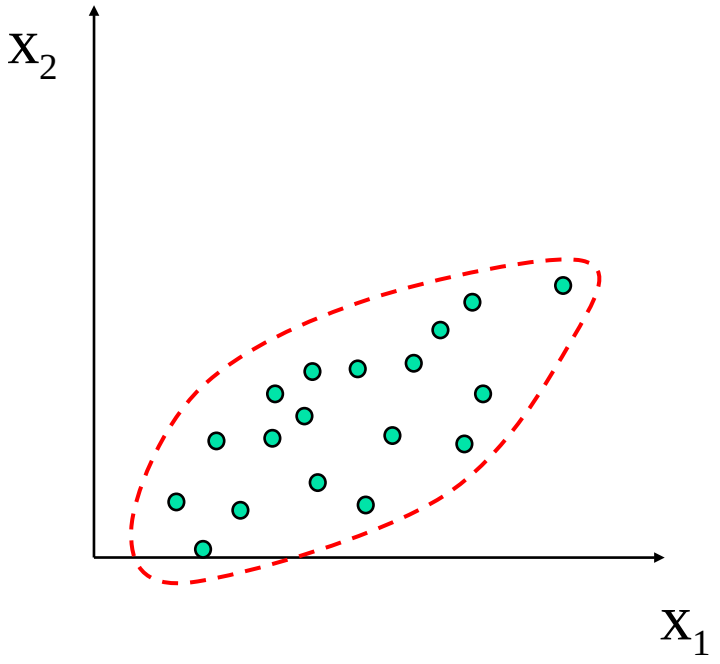
PCA

Find a projection that captures the largest amount of variation in data



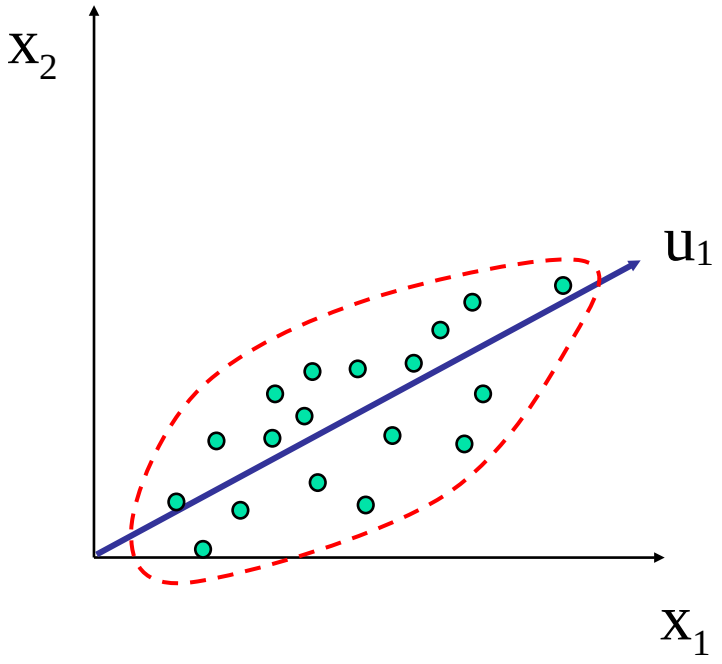
PCA

Find a projection that captures the largest amount of variation in data



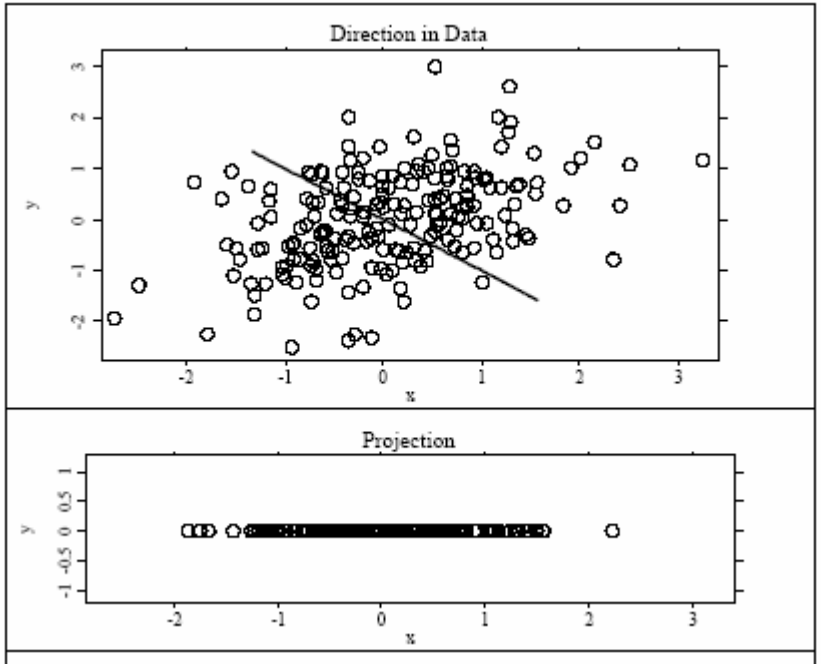
PCA

Find a projection that captures the largest amount of variation in data



PCA

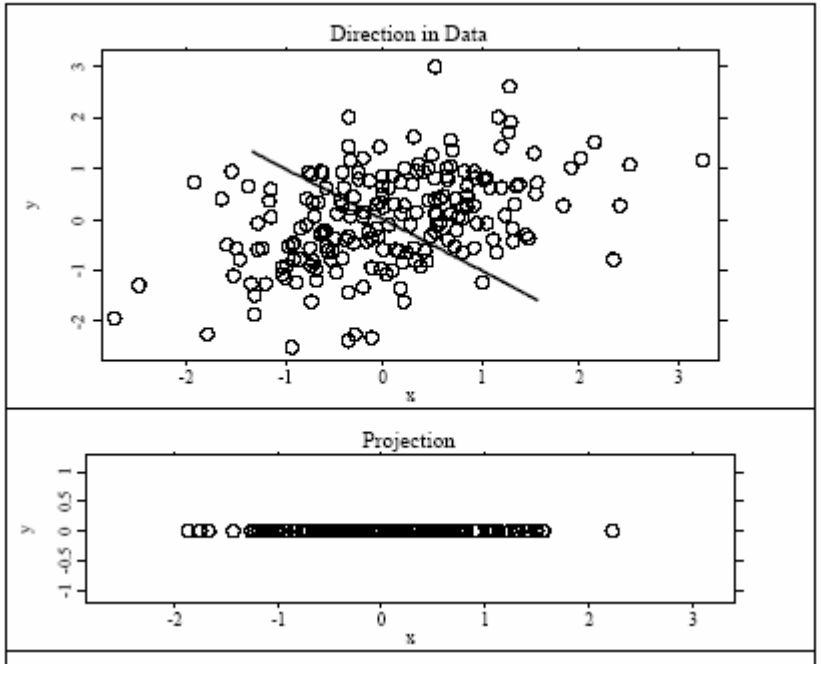
Can we intuitively see the projection with the largest variation?

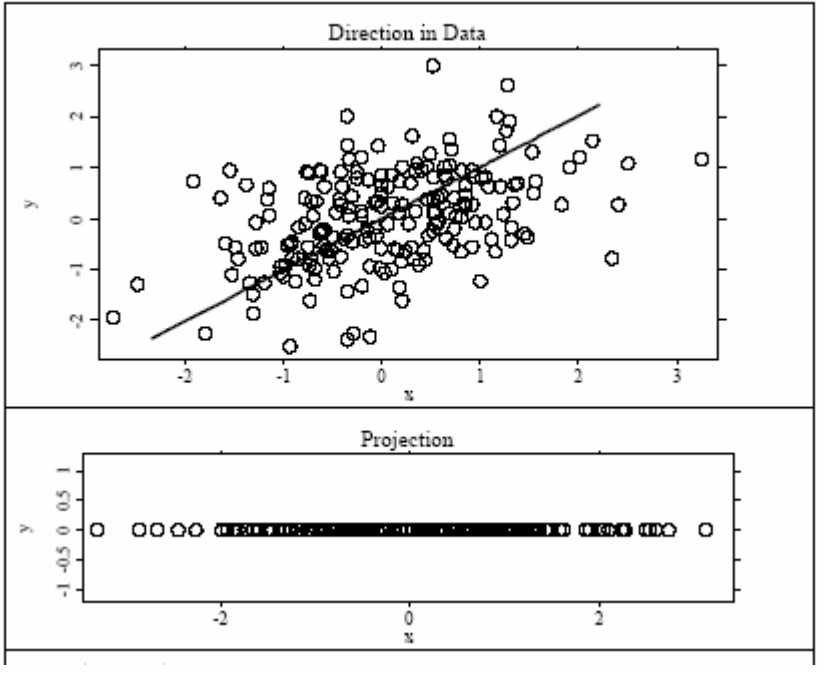


PCA

Can we intuitively see the projection with the largest variation?

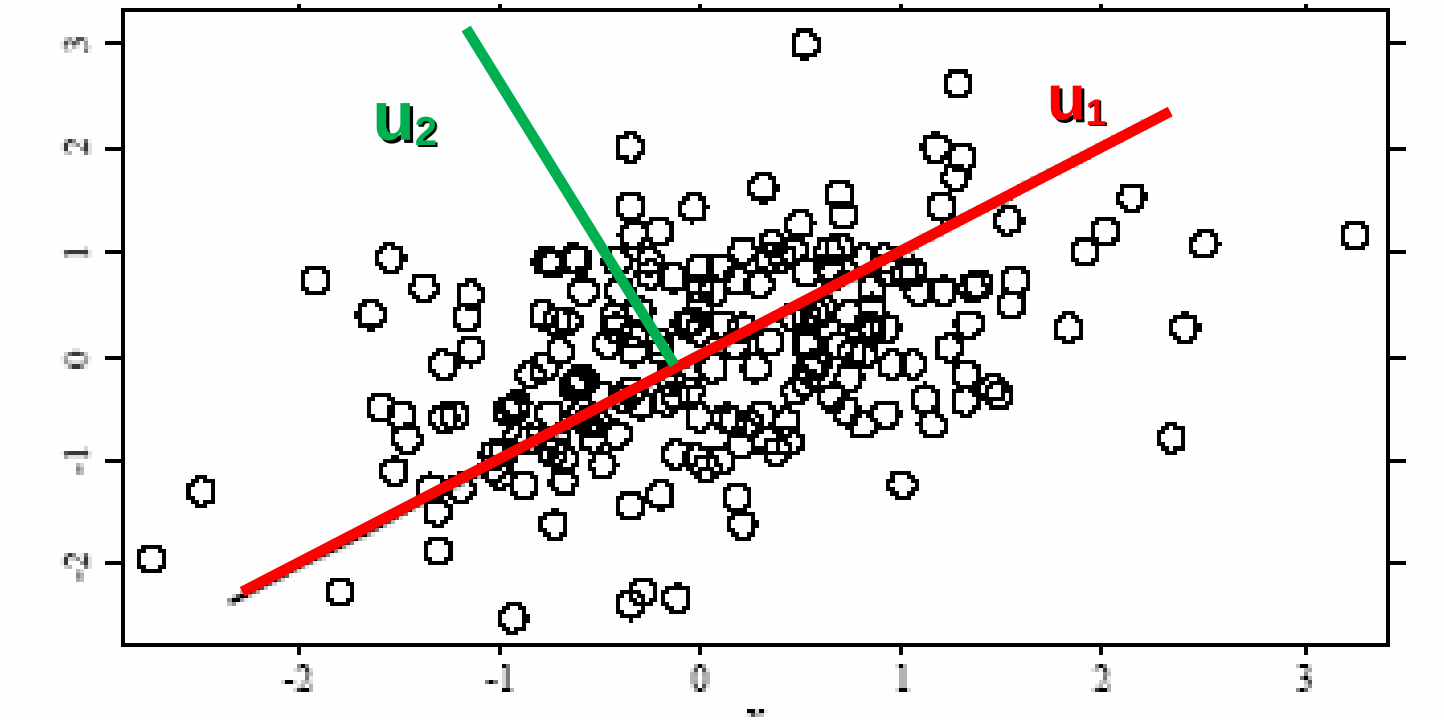
Better !





PCA

Orthogonality



PCA

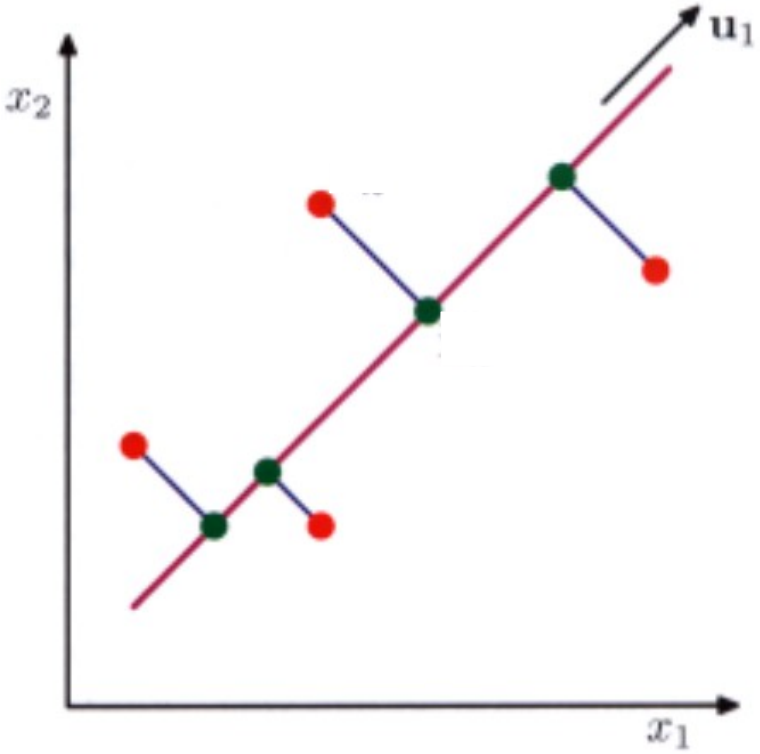
- It can be viewed as a **rotation of the existing axes** to new positions in the space defined by original variables
- New axes are **orthogonal** and represent the directions with maximum variability

PCA

- Principal component analysis is useful if there is some **redundancy** in variables or the **data do not ‘span’ the whole of n dimensional space**
- Redundancy means that some of the variables are **correlated** with one another
- Because of this redundancy or space not covered, it is possible to reduce the observed variables into a **smaller number of principal components** (artificial variables)
- The components account for **most of the variance** in the observed variables

Two definitions of PCA

Principal component analysis seeks a space of lower dimensionality, known as the principal subspace and denoted by the magenta line, such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots). An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors, indicated by the blue lines.



C.M. Bishop – Pattern recognition and machine learning - Springer

Principal components

- A principal component can be defined as a **linear combination** of optimally-weighted observed variables
- The number of components that can be extracted in a principal component analysis is equal to the number of observed variables (n)
- Often only the **first few components account for meaningful amounts of variance** (p<n)
- This means that the first component will be correlated with some/many of the observed variables.
- When the analysis is complete, the **resulting components** will display varying degrees of correlation with the observed variables, but are **completely uncorrelated with one another**

PCA: orthonormal change of basis P

- The feature transformation is a **change of the basis with orthonormal vectors**
- Find **orthonormal P** (n x n) such that $X' = XP^t$
- With matrix **S' = cov(X') diagonalized (variables uncorrelated)**
- The (first p) rows of P are **the principal components of X**

How to find P with cov(X') diagonalized

$$cov(X') = \frac{1}{m-1}(X')^t(X')$$

if X' has zero mean

$$= \frac{1}{m-1}(XP^t)^t(XP^t)$$

$$= \frac{1}{m-1}PX^tXP^t$$

$$= \frac{1}{m-1}P(X^tX)P^t = \frac{1}{m-1}PAP^t$$

How to find P with cov(X') diagonalized

- $A=X^tX$ is symmetric (n x n)
- Therefore there is a matrix E of eigenvectors of A and a diagonal matrix D such that: $A = EDE^t$
- Now define P to be the transpose of the matrix E of eigenvectors: $P := E^t$
- Then we can write A as: $A = P^t DP$

How to find P with cov(X') diagonalized

- The inverse of an orthogonal matrix is its transpose (due to its definition): $P^{-1} = P^t$

$$cov(X') = \frac{1}{m-1} PAP^t$$
$$= \frac{1}{m-1} PP^t DPP^t$$
$$= \frac{1}{m-1} PP^{-1} DPP^{-}1 = \frac{1}{m-1} D$$

How to find P with cov(X') diagonalized

- P diagonalizes cov(X')
- Where P is the transpose of the matrix of eigenvectors of $A=XX^t$
- The principal components of X are the eigenvectors of $A=X^tX$ (the rows of P)
- The i_{th} diagonal value of cov(X') is the variance of X' along u_i (along the i_{th} principal component, the i_{th} row of P)

PCA

- Remark: $\frac{1}{m-1}X^tX$ is the covariance matrix of X, if X has zero mean
- Essentially, we need to take the covariance matrix of the original matrix X and compute:
  - Eigenvalues: explained variance
  - Eigenvectors: new axis, principal components

Principal components

Principal component 1 (PC1)

- The eigenvalue with the largest absolute value will indicate that the data have the largest variance along its eigenvector, the direction along which there is greatest variation

Principal component 2 (PC2)

- the direction with maximum variation left in data, orthogonal to the PC1, i.e, the eigenvector corresponding to the second largest eigenvalue

Steps of PCA

- Let $\overline{X}$ be the mean matrix (all rows are equal, corresponding to the mean of all rows)
- Adjust the original data by the mean $X - \overline{X}$
- Compute the covariance matrix S of X : $S = \frac{1}{m-1}(X - \overline{X})^t(X - \overline{X})$
- Find the eigenvectors and eigenvalues of S
- For matrix S, vectors **u** (=column vector) having same direction as S**u**, with factor **λ**

Steps of PCA

- Eigenvalues λ_i corresponds to variance on each component i=1,…,n
- Thus, sort by λ_i
- Take the first p eigenvectors $\mathbf{u}_i$; where p is the number of top eigenvalues. Form matrix P
- These are the directions with the largest variance
- Project the original data: $X'=XP^t$

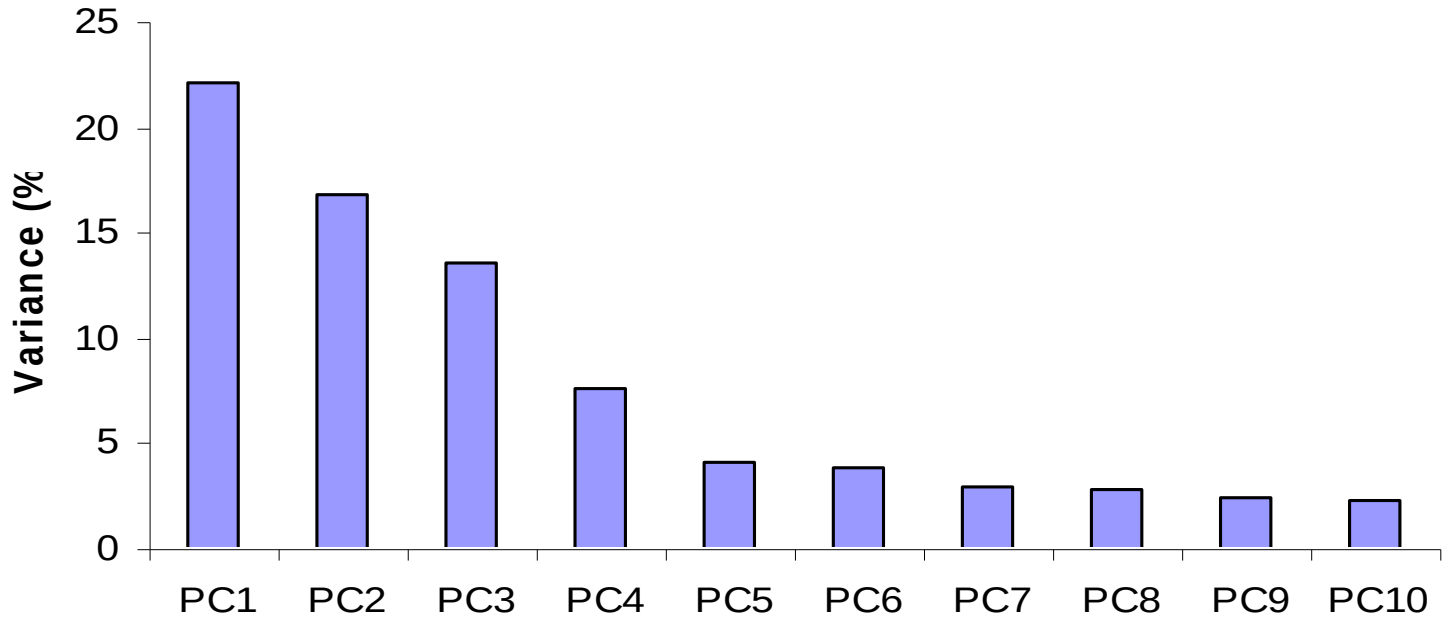
Eigenvalues and Variance

- Eigenvalues λ_i are used for calculation of [% of total variance] (V_i) for each component i:

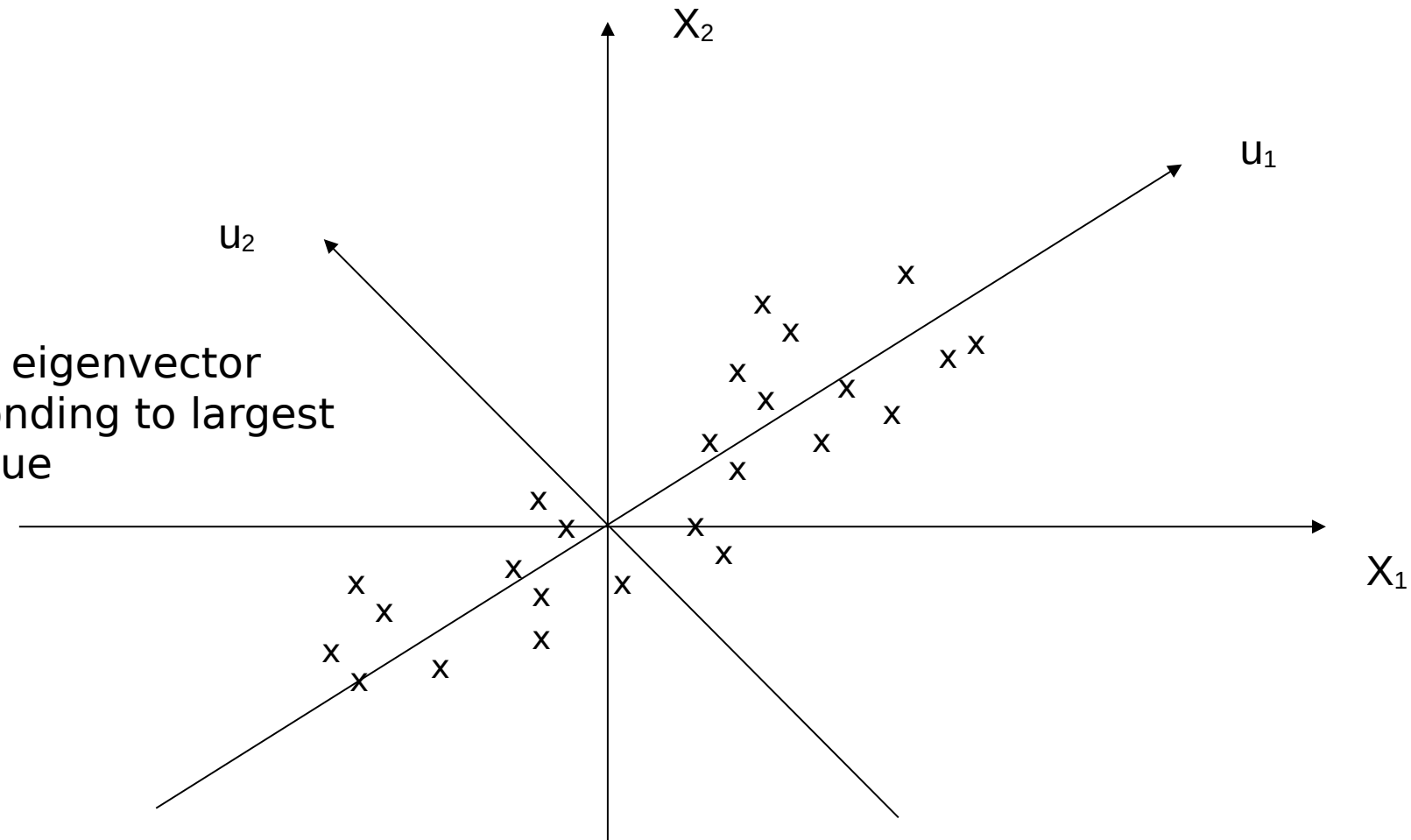
$$V_i = 100 \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

$$\sum_{i=1}^{n} \lambda_i = n$$ (if data standardized)

Eigenvalues and Variance



New basis



u_1 is the eigenvector corresponding to largest eigenvalue

Principal components

- Properties
  - linear combinations of the original variables
  - uncorrelated with each other
  - capture as much of the original variance as possible

Eigenvalue maximize variance

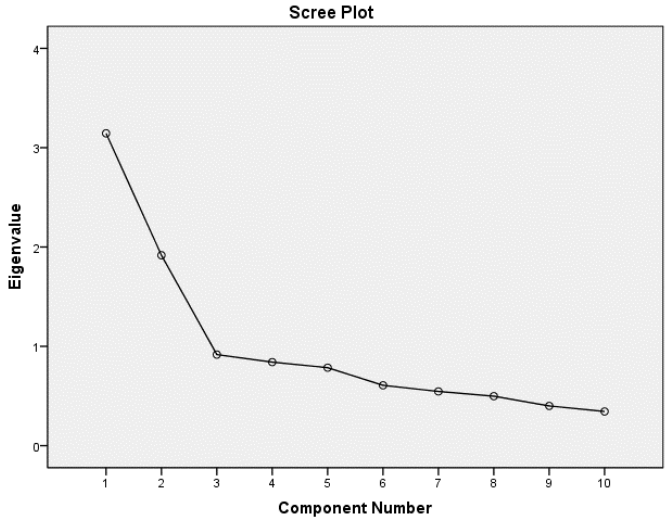
- Sketch of proof
  - p=1
  - Maximize variance of projected data
  - Constrained it to normal vector
  - Lagrange multiplier
  - Projector must be an eigenvectors of S
  - Max variance corresponds to the max eigenvalue
  - Proceed by induction

Remarks

- If you multiply one variable by a scalar you get different results
- This is because it uses covariance matrix (and not Pearson correlation)
- PCA should be applied on data that have approximately the same scale in each variable
- We can standardize, so that the new variables all have the same variance → each variable have the same weight

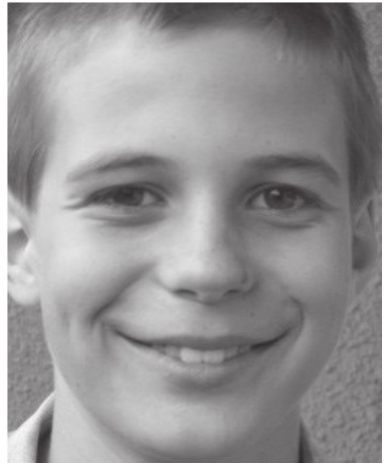
When to stop (choose p)?

- Kaiser criterion: keep PCs with eigenvalues >1 (variance explained more than a single original variable)
- Proportion of variance explained: enough PCs to have cumulative variance explained that is larger than a threshold (e.g., 66%)
- Elbow method: start of the bend in the line (point of inflexion) indicate how many components are retained



PCA example

- Image compression using PCA
  - 321x261 pixels
- An image can be seen as a vector
  - n=83781 pixel vector



PCA example

- 32 images, i.e., objects

PCA example

- We have only 32 observations and 83781 unknowns in our example!
- PCA is still applicable
- The eigenvectors we derive are called eigen-images, after rearranging back from the 1D vector to a rectangular image

PCA example

- Let us perform the dimensionality reduction from 83781 to 4 in our example
- $X' = XP^t$
- X (32 x 83781)
- P (4 x 83781)
- X’ (32 x 4)

PCA example

- Eigenvectors $\mathbf{u}_i$, i = 1,...,4




PCA example

- Reconstruction of the image from four basis vectors $\mathbf{u}_i$, i = 1,…,4 which can be displayed as images by rearranging the (long) vector back to the matrix form
- The mean value of images subtracted when data were normalized earlier has to be added

PCA example

- Reconstruction

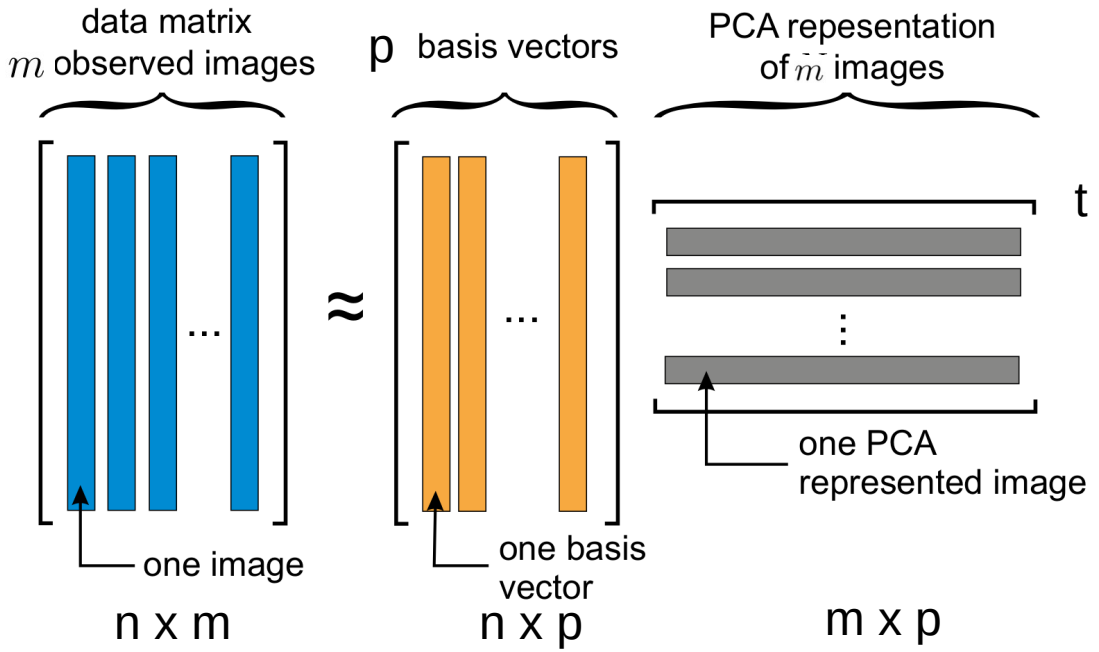
$$X' = XP^t \longrightarrow X \approx \hat{X} = X'P$$

PCA example

- Reconstruction

$$X \approx \hat{X} = X' P$$

$$X^t \approx \widehat{X^t} = P^t X'^t$$



PCA example

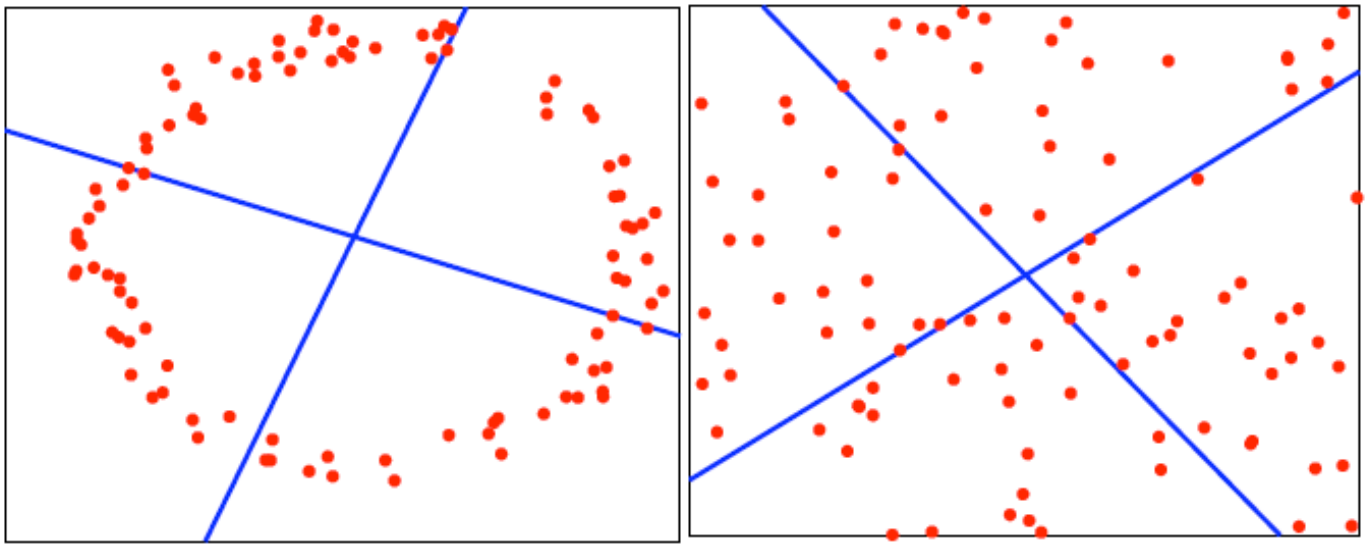
- 4 components, 32 reconstructed images

PCA example

- Comparison: original vs. 4 components

Non-linear dimensionality reduction

Extensions of PCA



- PCA will make no difference between these two examples
- We need non-linear axes

Extensions of PCA – Kernel PCA

- Instead of using directly points **x** as they are, we can go to some different feature space f(**x**)
- E.g., polar coordinates instead of cartesian coordinates
- In this new space, we can do PCA
- Result will be non-linear in the original space

Extensions of PCA – Kernel PCA

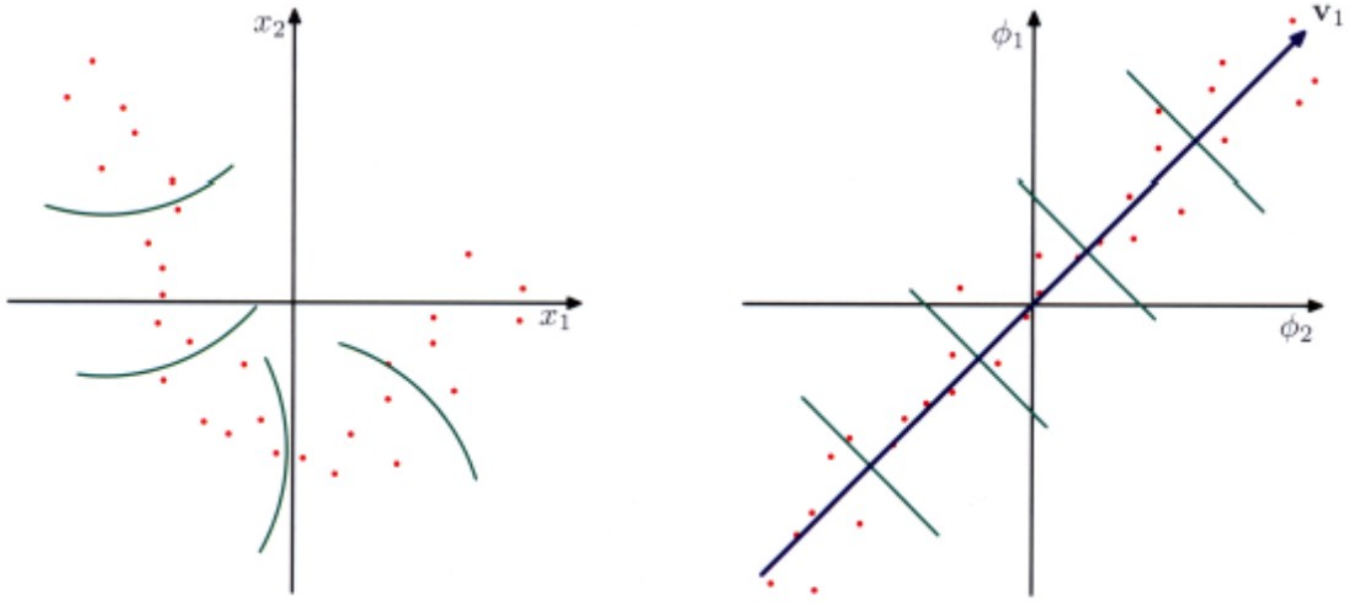


Figure 12.16 Schematic illustration of kernel PCA. A data set in the original data space (left-hand plot) is projected by a nonlinear transformation $\phi(\mathbf{x})$ into a feature space (right-hand plot). By performing PCA in the feature space, we obtain the principal components, of which the first is shown in blue and is denoted by the vector $\mathbf{v}_1$. The green lines in feature space indicate the linear projections onto the first principal component, which correspond to nonlinear projections in the original data space. Note that in general it is not possible to represent the nonlinear principal component by a vector in $\mathbf{x}$ space.

C.M. Bishop – Pattern recognition and machine learning - Springer

Extensions of PCA – Kernel PCA

- Works on the projection f of the data in the new feature space
- f is such that the kernel $K(x,y)=f(x)^t f(y)$ does not have negative eigenvalues
- We can work directly with kernel matrix K instead of performing the projection (to save computation)

t-SNE

- t-distributed stochastic neighbor embedding (t-SNE)
- **Nonlinear dimensionality reduction** technique for embedding high-dimensional data for visualization
- **Similar objects** are modeled by **nearby points** and **dissimilar objects** are modeled by **distant points** with high probability

t-SNE

- Two iterative stages:
  1. Constructs a **probability distribution over pairs of high-dimensional** objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability
  2. Defines a **similar probability distribution over the points in the low-dimensional** map
- Perform them iteratively until **Kullback–Leibler divergence between the two distributions is minimized**

https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

t-SNE

- t-SNE **shrinks widespread data and expands densely packed data**
- t-SNE is often able to recover **well-separated clusters**, and with special parameter choices, approximates a simple form of spectral clustering
- The visual clusters can be **influenced** strongly by the chosen parameterization. Some of the clusters might be **false finding**

t-SNE - Complexity

- For a data set with m elements, t-SNE runs in $O(m^2)$ time and requires $O(m^2)$ space

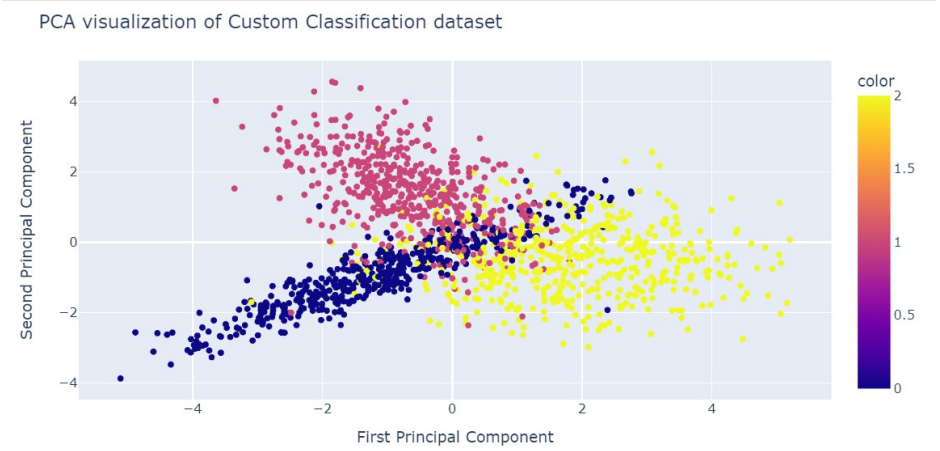
https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

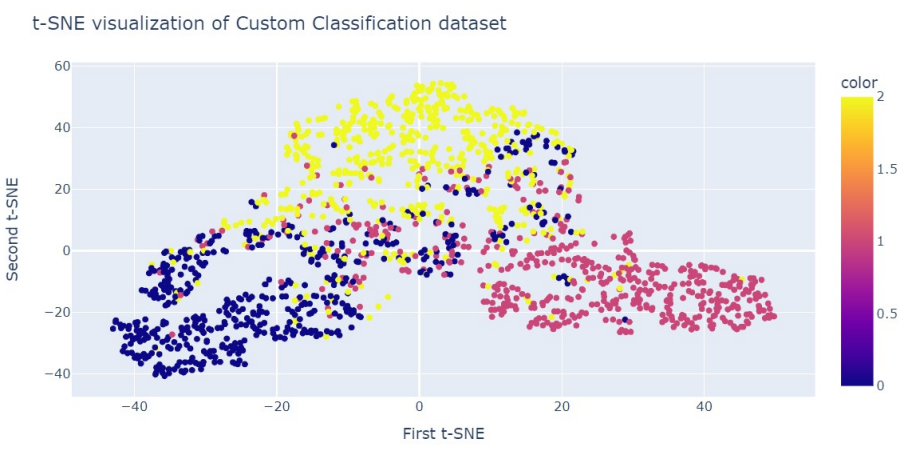
PCA vs t-SNE

- **t-SNE** is concerned with **preserving small pairwise distances** whereas, **PCA** focuses on maintaining large pairwise distances to **maximize variance**
- **PCA preserves the variance in the data**, whereas **t-SNE preserves the relationships between data points in a lower-dimensional space**, making it quite a good algorithm for visualizing complex high-dimensional data.

PCA vs t-SNE

- Examples

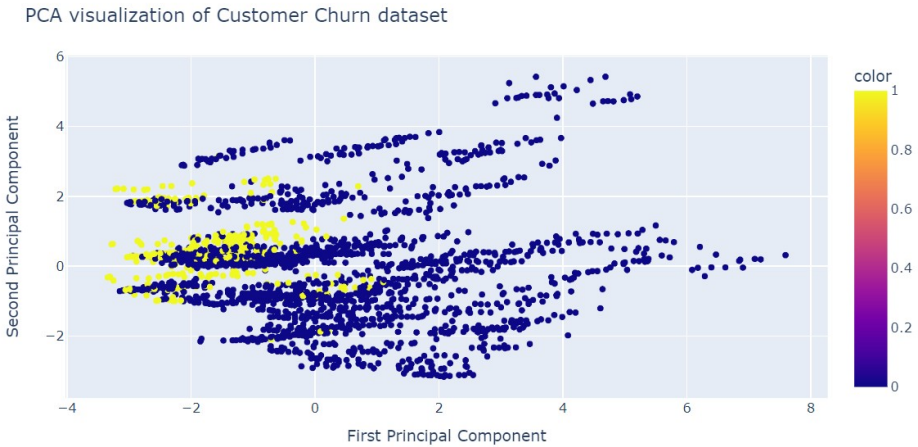


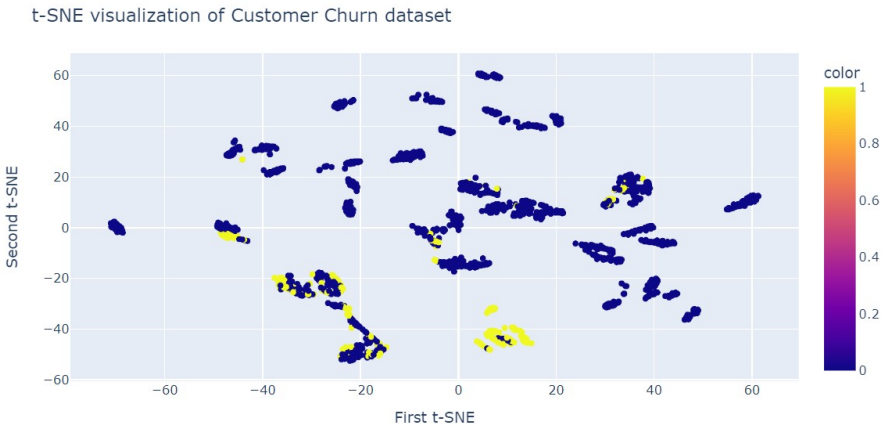


https://www.datacamp.com/tutorial/introduction-t-sne

PCA vs t-SNE

- Examples





https://www.datacamp.com/tutorial/introduction-t-sne

Any questions?

Self-assessment quiz

- Consider the following 5 samples with 4 features. Compute the principal components and their explained variance
- For computing eigenvalues and eigenvectors, you can use python:

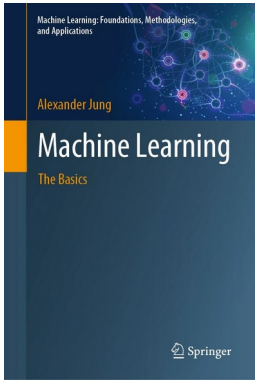
```
from numpy.linalg import eig

eigenvalues,eigenvectors=eig(A)
```

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

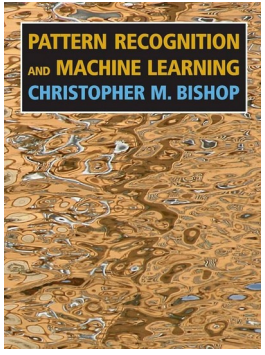
https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9

References: readings

- Chapter 9



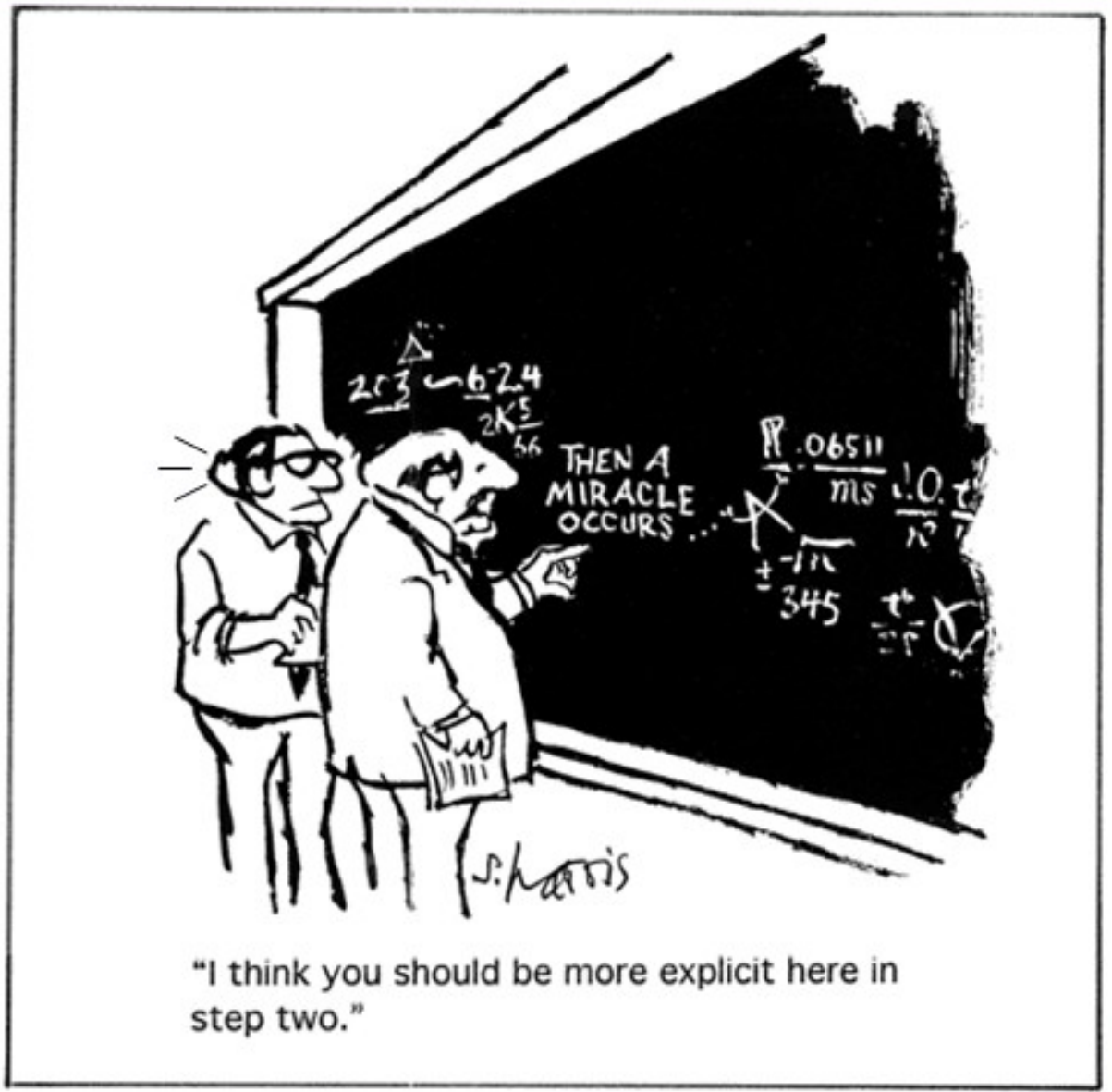
- Chapter 12



- t-SNE further readings: https://pt.slideshare.net/david.kh/visualizing-data-using-tsne
  https://distill.pub/2016/misread-tsne/

Slide acknowledgments

- Kristel Van Steen - Université de Liege - Institut Montefiore
- Aidong Zhang – University at Buffalo
- PICASso group - Princeton University - https://www.cs.princeton.edu/picasso
- Václav Hlaváč - Czech Technical University in Prague



"I think you should be more explicit here in step two."

https://cartoonbank.com/