Chapter 2 - Unsuperised Learning

Dimensionality Reduction and Clustering

Maschinelles Lernen 1 -
Grundverfahren WS19/20

Prof. Gerhard Neumann
KIT, Institut für Anthrophomatik und Robotik

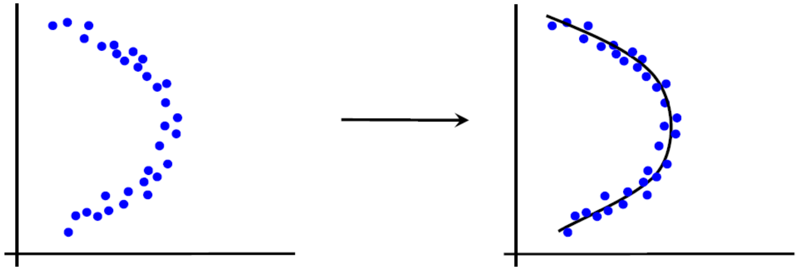
Wrap-Up for Chapter 1: “Simple” Supervised Learning

Algorithm	Reg / Class	Representation	Optimization / Loss	Pros / Cons
Linear Regression	Reg	Feature Space	Squared Error / Least Square Solution	+ Very fast, data-efficient - Heavily depends on features
Logistic Regression	Class	Feature Space	Bernoulli: log-likelihood Gradient Descent	+ data-efficient, fast - „— " —
K-NN	Reg / Class	Instances	Find nearest neighbors	+ very flexible - slow inference, curse of dim!
Random Forest	Reg / Class	Tree / Many Trees Forest	——— Splitting criterion	+ very flexible, fast + high-dimensions - do not reach SOTA performance

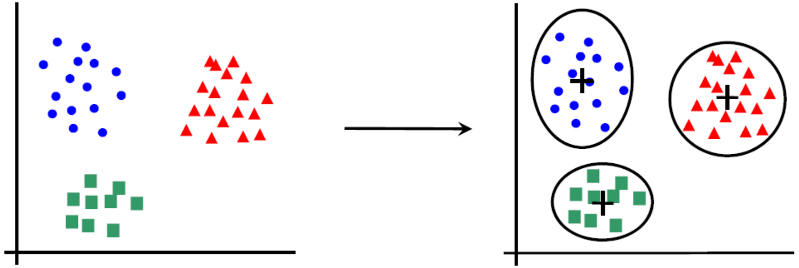
Unsupervised Learning

Trainings data does not include target values, find “structure” in the data

(1) **Dimensionality reduction:**



(2) **Clustering**



(3) **Density estimation:** Generative model of the data

Dimensionality Reduction

Learning Outcomes

- Understand what dimensionality reduction means and why do use it
- Understand what we mean with a “projection” of a vector
- What makes a dimensionality reduction a “good” reduction
- What are the principal components in the data and what is the relation to the covariance matrix
- Learn about constraint convex optimization

Today‘s Agenda!

Dimensionality Reduction:

- Linear Dimensionality Reduction
- Linear Orthogonal Projections
- Reproduction Error
- Principal Component Analysis

Basics: Convex Constraint Optimization

- Lagrangian Multipliers and Constraint Optimization
- Dual Optimization Problem

Slides are largely based on
Slides from Jan Peters

Dimensionality Reduction

Supervised Learning:

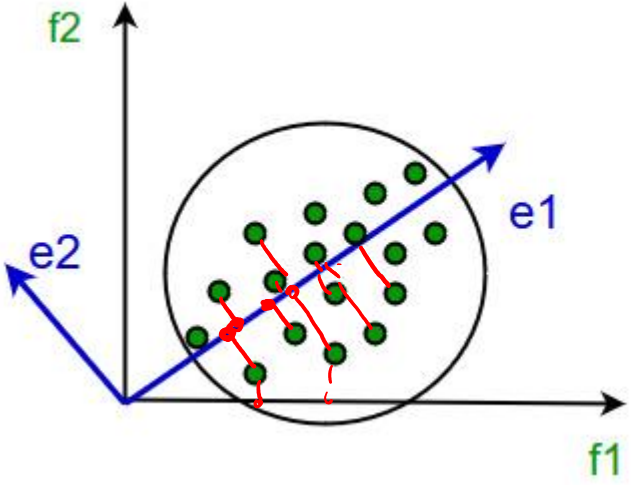
- Learn a mapping from input x to output y

Sometimes, it is quite helpful to analyze the data points themselves

- Unsupervised learning
- Particularly:
  - Reduce the dimensionality of the data

Possible application:

- Visualization of the data
- Preprocessing for any learning algorithm



Motivation from Linear Least-squares Regression

- In least-squares linear regression the parameters are computed as

$$\boldsymbol{w} = \underline{(\boldsymbol{X}^T\boldsymbol{X})}^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

where $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ and $\boldsymbol{y} \in \mathbb{R}^{n\times 1}$

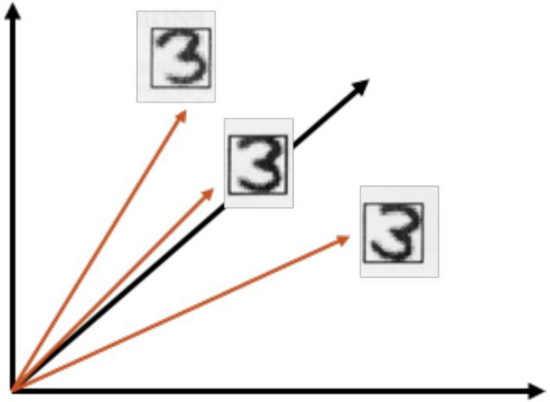
- We need to invert a d × d matrix, which naively costs $O(d^3)$

- Hence, it would be helpful to find a new d_{new} << d to gain computational advantage while not loosing prediction performance

Dimensionality Reduction

- How can we find more efficient representations for our data?
- How can we capture the “essence” of the data?

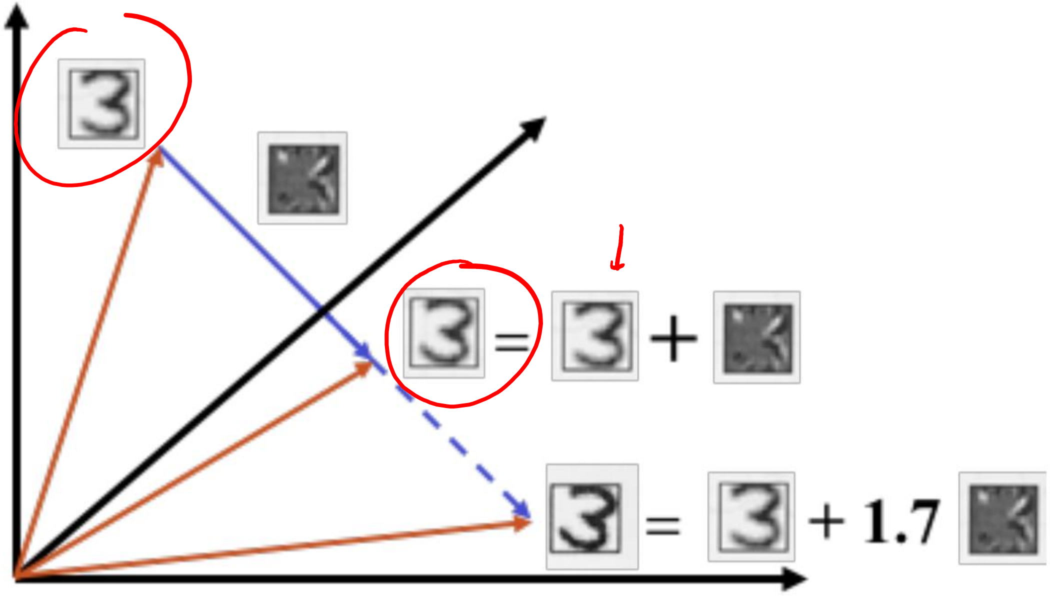
Example: images of the digit 3



- The images can be represented as points in a high-dimensional space (e.g., with one dimension per pixel, in a 4k image there are around 9 million dimensions!)

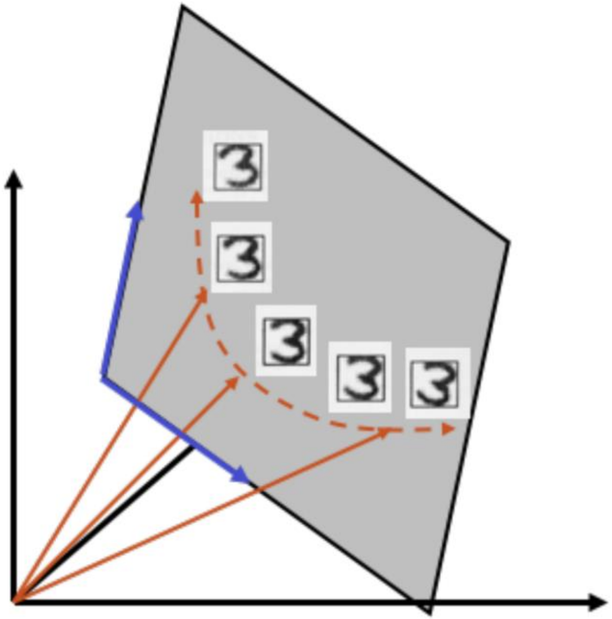
Linear Dimensionality Reduction

To make things easier, we will once again assume linear models. A data point (here: one image) can be written as a linear combination of bases (here: basis images)



Linear Dimensionality Reduction

- What linear transformations of the data can be used to define a lower-dimensional subspace that captures most of the structure?



Linear Dimensionality Reduction

Problem definition:

- Original data point i: $\boldsymbol{x}_i \in \mathbb{R}^D$
- Low-dimensional representation of data point i: $\boldsymbol{z}_i \in \mathbb{R}^M$ with D >> M
- **Goal:** find a mapping

$$\boldsymbol{x}_i \rightarrow \boldsymbol{z}_i$$

- Restrict this mapping to be a linear function

$$\boldsymbol{z}_i = \boldsymbol{W}\boldsymbol{x}_i, \text{ with } \boldsymbol{W} \in \mathbb{R}^{M \times D}$$

Orthonormal Basis Vectors

We can always write a vector in terms of an orthonormal basis coordinate system

$$\boldsymbol{x} = \sum_{i=1}^{D} z_i \boldsymbol{u}_i, \text{ where } \boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij} \text{ and } \delta_{ij} = 1 \text{ if } i = j, \text{ 0 otherwise}$$

- **Orthonormality condition:** The product of 2 different basis vectors is 0. The norm of each basis vector is 1.

Example:

$$\begin{bmatrix} 3 \\ 7 \end{bmatrix} = \textcircled{3} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \textcircled{7} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Projections

The coefficients z_i can be obtained by projecting **x** on the basis vector u_i

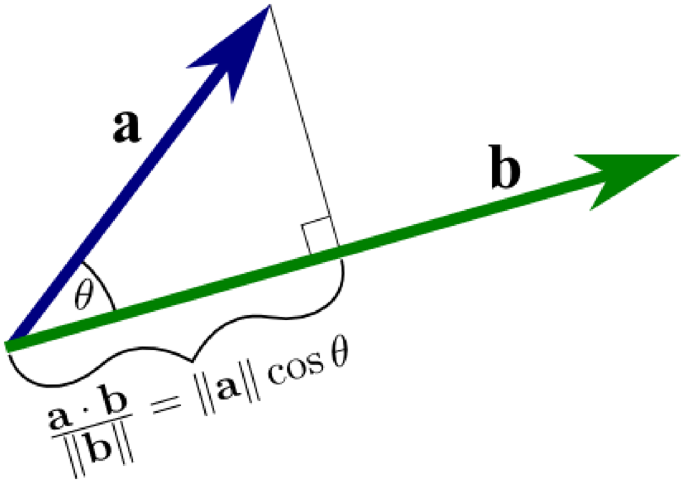
$$\underbrace{z_i}_{\text{scalar coefficient}} = \underbrace{\boldsymbol{u}_i^T \boldsymbol{x}}_{\text{projection}}$$

Example:

$$\boldsymbol{x} = z_1 \boldsymbol{u}_1 + z_2 \boldsymbol{u}_2$$

$$\boldsymbol{u}_1^T \boldsymbol{x} = z_1 \underbrace{\boldsymbol{u}_1^T \boldsymbol{u}_1}_{=1} + z_2 \underbrace{\boldsymbol{u}_2^T \boldsymbol{u}_1}_{=0} = z_1$$

Projection of 2 vectors



a
b
θ
$\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|} = \|\mathbf{a}\| \cos\theta$

Decomposition

Use M << D basis vectors:

$$\boldsymbol{x} = \underbrace{\sum_{i=1}^{M} z_i \boldsymbol{u}_i}_{\tilde{\boldsymbol{x}} \approx \boldsymbol{x}} + \underbrace{\sum_{j=M+1}^{D} z_j \boldsymbol{u}_j}_{\text{skip}}$$

Find the M basis vectors u_i that minimize the mean squared reproduction error:

$$\arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} E(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M) = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2$$

Minimizing the error

Assuming a single basis vector, the error can be written as

$$E(\boldsymbol{u}_1) = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \overset{z_{i1} u_1}{\tilde{\boldsymbol{x}}_i}||^2 = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \underbrace{(\boldsymbol{u}_1^T \boldsymbol{x}_i)}_{\boxed{z_{i1}}} \boldsymbol{u}_1||^2$$

$$= \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - 2(\boldsymbol{u}_1^T \boldsymbol{x}_i)^2 + (\boldsymbol{u}_1^T \boldsymbol{x}_i)^2 \underbrace{\boldsymbol{u}_1^T \boldsymbol{u}_1}_{1} = \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - \underbrace{(\boldsymbol{u}_1^T \boldsymbol{x}_i)^2}_{z_{i1}^2}$$

$$= \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - z_{i1}^2$$

Minimizing the error

The error can be written as

$$E(\boldsymbol{u}_1) = \sum_{i=1}^{N} \xcancel{\boldsymbol{x}_i^T \boldsymbol{x}_i} - z_{i1}^2$$

$$\Rightarrow \arg\min_{\boldsymbol{u}_1} E(\boldsymbol{u}_1) = \arg\max_{\boldsymbol{u}_1} \sum_{i=1}^{N} z_{i1}^2 = \arg\max_{\boldsymbol{u}_1} \sum_{i=1}^{N} (\boldsymbol{u}_1^T \boldsymbol{x}_i)^2$$

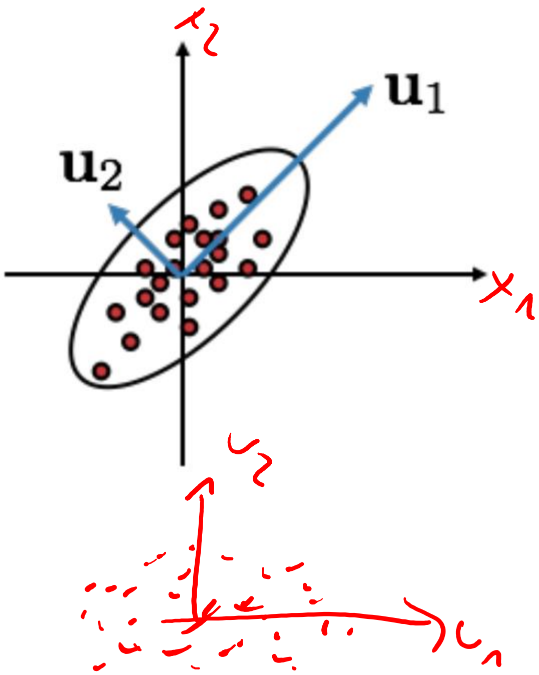
- Minimizing the error is equivalent to maximizing the variance of the projection. (Assuming a zero mean on the data)
- We can ensure a zero mean projection by subtracting the mean from the data

$$\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$$

Illustration

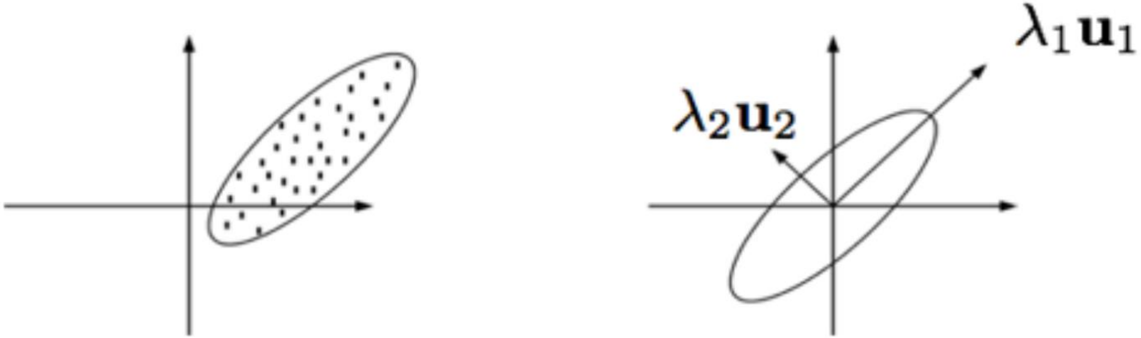
$$\tilde{\boldsymbol{x}} = \sum_{i=1}^{M} z_i \boldsymbol{u}_i + \boldsymbol{\mu}$$

- Projecting onto $\mathbf{u_1}$ captures the majority of the variance and hence projecting onto it minimizes the error
- Note that these axes are orthogonal and decorrelate the data
  - i.e. in the coordinate frame of these axes, the data is uncorrelated (side note: this only works for Gaussians)



Principle component analysis (PCA)

Goal: find the so-called principal directions, and the variance of the data along each principal direction



- λ_i is the marginal variance along the principal direction $\boldsymbol{u}_i$

Principle component analysis

- The first principal direction **u₁** is the direction along which the variance of the projected data is maximal

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \frac{1}{N}\sum_{i=1}^{N}\Big(\overbrace{\boldsymbol{u}^T \underbrace{(\boldsymbol{x}_i - \boldsymbol{\mu})}_{\bar{\boldsymbol{x}}_i}}^{z_i}\Big)^2 \quad \text{s.t. } \boldsymbol{u}^T\boldsymbol{u} = 1$$

  - The directions all have unit norm

- The second principal direction maximizes the variance of the data in the orthogonal complement of the first principal direction

Derivation…

- **Objective in matrix form…**

$$E(\boldsymbol{u}) = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})\right)^2 \qquad a^2 = a * a^T$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\underbrace{\overbrace{\boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})}^{\text{scalar}}\overbrace{(\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{u}}^{\text{scalar}}}_{\text{outer product} \to D \times D}\right)$$

$$= \boldsymbol{u}^T\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T\right)}_{\text{covariance } \boldsymbol{\Sigma}}\boldsymbol{u} = \boldsymbol{u}^T\boldsymbol{\Sigma}\boldsymbol{u}$$

Matrix Form

- The objective can be written in terms of the sample covariance!

Derivation…

We obtain the following constrained optimization problem

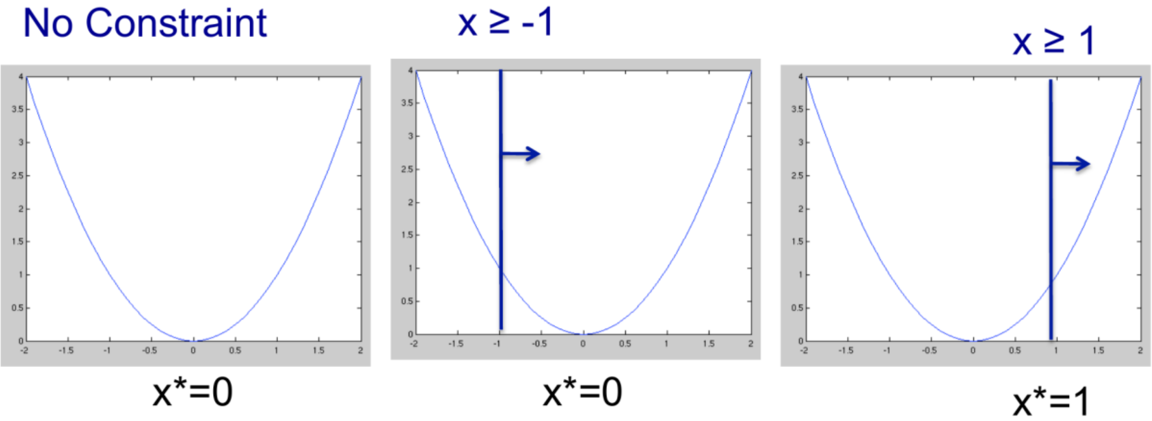
$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \ \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} \quad \text{s.t. } \boldsymbol{u}^T \boldsymbol{u} = 1$$

We need to look at constraint optimization first!

Constraint Optimization

Basics: Constrained Optimization

Simple constrained optimization problem: $\overset{\text{objective}}{\arg\min_x x^2} \quad \overset{\text{constraint}}{\text{s.t.} \ x \geq b}$



No Constraint

x ≥ -1

x ≥ 1

x*=0

x*=0

x*=1

How do we solve the constrained optimization problem? **Lagrangian Multipliers!**

Basics: Lagrangian Multipliers

$$\min_x x^2 \quad \text{s.t. } x \geq b$$

The Lagrangian:

- L = objective - multiplier * constraint

$$L(x,\lambda) = \underbrace{x^2}_{\text{objective}} - \underbrace{\lambda}_{\text{multiplier}} \cdot \underbrace{(x-b)}_{\text{constraint}}$$

Lagrangian optimization:

$$\min_x \max_\lambda L(x,\lambda), \quad \text{s.t. } \lambda \geq 0$$

Why is this equivalent?

Min **fights** ***max*****!**

- $x < b :$
  - $(x-b) < 0 \rightarrow \max_\lambda -\lambda(x-b) = \infty$
  - *min* won't let that happen
- $x > b :$
  - $(x-b) > 0, \lambda \geq 0 \rightarrow \lambda^* = 0$
  - L is the same as original objective
- $x = b :$
  - λ can be anything
  - L is the same as original objective

Min **forces** ***max*** **to behave such that constraints are satisfied**

General formulation

General Formulation: $$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \ \text{for } i = 1 \dots K$$

- Several inequality constraints (equality constraints also possible)

Lagrangian optimization: $$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\lambda}), \qquad L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{K} \lambda_i (h_i(\boldsymbol{x}) - b_i)$$
$$\text{s.t.} \ \lambda_i \geq 0, \ \text{for } i = 1 \dots K$$

Dual formulation

Primal optimization problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \ \text{for } i = 1 \dots K$$

Dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \quad g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda})$$
$$\text{s.t.} \quad \lambda_i \geq 0, \ \text{for } i = 1 \dots K$$

- *g* is also called the dual function of the optimization problem
- We essentially swapped min and max in the definition of L

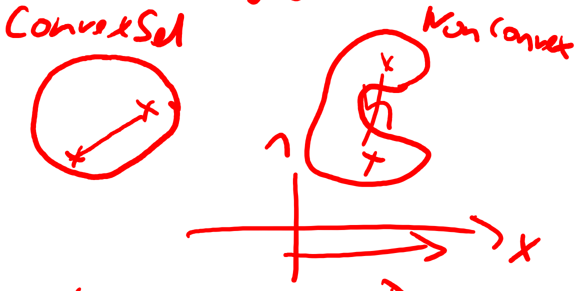
Slaters condition: For a convex objective and convex constraints, solving the dual is equivalent to solving the primal!

- Optimal primal parameters can be obtained from optimal dual parameters, i.e.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$$

Example:

Convex Constraints:
Set of allowed solutions is convex



$$\min_x x^2 \quad \text{s.t.} \;\; x \geq 1$$

1) Lagrangian $L(x, \lambda) = x^2 - \lambda(x-1)$

2) Find $x^* = \text{argmin}\, L(x, \lambda)$: $\frac{\partial L}{\partial x} = 2x - \lambda \overset{!}{=} 0 \Rightarrow x^* = \frac{\lambda}{2}$

3) Dual-Function $g(\lambda) = L(x^*, \lambda) = \frac{\lambda^2}{4} - \lambda\left(\frac{\lambda}{2} - 1\right) = -\frac{\lambda^2}{4} + \lambda$

4) Find $\lambda^* = \text{argmax}\, g(\lambda)$: $\frac{\partial g}{\partial \lambda} = -\frac{\lambda}{2} + 1 = 0 \Rightarrow \lambda^* = 2$

5) $x^* = \frac{\lambda^*}{2} = 1$

Back to the PCA Derivation…

We obtain the following constrained optimization problem

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \ \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} \quad \text{s.t. } \boldsymbol{u}^T \boldsymbol{u} = 1$$

- We now know what to do… Lagrangian optimization

The Lagrangian is given by:

$$L(\boldsymbol{u}, \lambda) = \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} - \lambda(\boldsymbol{u}^T \boldsymbol{u} - 1)$$

- Optimal solution for u:

$$\frac{\partial L(\boldsymbol{u}, \lambda)}{\partial \boldsymbol{u}} = \cancel{2}\boldsymbol{\Sigma}\boldsymbol{u} - \cancel{2}\lambda\boldsymbol{u} \overset{!}{=} \boldsymbol{0} \quad \Rightarrow \boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$$

This is an Eigen-value problem!

Basics: Eigenvalues and Eigenvectors

- Let the Eigenvectors and Eigenvalues of **C** be **u**$_{\mathbf{k}}$ and λ_k for $k \leq D$ i.e.,

$$\boldsymbol{C}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k \ \text{ with } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

Ordered list of Eigenvalues

- In matrix form:

$$\boldsymbol{C}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda} \ \text{ with } \boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_D) \text{ and } \boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D]$$

- Because **U** is orthonormal (eigenvectors have unit norm), we know that $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$

- This mean that we can decompose **C** as

$$(\boldsymbol{C}\underbrace{\boldsymbol{U})\boldsymbol{U}^T}_{I} = (\boldsymbol{U}\boldsymbol{\Lambda})\boldsymbol{U}^T \Rightarrow \boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

Basics: Eigenvalues and Eigenvectors

Every positive definite symmetric matrix can be decomposed in its Eigendecomposition

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T = \underbrace{\begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_D \end{bmatrix}}_{\text{Eigenvectors}} \underbrace{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_D \end{bmatrix}}_{\text{Eigenvalues}} \begin{bmatrix} \boldsymbol{u}_1^T \\ \vdots \\ \boldsymbol{u}_D^T \end{bmatrix}$$

Back to PCA

Eigenvalues-Eigenvectors of the covariance matrix

$$\boldsymbol{\Sigma u} = \lambda \boldsymbol{u}$$

- The largest Eigenvalue gives us the maximal variance
- The corresponding Eigenvector gives us the direction with maximal variance

Principal Component Analysis

- **Observation:** If $\lambda_k \approx 0$ for k > M for some M << D, then we can use the subset of the first D eigenvectors to define a basis for approximating the data vectors with loosing accuracy

$$\boldsymbol{x}_i - \boldsymbol{\mu} = \underbrace{\sum_{j=1}^{M} z_{ij}\boldsymbol{u}_j}_{\tilde{\boldsymbol{x}}} + \underbrace{\sum_{j=M+1}^{D} z_{ij}\boldsymbol{u}_j}_{\text{close to } 0} \Rightarrow \boldsymbol{x}_i \approx \boldsymbol{\mu} + \sum_{j=1}^{M} z_{ij}\boldsymbol{u}_j$$

- This representation has the minimal mean squared error (MSE) of all linear representations of dimension D

$$\operatorname*{arg\,min}_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} E(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M) = \operatorname*{arg\,min}_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2$$

Principal Component Analysis

Now we know how we can represent our data in a lower dimensional space in a principled way

- Center the data around the mean (compute the mean of the data and subtract it)
- Compute the covariance matrix, decompose it, and choose the first M largest Eigenvalues and corresponding Eigenvectors
- This gives us an (Eigen)basis for representing the data
  - **Projection to low-D:** $\boldsymbol{z}_i = \boldsymbol{B}^T(\boldsymbol{x}_i - \boldsymbol{\mu})$
  - **Reprojection to high-D:** $\tilde{\boldsymbol{x}}_i = \boldsymbol{\mu} + \boldsymbol{B}\boldsymbol{z}_i$

  with $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_M \end{bmatrix}$
- It is also common to normalize the variance of each dimension (i.e. unit variance)

How to choose M

- A larger M leads to a better approximation. In the limit, when M = D we stay in the initial data dimensions
- There are at least 2 good possibilities for choosing M
  - Choose D based on application performance, i.e. choose the smallest M that makes the application work well enough
  - Choose D so that the Eigenbasis captures some fraction of the variance (for example η = 0.9).
    The eigenvalue λ_i describes the marginal variance captured by $\mathbf{u}_i$

$$\text{Choose } D \text{ s.t. } \sum_{i=1}^{M} \lambda_i = \eta \underbrace{\sum_{i=1}^{D} \lambda_i}_{\text{Total variance of the data}}$$

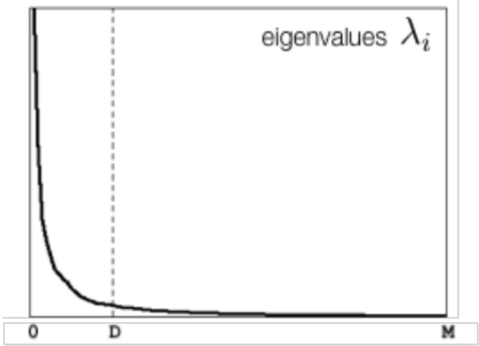


Image representation with PCA

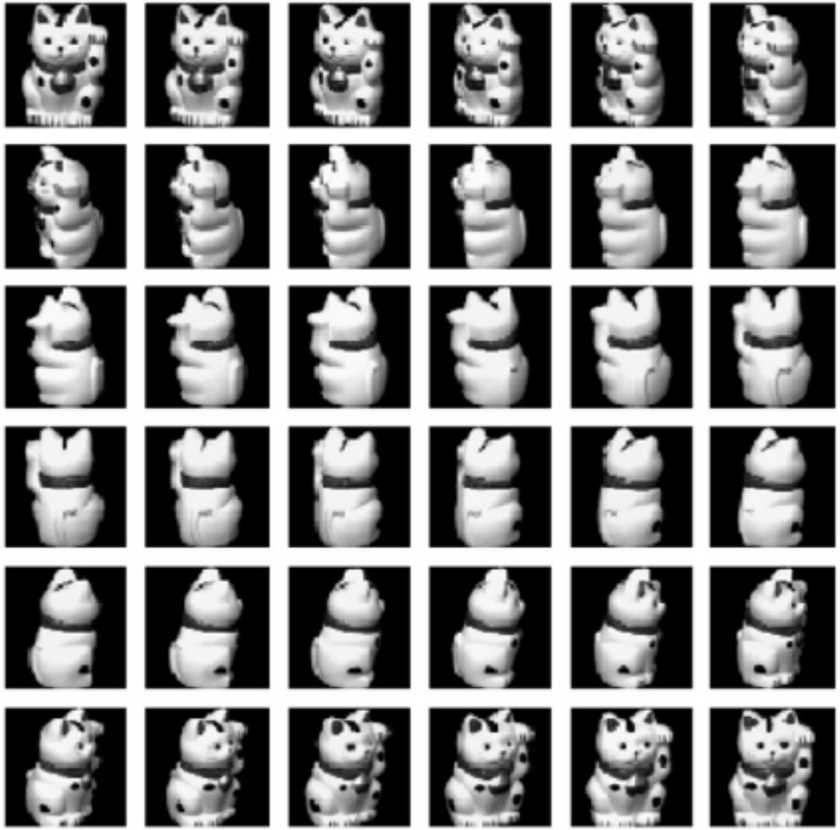
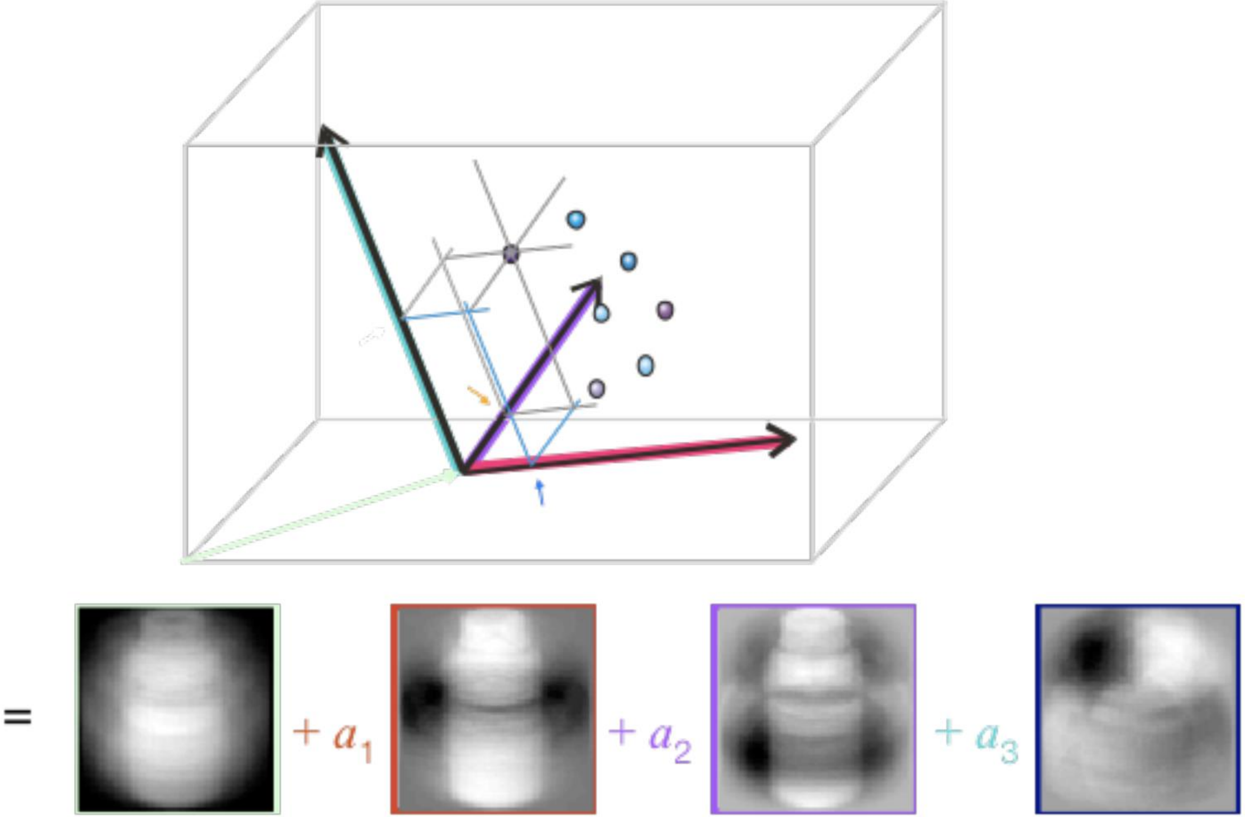


Image representation with PCA



Eigenfaces

- The first popular use of PCA for object recognition was for the detection and recognition of faces [Turk and Pentland, 1991]
- Collect a face ensemble
- Normalize for contrast, scale, & orientation
- Remove backgrounds
- Apply PCA & choose the first M eigen-images that account for most of the variance of the data

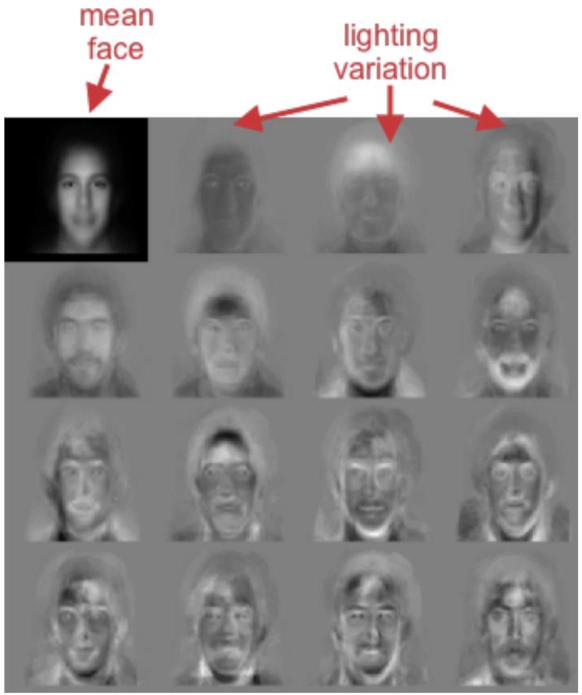
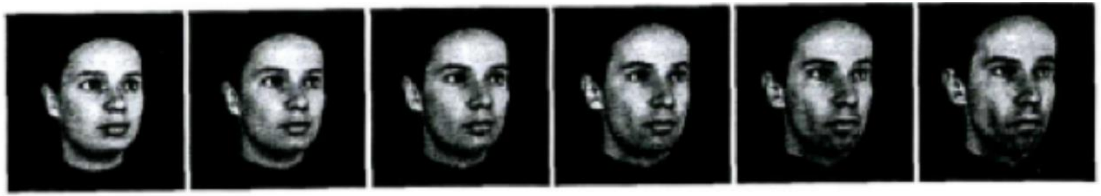
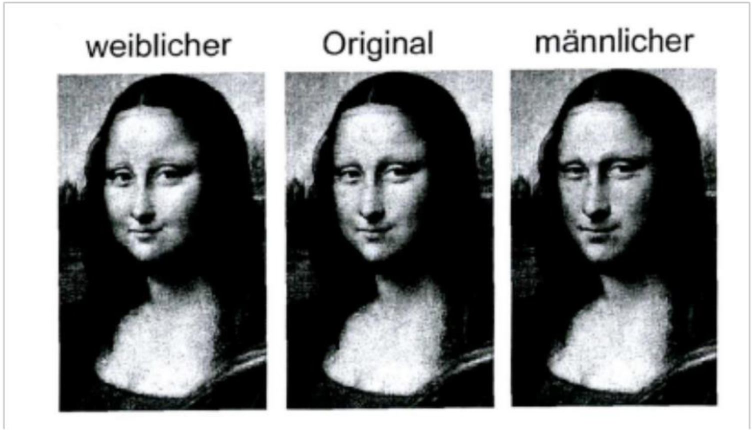


Image Morphing with PCA





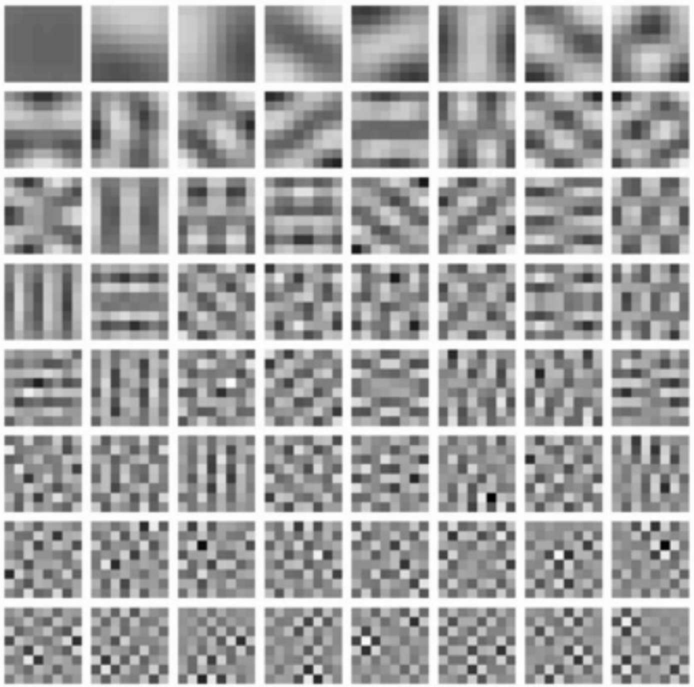
weiblicher Original männlicher

Generic Image Ensembles

Is there a low-dimensional model describing natural images?

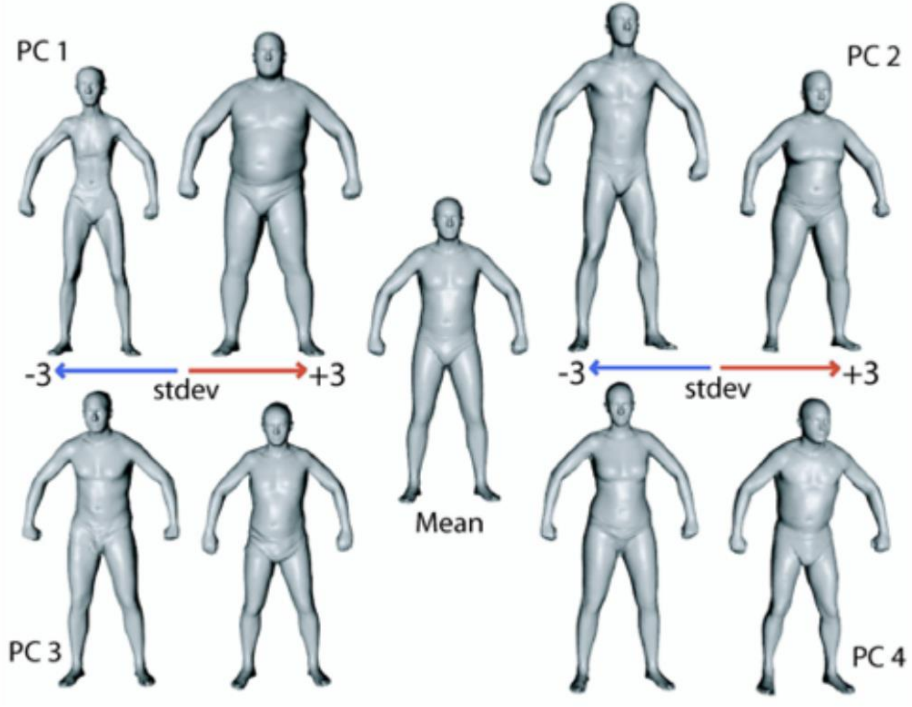
PCA of natural image patches

8x8 image patches



PCA Model of body shapes

- PCA on a detailed triangle model of human bodies [Anguelov et al. 05]



Wrap-up

Summary:

- PCA projects the data into a linear subspace
- PCA maximizes the variance of the projection
- PCA minimizes the error of the reconstruction
- We just covered the most simple linear dimensionality reduction technique
  - Many more sophisticated techniques exist
  - Kernel PCA, Auto-Encoders, t-SNE, non-negative matrix factorization (interesting, but no time to cover those...)

Applications:

- PCA allows us to transform a high-dimensional input space to a low-dimensional feature space, while capturing the essence of the data
- PCA finds a more natural coordinate system for the data
- PCA is a very common preprocessing step for high-dimensional input data

Self-test questions

What have we learned today?

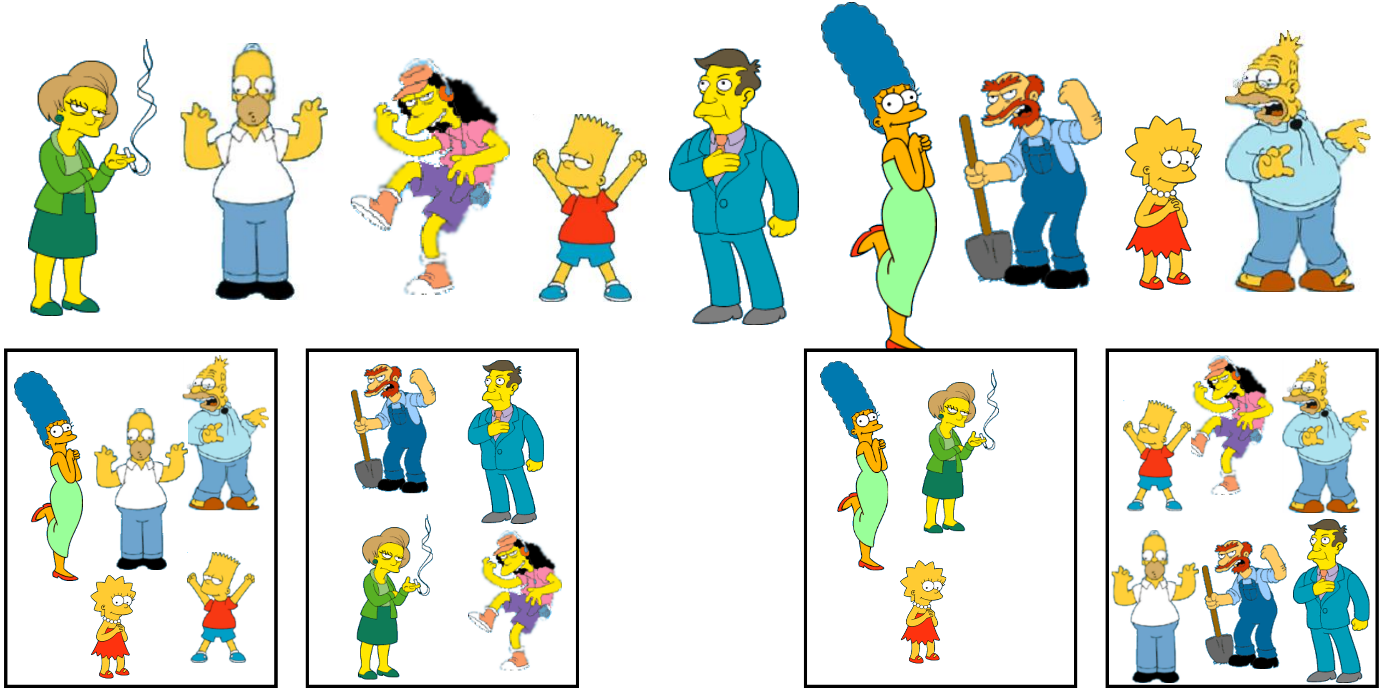
- What does dimensionality reduction mean?
- How does linear dimensionality reduction work?
- What is PCA? What are the three things that it does?
- What are the roles of the Eigenvectors and Eigenvalues in PCA?
- Can you describe applications of PCA?

Clustering

Slides based on lectures from Stefan Roth (TU Darmstadt)
and Carla Brodley (Tulft University)

Clustering is Subjective

What is the natural grouping of these objects?



Simpson's Family School Employees Females Males

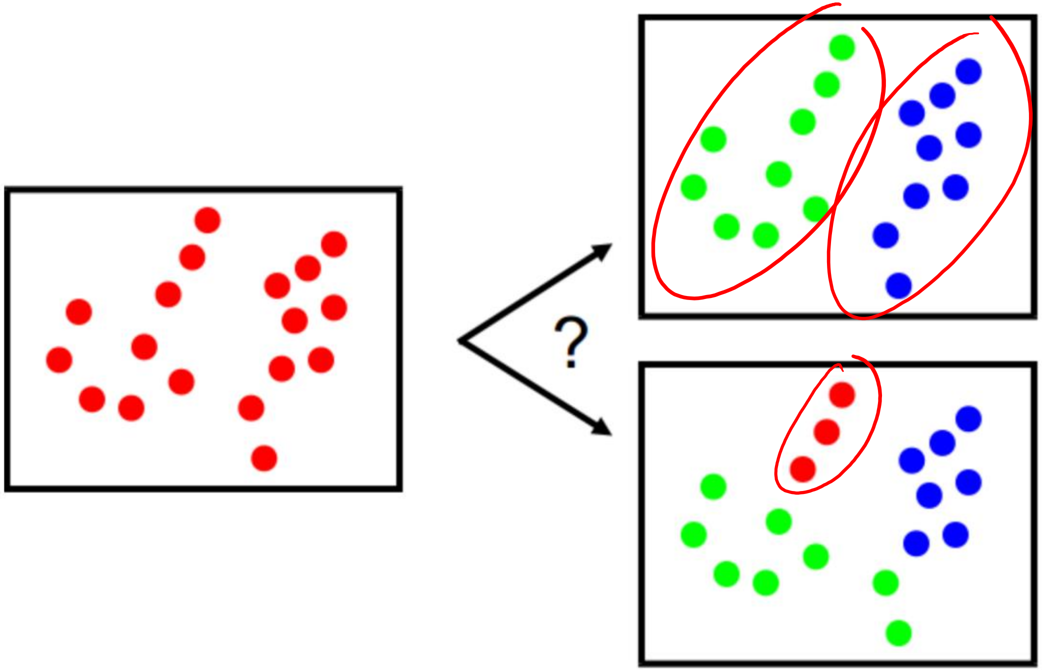
Clustering: Finding structure in the data

What are the correct clusters?

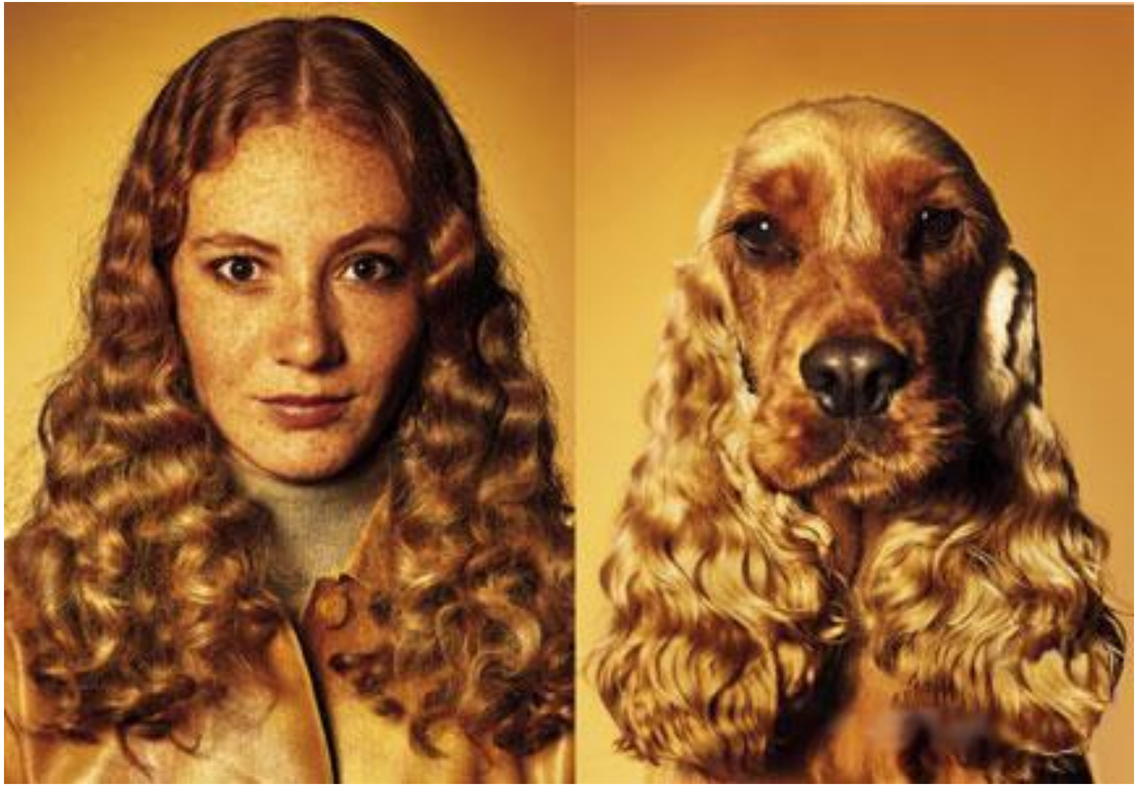
- Ground truth often not available

Similarity measure

- clustering relies on measure of similarity
- e.g. position in space (Euclidean vs. log-polar coordinates), weighting of different dimensions (features)...



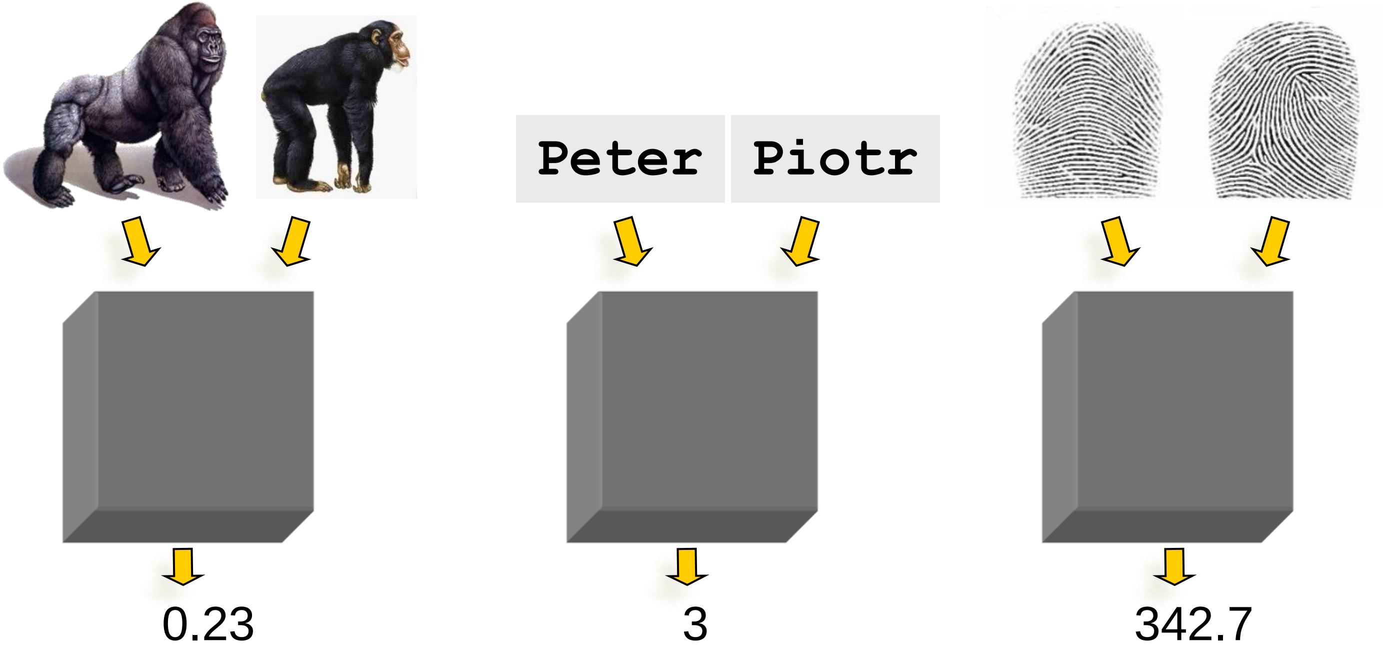
What is similarity?



Similarity is hard to define, but…
"*We know it when we see it*"

Defining Distance Measures

- **Definition**: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



Peter Piotr

0.23

3

342.7

What properties should a distance measure have?

- $D(A,B) = D(B,A)$ ***Symmetry***

  Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

- $D(A,A) = 0$ ***Constancy of Self-Similarity***

  Otherwise you could claim "Alex looks more like Bob, than Bob does."

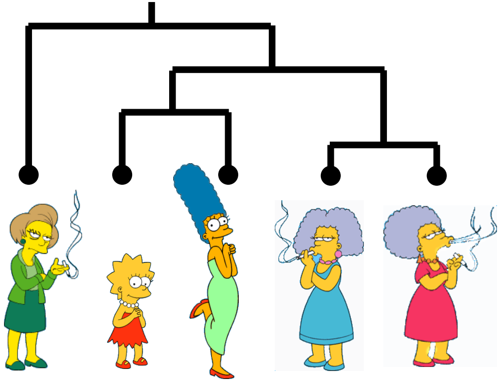
- $D(A,B) = 0$ iif $A= B$ ***Positivity (Separation)***

  Otherwise there are objects in your world that are different, but you cannot tell apart.

- $D(A,B) \leq D(A,C) + D(B,C)$ ***Triangular Inequality***

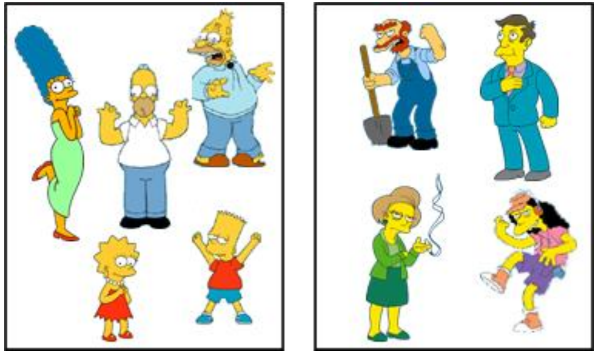
  Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

Basic Clustering Algorithms



Hierarchical clustering methods

- Bottom-up (merging)
- Top-down (splitting, not covered)



Flat clustering algorithms

- K-Means
- Mixture models (see density estimation lecture)

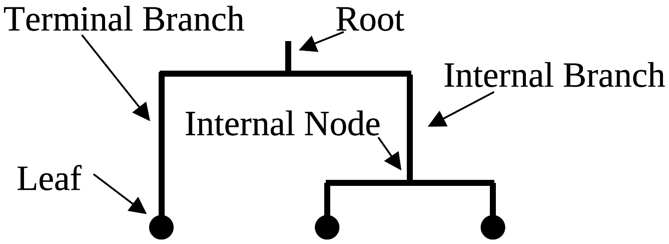
Other clustering methods:

- Spectral clustering (not covered)

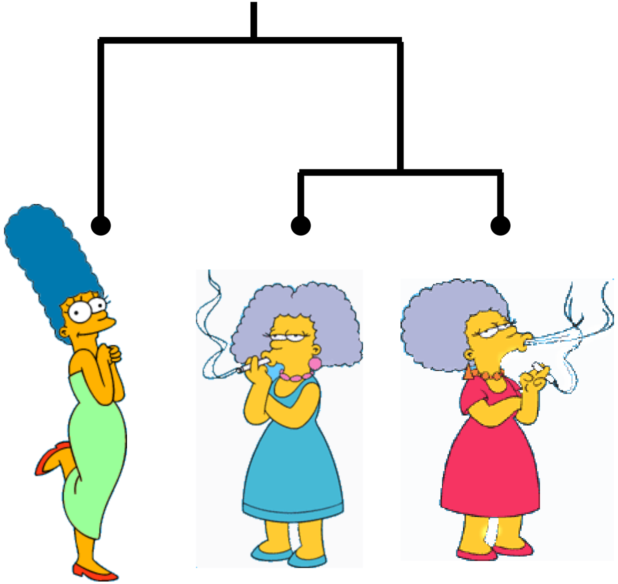
Hierarchical Clustering: Dentograms

Dendogram:

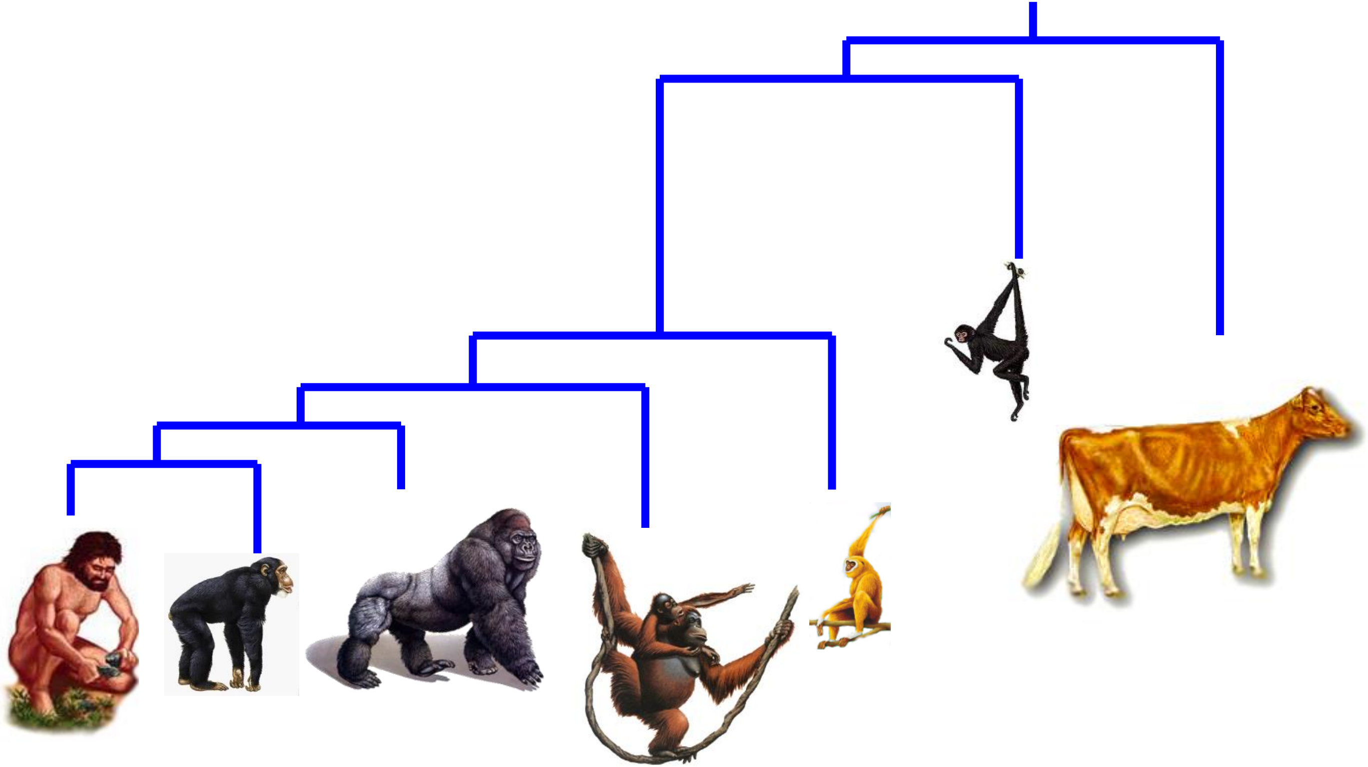
- A useful tool for summarizing similarity measurements



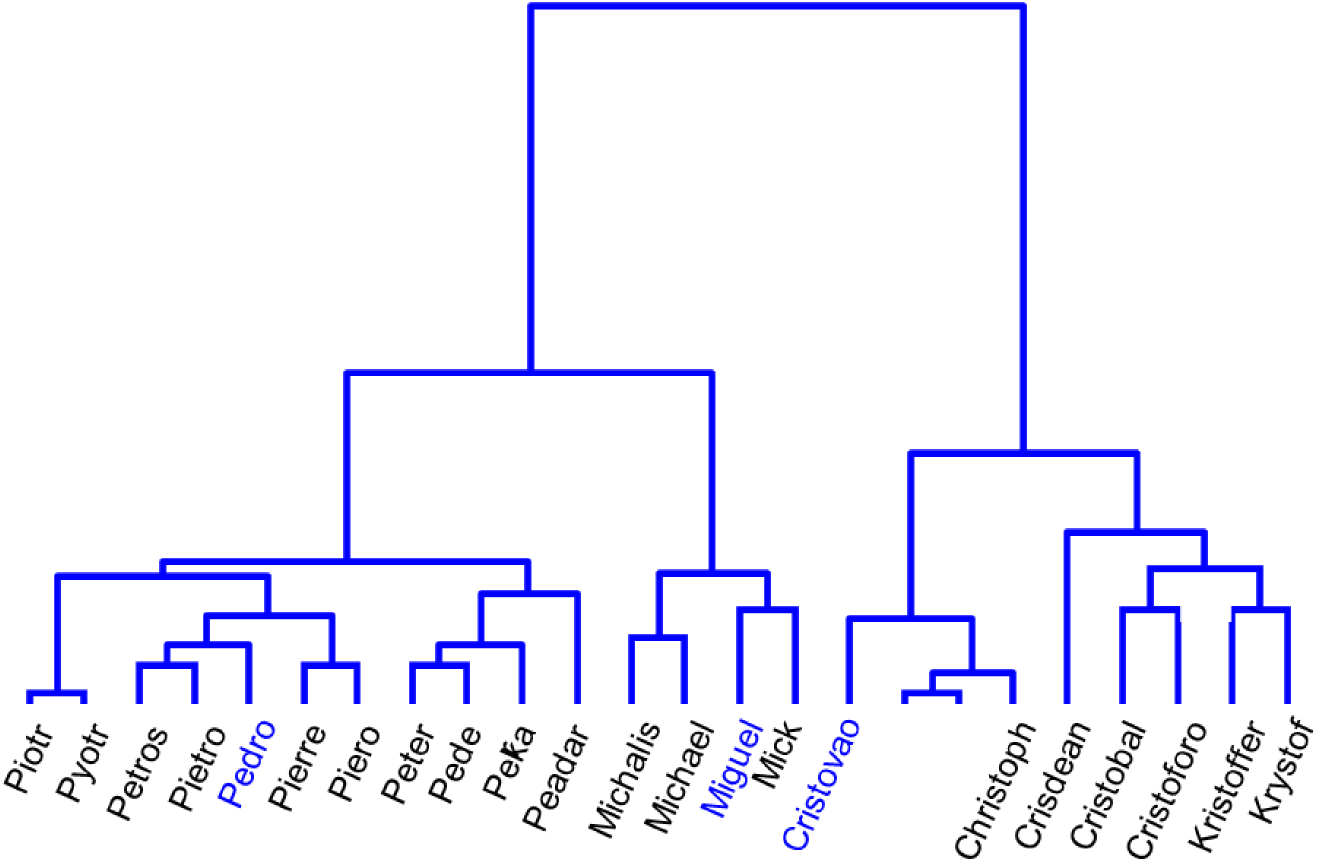
- The similarity between two objects in a dendrogram is represented as the height of the lowest internal node they share



Example: Species based on genetic difference

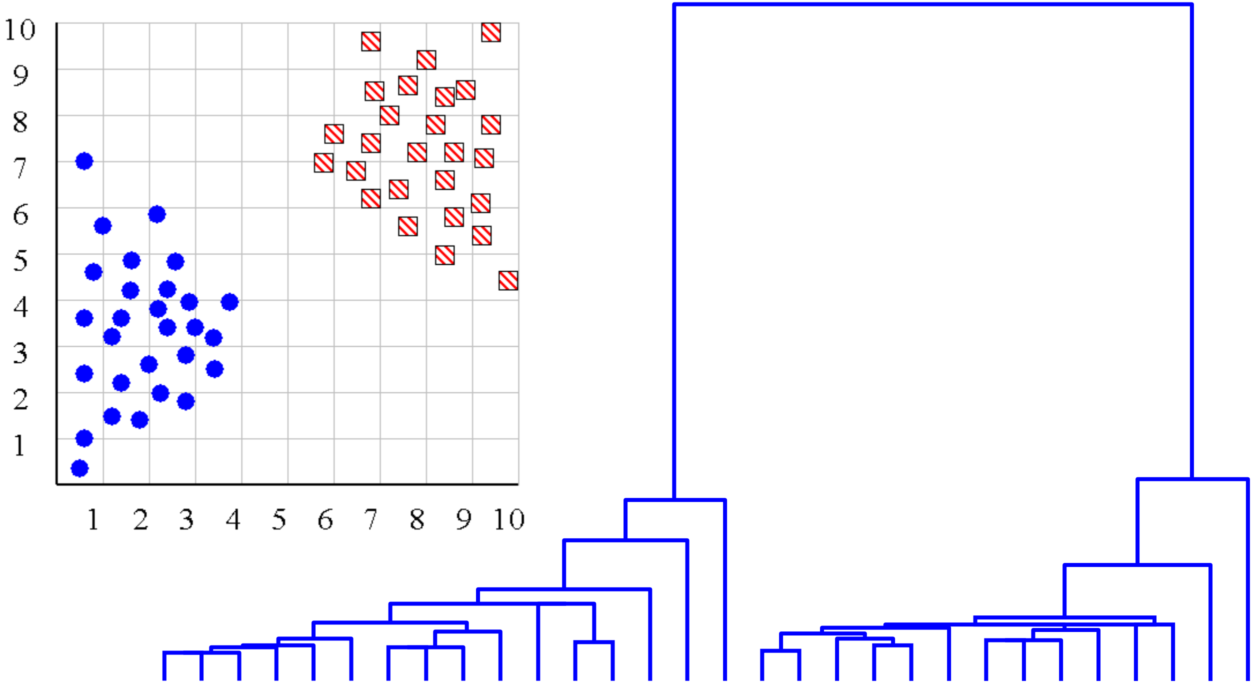


Example: Names based on string edit distance



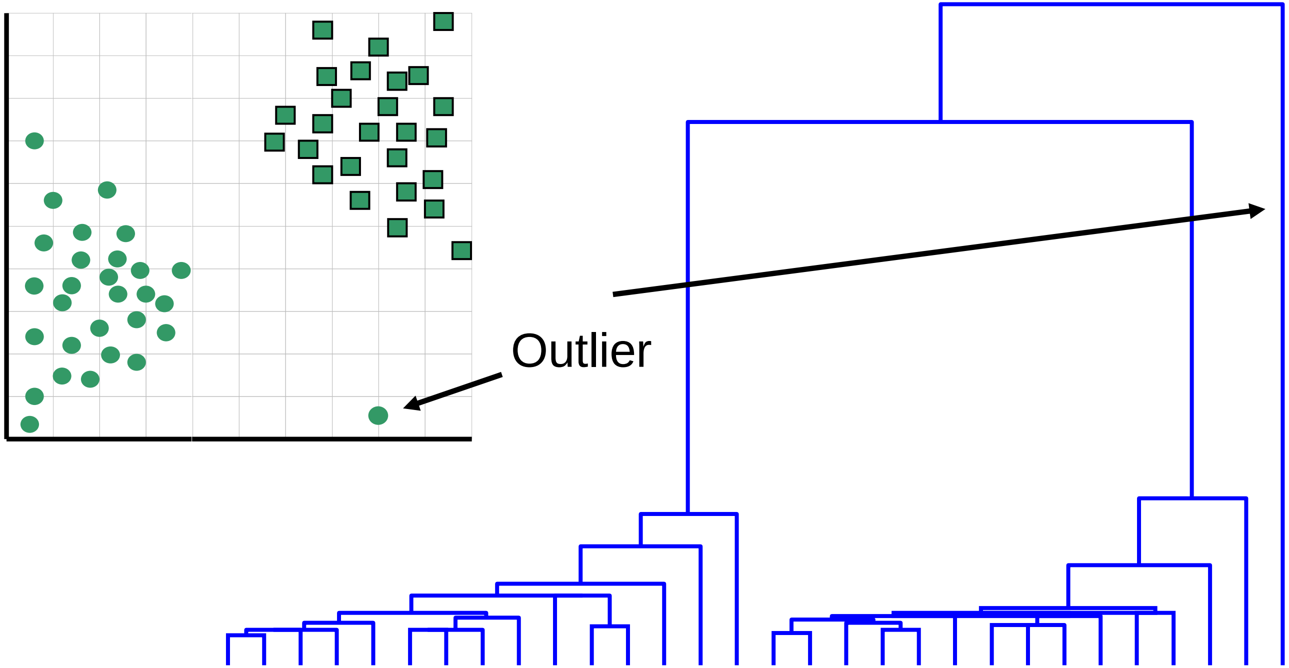
Properties of Dentograms

We can look at the dendrogram to determine the “correct” number of clusters.



Properties of Dentograms

Detecting Outliers: The single isolated branch is suggestive of a data point that is very different to all others



Hierarchical Clustering

Since we cannot test all possible trees we will have to heuristic search of all possible trees. We could do this..

- **Bottom-Up (agglomerative):** Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

- **Top-Down (divisive):** Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides. (not covered)

Bottom-Up Clustering

Hierarchical agglomerative clustering

- sequentially merge the pairs of closest points

Init:

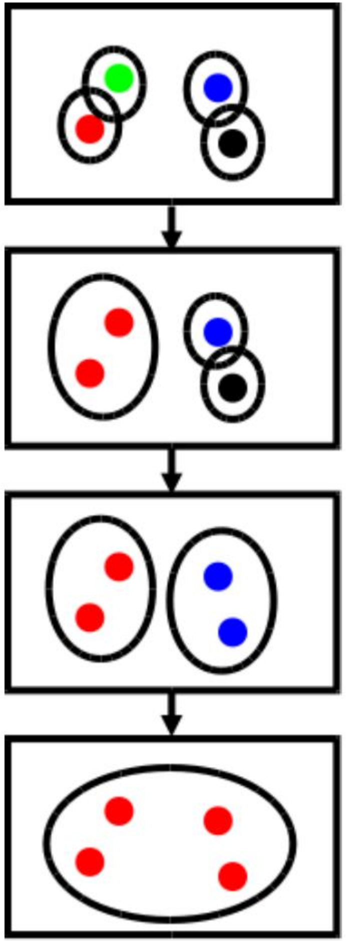
- Each of the n samples becomes a cluster itself

Iterate:

(1) find closest clusters and merge them
(2) proceed until we have a single cluster

Required:

- Distance measure between two samples
- Distance measure between clusters

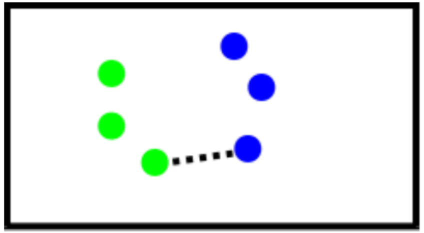


Similarity between clusters

Cluster Linkage: define distances between clusters

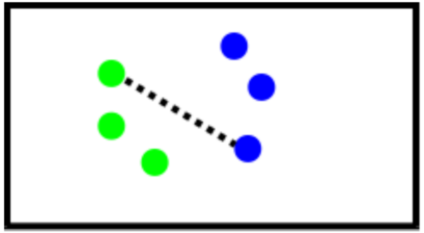
- **Single Linkage:** Minimum distance between two points from the 2 cluster

$$d(C_k, C_l) = \min_{\boldsymbol{x}_i \in C_k} \min_{\boldsymbol{x}_j \in C_l} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

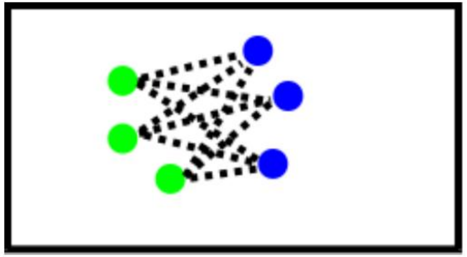


- **Complete Linkage:** Maximum distance between two points from the 2 cluster

$$d(C_k, C_l) = \max_{\boldsymbol{x}_i \in C_k} \max_{\boldsymbol{x}_j \in C_l} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$



Similarity between clusters

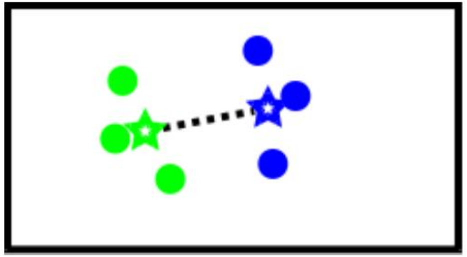


Cluster Linkage: define distances between clusters

- **Average Linkage:** Average distance between any 2 pairs

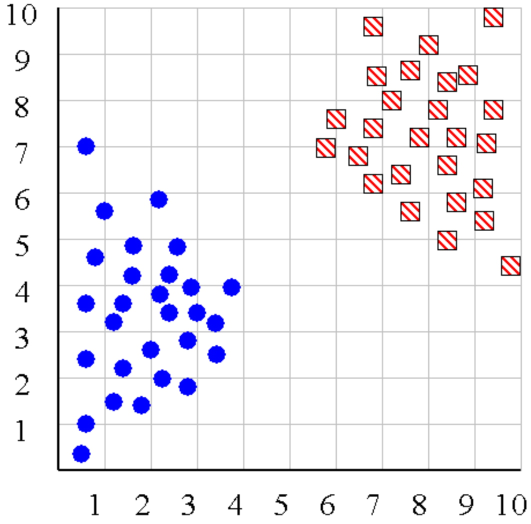
$$d(C_k, C_l) = \frac{1}{|C_l||C_k|} \sum_{\boldsymbol{x}_i \in C_l} \sum_{\boldsymbol{x}_j \in C_k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

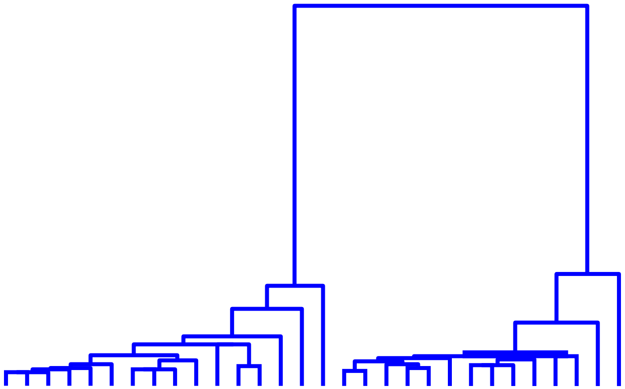
- **Centroid Linkage:** Distance between the 2 centroids



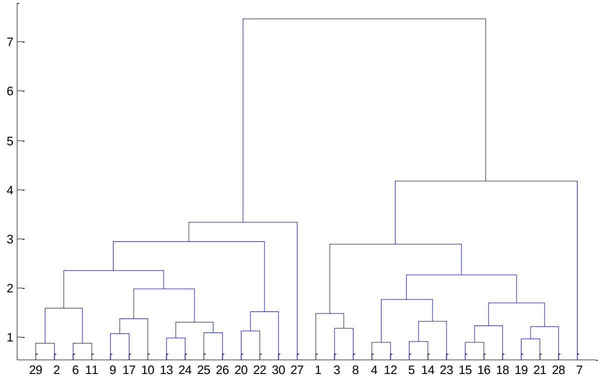
$$d(C_k, C_l) = d\Big(\frac{1}{|C_l|} \sum_{\boldsymbol{x}_i \in C_l} \boldsymbol{x}_i, \frac{1}{|C_k|} \sum_{\boldsymbol{x}_j \in C_k} \boldsymbol{x}_j\Big)$$

Influence of linkage method





Single linkage



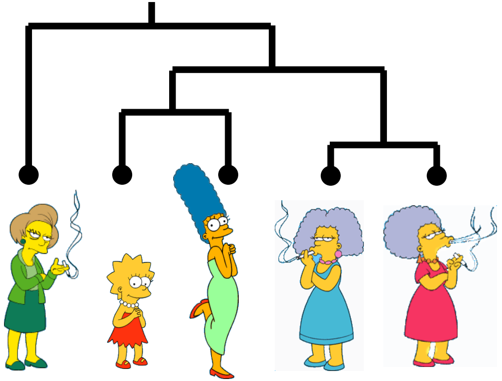
Average linkage

Hierarchal Clustering Methods Summary

Wrap-Up:

- No need to specify the number of clusters in advance
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least O(n^2), where *n* is the number of total objects
- Like any heuristic search algorithms, local optima are a problem
- Interpretation of results is (very) subjective

Basic Clustering Algorithms

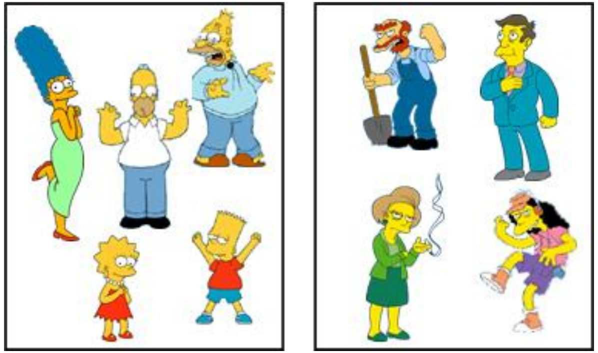


Hierarchical clustering methods

- Bottom-up (merging)
- Top-down (splitting, not covered)

Flat clustering algorithms

- K-Means
- Mixture models (see density estimation lecture)



Other clustering methods:

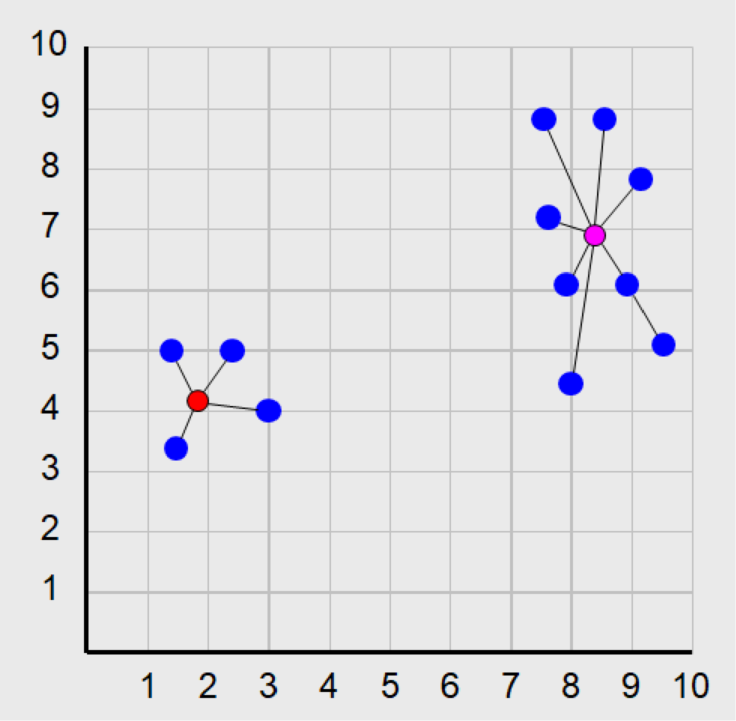
- Spectral clustering (not covered)

K-Means Algorithm

Goal: minimize quantization error!

- Given data $D = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$
- Search for cluster centers/prototypes/centroid $C = \{\boldsymbol{c}_1, \dots, \boldsymbol{c}_k\}$
- Denote with c(x) the closest centroid vector $\boldsymbol{c} \in C$ to $\boldsymbol{x}$
- Here sum of squared distances (SSD) (or sum of squared error) denotes quantization error

$$\mathrm{SSD}(C;\mathcal{D}) = \sum_{i=1}^{n} d(\boldsymbol{x}_i, c(\boldsymbol{x}_i))^2$$



K-Means Algorithm

Iterative Procedure

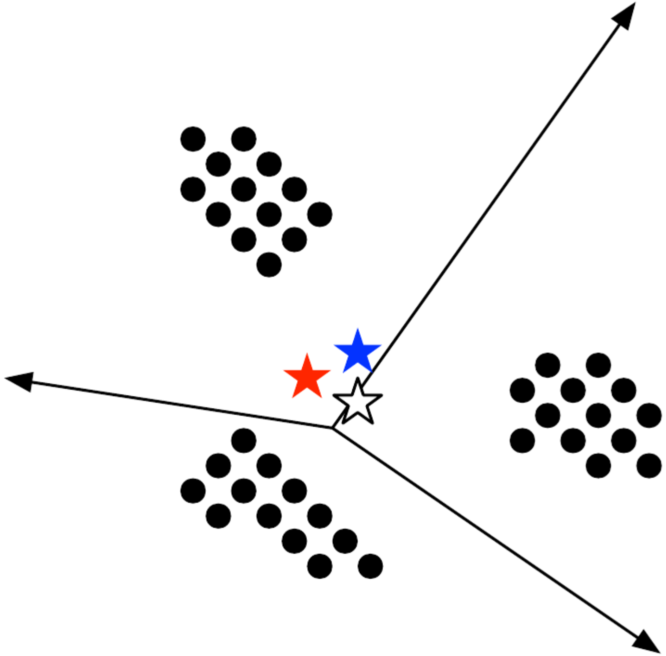
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step 1: The stars are cluster centers, randomly assigned at first.

K-Means Algorithm

Iterative Procedure

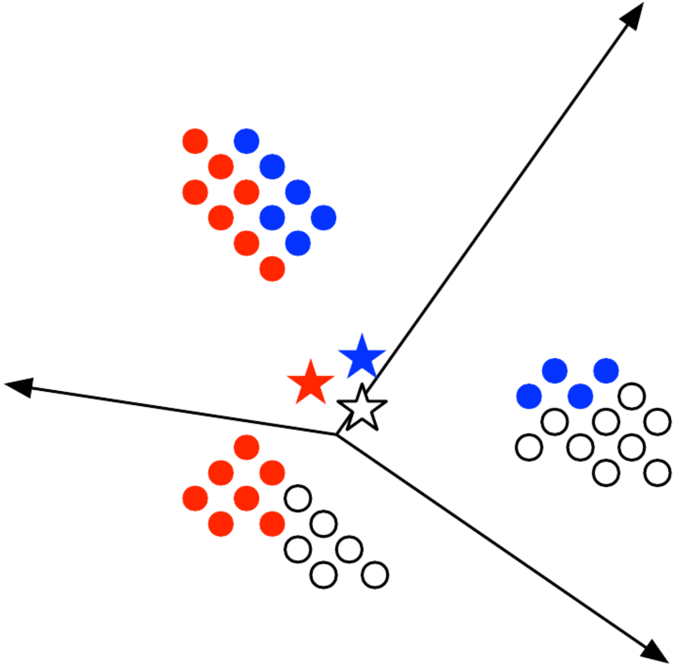
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step 2: Assign each example to its nearest cluster center.

K-Means Algorithm

Iterative Procedure

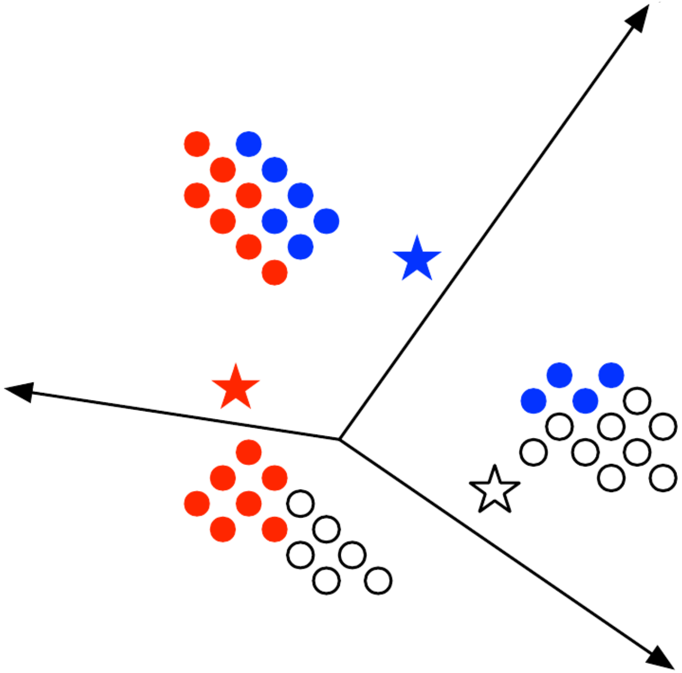
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step 3: Adjust the centroids to be the means of the samples assigned to them.

K-Means Algorithm

Iterative Procedure

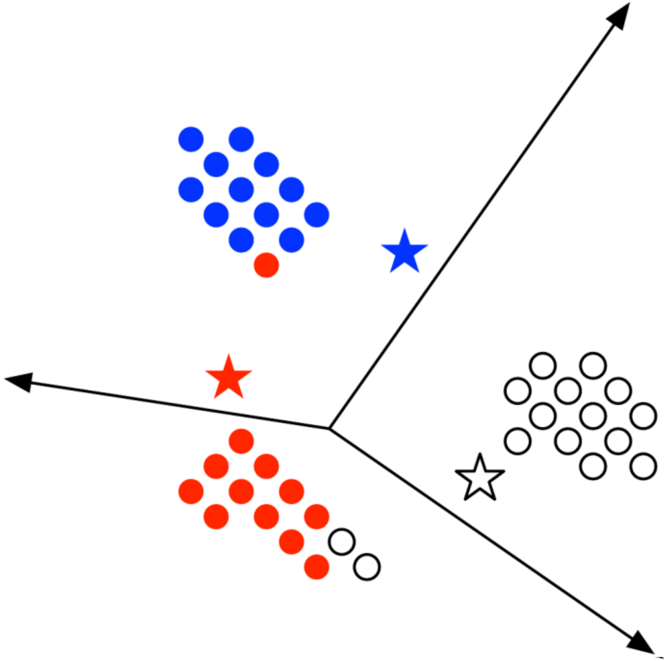
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step 4: Assign each example to its nearest cluster center.

K-Means Algorithm

Iterative Procedure

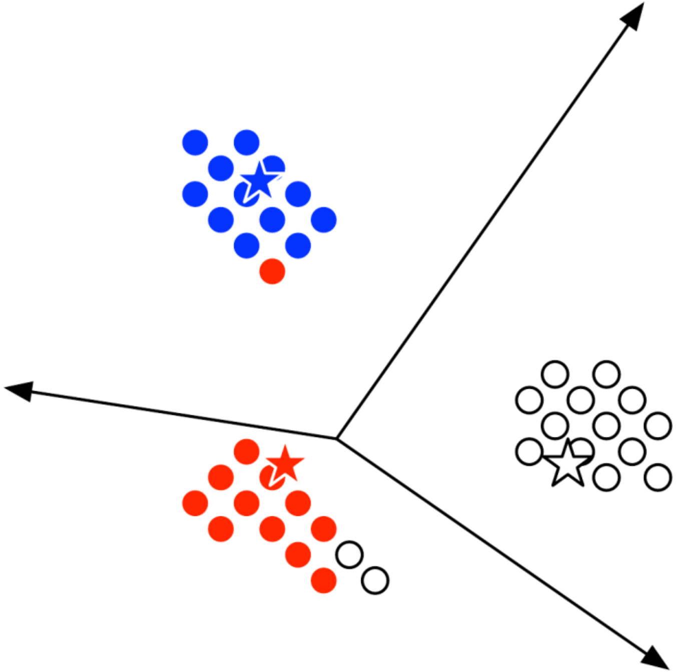
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step 5: Adjust the centroids to be the means of the samples assigned to them.

K-Means Algorithm

Iterative Procedure

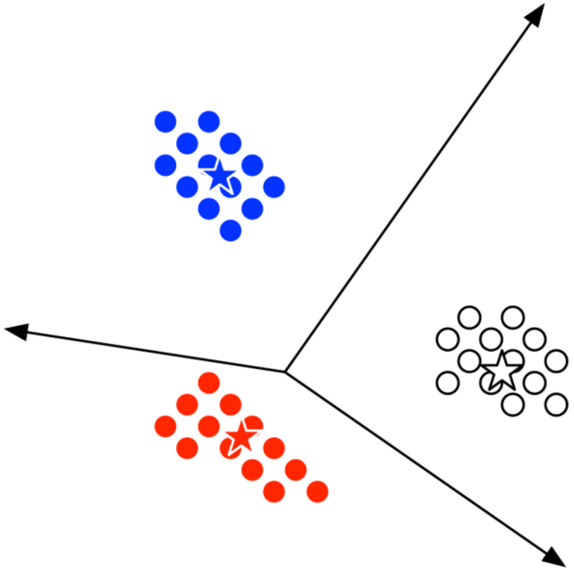
1. Pick K arbitrary cluster centers
2. Assign each sample to its closest centroid

$$z_n = \arg\min_k ||\boldsymbol{c}_k - \boldsymbol{x}_n||^2$$

3. Adjust the centroids to be the means of the samples assigned to them

$$\boldsymbol{c}_k = \frac{1}{|X_k|} \sum_{\boldsymbol{x}_i \in X_k} \boldsymbol{x}_i, \quad X_k = \{\boldsymbol{x}_n | z_n = k\}$$

4. Goto step 2 until no change



Step N: Convergence...

Does K-Means converge?

To analyze convergence, we write SSD in terms of assignments z_n

$$\mathrm{SSD}(C;\mathcal{D}) = \sum_{i=1}^{n} d(\boldsymbol{x}_i, c(\boldsymbol{x}_i))^2 = \sum_{i=1}^{n} \sum_{k} q_{nk} d(\boldsymbol{x}_i, \boldsymbol{c}_k)^2,$$

where $q_{nk} = \mathbb{I}(z_n = k)$ is 1 if the n-th example is assigned to the k-th cluster and 0 otherwise (1-hot coding)

- **Assignment Step:** Minimizes SSD w.r.t. z_n
  - Sets q_{nk} of nearest cluster to 1, all other values are 0
- **Adjustment Step:** Minimizes SSD w.r.t. centroids $\mathbf{c_k}$

$$\boldsymbol{c}_k = \frac{1}{\sum_n q_{nk}} \sum_{n=1}^{N} q_{nk} \boldsymbol{x}_n$$

  - Average vector is the vector with minimum squared distance to all assigned samples

K-Means analysis

Does K-Means converge? Yes, it (locally) minimizes the SSD!

- We have only a finite number of possible values for the centroid
- Every assignment or adjustment step is reducing the SSD (or it stays constant)

Does K-Means converge to the global minimal cost solution? No!

- The objective is an NP-Hard problem, so we can’t expect any algorithm to minimize the cost without essentially checking (near to) all assignments.
- It heavily depends on the initialisation of the centroids

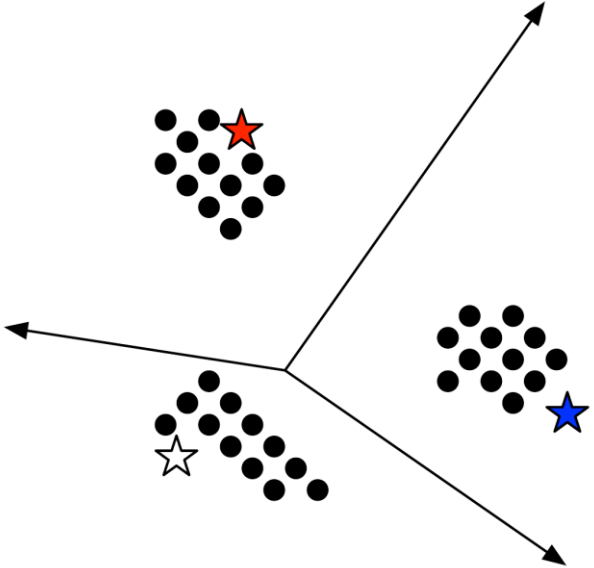
K-means++

Furthest First Initialization:

- Pick a random data-point as first center
- **for k ∈ {2, . . . , K} do**
  - find the example that is furthest from all previously selected means

$$\text{let } n = \arg\max_{n \in \{1,\ldots,N} \left(\min_{k' \in \{1,\ldots k-1\}} ||\boldsymbol{x}_n - \boldsymbol{c}'_k||^2 \right)$$

  - Assign centroid: $\boldsymbol{c}_k = \boldsymbol{x}_n$



Furthest first initialization in action...

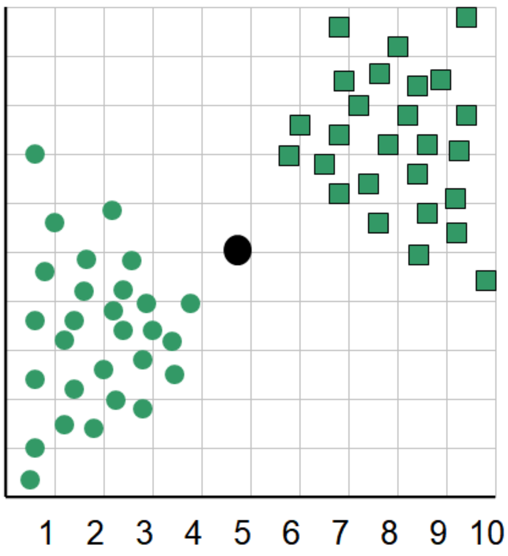
- Converges (in this case) after 1 adjustment

Number of clusters

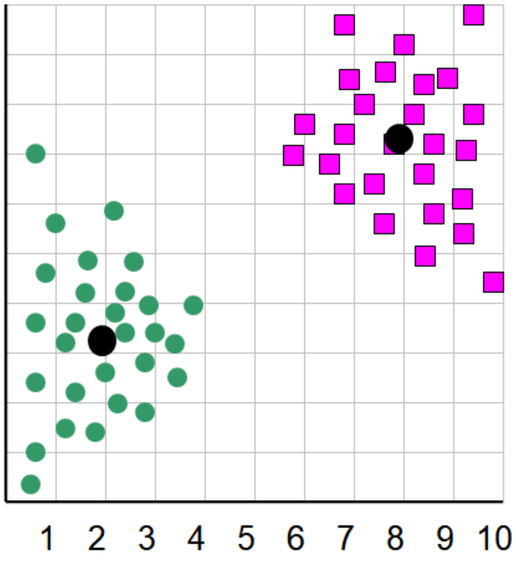
How to choose K?

- Based on ’good’ function value decrease on ’holdout’ set, cross validation (good but expensive)
- “Knee-finding method” (similar to PCA, heuristic but cheap)

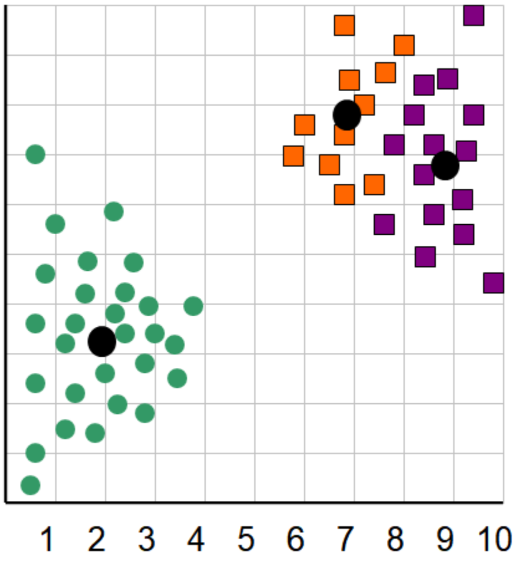
When k = 1, the objective function is 873.0



When k = 2, the objective function is 173.1

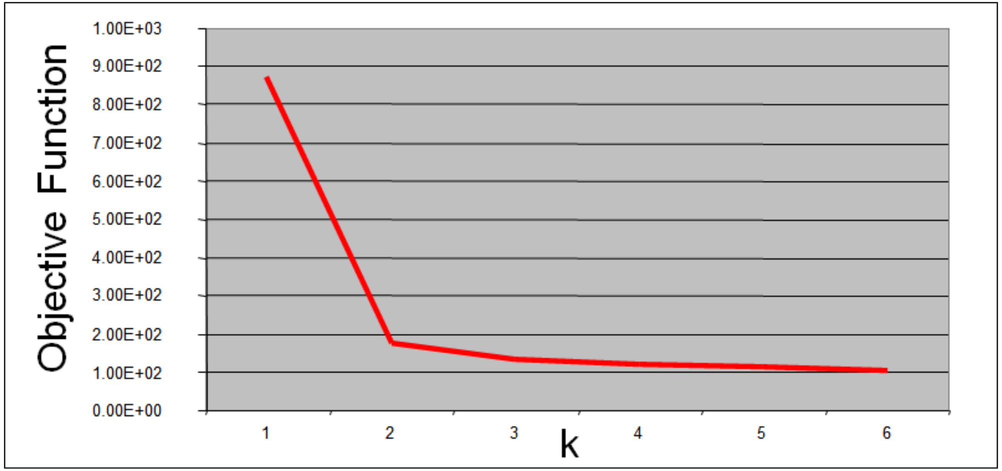


When k = 3, the objective function is 133.6



“Knee-finding” method

- We can plot the objective function (SSD) values for k equals 1 to 6…
- SSD will decrease with higher k (on average)
- The abrupt change at k = 2, is highly suggestive of two clusters in the data.
- This technique for determining the number of clusters is known as “knee finding” or “elbow finding”



Wrap-Up

Strengths:

- K-means usually converges very quickly in practice.
- K-means++ still not guaranteed to find the global optima
  - in practice, we can get stuck.
  - often try multiple initializations (use a little randomness in K-means++ and run the algorithm multiple times).

Weaknesses:

- Applicable only when mean is defined, then what about categorical data?
- Unable to handle noisy data and outliers
- Not suitable to discover clusters with non-convex shapes

Self-test Questions

What you should know now:

- How is the clustering problem defined? Why is it called “unsupervised”?
- How do hierarchical clustering methods work? What is the rule of the cluster-2-cluster distance and which distances can we use?
- How does the k-mean algorithm work? What are the 2 main steps?
- Why does the algorithm converge? What is it minimizing?
- Does k-means finds a the global minimum of the objective?