INF442
Algorithms for Data Analysis in C++

Steve Oudot

steve.oudot@polytechnique.edu

Data Science slides booklet

Introduction to Data Science

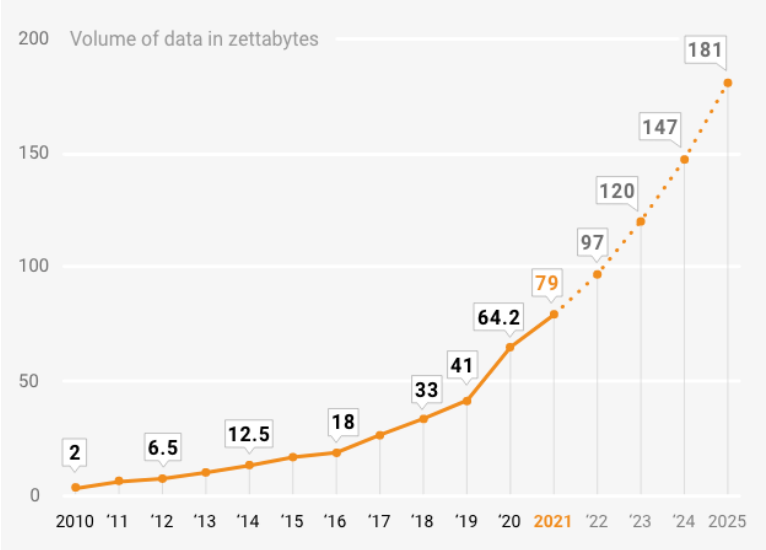
The big data era

Key figures:

- **size of 'global data sphere'** (including 10% of *unique data*)**:**

2 zB (2010) $\rightarrow$ 79 zB (2021) $\overset{\text{predict.}}{\rightarrow}$ 181 zB (2025) (1 zB $= 10^{21}$ Bytes)

— source: International Data Corporation



The big data era

Key figures:

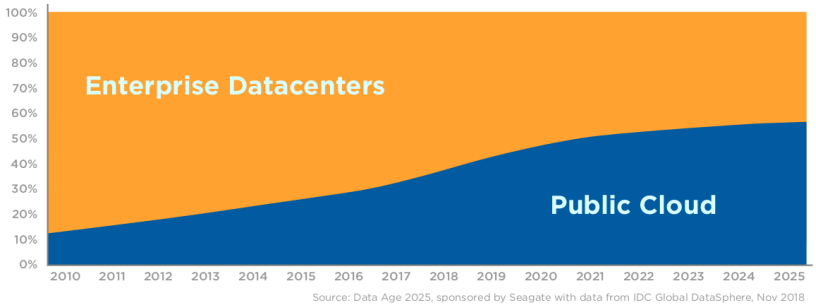
- **size of 'global data sphere'** (including 10% of *unique data*)**:**

2 zB (2010) $\rightarrow$ 79 zB (2021) $\overset{\text{predict.}}{\rightarrow}$ 181 zB (2025) (1 zB $= 10^{21}$ Bytes)

— source: International Data Corporation

- **correlated with World's storage capacity**

— data centers and cloud (45% - 55% in 2025)



The big data era

Key figures:

- **size of 'global data sphere'** (including 10% of *unique data*)**:**

2 zB (2010) $\rightarrow$ 79 zB (2021) $\overset{\text{predict.}}{\rightarrow}$ 181 zB (2025) (1 zB $= 10^{21}$ Bytes)

— source: International Data Corporation

- **correlated with World's storage capacity**

— data centers and cloud (45% - 55% in 2025)

- **exponential growth** ($+\,30\%$ each year on average)

— expected to be sustained on the long run

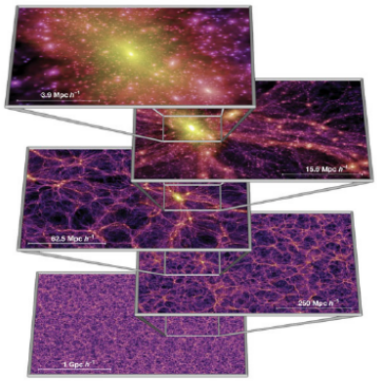
- small fraction of data is processed/analyzed

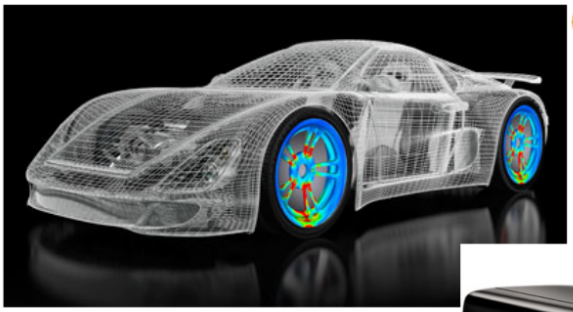
— **shortage of trained data scientists**

Data production

Data are produced at an unprecedented rate by:

- Industry / Economy
- Sciences
- End users

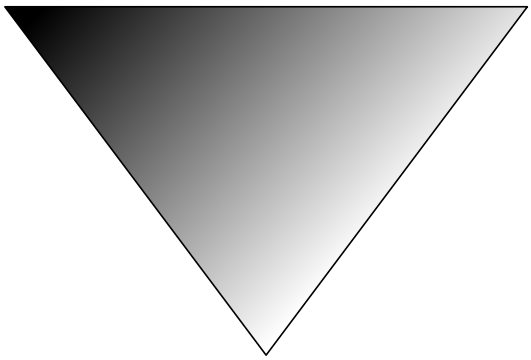




Challenges

Complex data

(non-linear, sparse, high-dimensional)



Corrupted data

(noise, outliers, missing values)

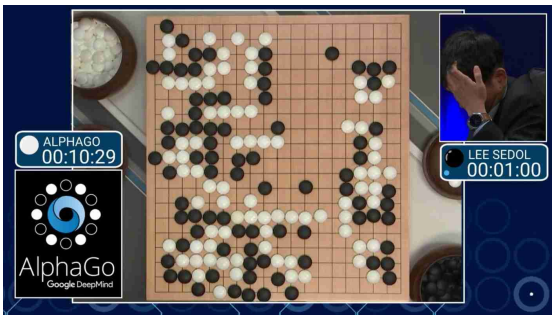
Big data

(streamed, online, distributed)

Data science's celebrated successes...

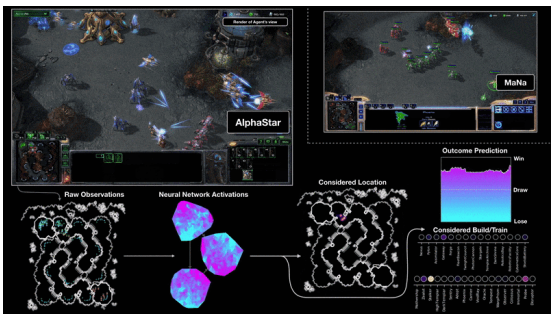
AI for games:

1997: IBM's *Deep Blue* wins chess match against world champion G. Kasparov



2016: DeepMind's *AlphaGo* wins Go match against 18-time world champion Lee Sedol

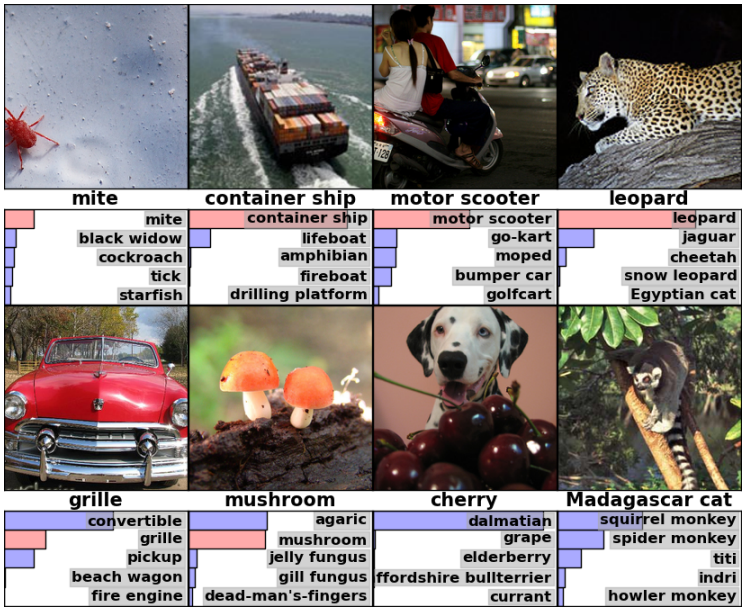
2019: DeepMind's *AlphaStar* beats Starcraft II professional players



Data science's celebrated successes...

ImageNet Challenge:

- database of $40 \cdot 10^6+$ images, structured in $20 \cdot 10^3+$ categories
- images collected on the Internet
- annotation process crowdsourced to Amazon Mechanical Turk

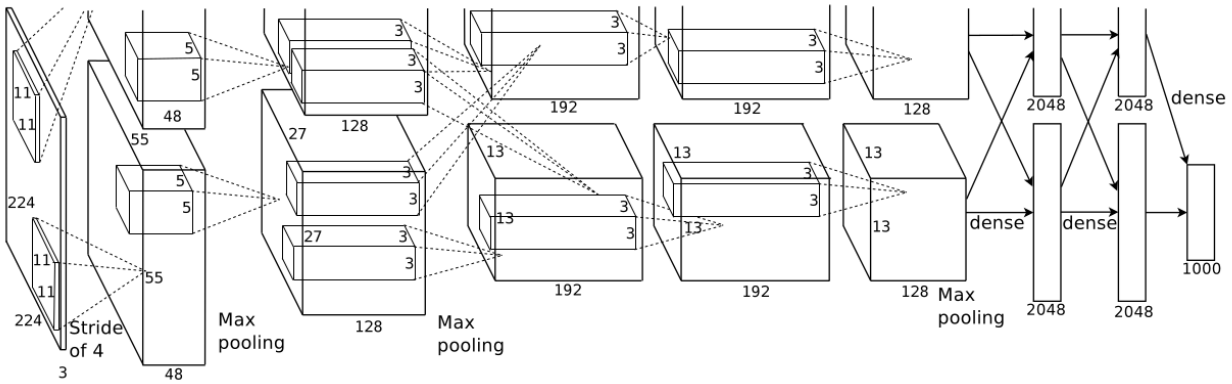


[J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei: ImageNet: A Large-Scale Hierarchical Image Database, CVPR 2009]

Data science's celebrated successes...

ImageNet Challenge:

- annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- until 2011, classification error rates around 25%
- 2012: breakthrough $\longrightarrow$ deep CNN (AlexNet) reduced error to 16%
- by now: error rates below 5%, performances better than human on narrow tasks

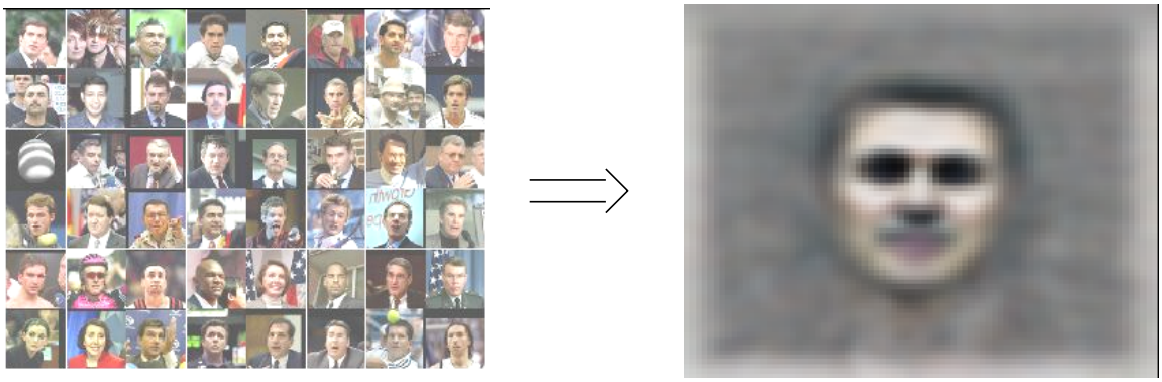


[Krizhevsky, Sutskever, Hinton: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012]

Data science's celebrated successes...

ImageNet Challenge:

- annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- until 2011, classification error rates around 25%
- 2012: breakthrough $\longrightarrow$ deep CNN (AlexNet) reduced error to 16%
- unsupervised pre-training leads to concept learning (e.g. human face, cat face)



[Le et al.: Building high-level features using large scale unsupervised learning, ICML 2012]

Data science's celebrated successes...

ChatGPT (Generative Pre-trained Transformer):

- chatbot released by OpenAI
- based on a Transformer neural network architecture
- pre-trained by Reinforcement Learning from Human Feedback (RLHF)

- able to generate realistic texts of various types: essays, poetry, recipes, code, etc.

- able to make accurate text translations, summaries, sentiment analysis, etc.

... and notorious failures

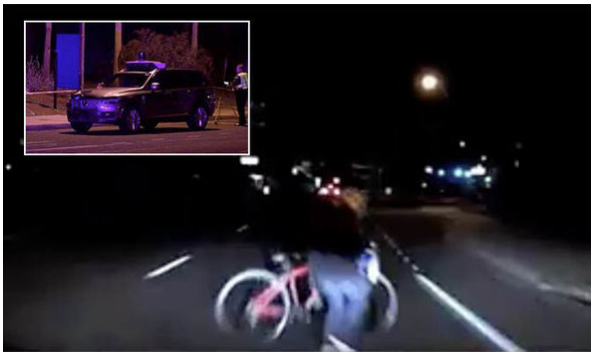
Microsoft's Tay:

- AI-powered chat bot, launched on Twitter (`@TayandYou`) on March 23, 2016
- learned from its interactions with people
- shut down only 16 hours after launch
- produced inflammatory, offensive (racist, sexually-charged) tweets
- training overrun by trolls
- numerous questions raised (technical, legal, ethical)

... and notorious failures

Other notorious recent AI failures:

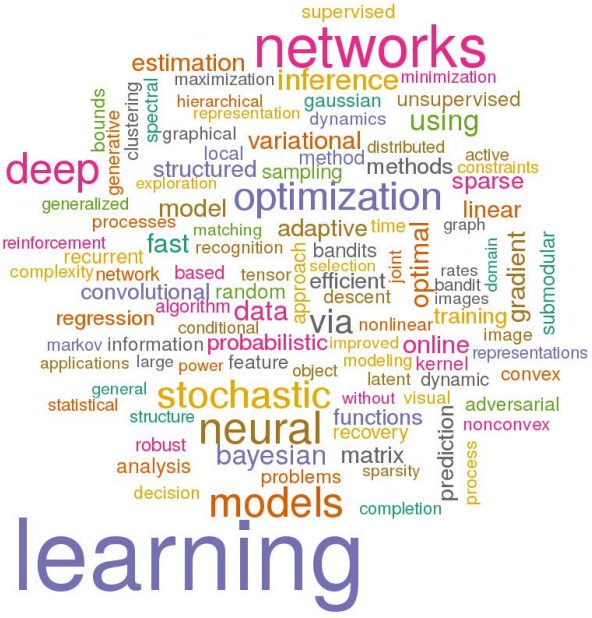
Amazon's AI recruiting tool proven to be gender-based



Uber's self-driving car kills pedestrian in Arizona

What is data science?

Aim: dev. tools to store, manipulate, analyze / extract knowledge from data



word cloud of paper titles at NIPS 2016

(source: `http://www.kaggle.com/benhamner/nips-papers`)

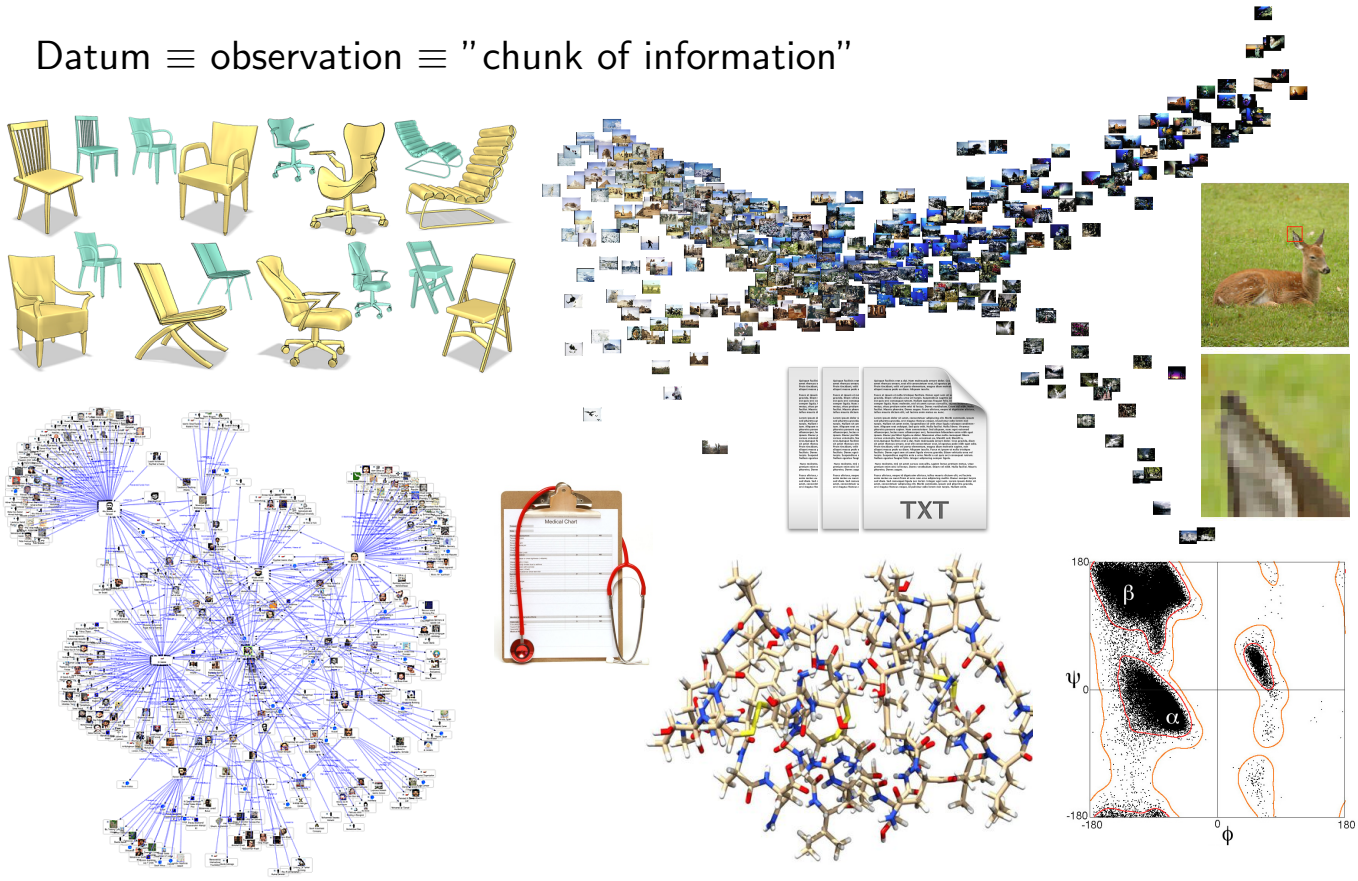
What is data science?

Core topics:

- statistical analysis
- machine learning / deep learning
- pattern recognition
- data mining
- optimization (convex / combinatorial)
- database management and distributed systems
- high-performance computing (streaming, distributed, cloud)

Data?

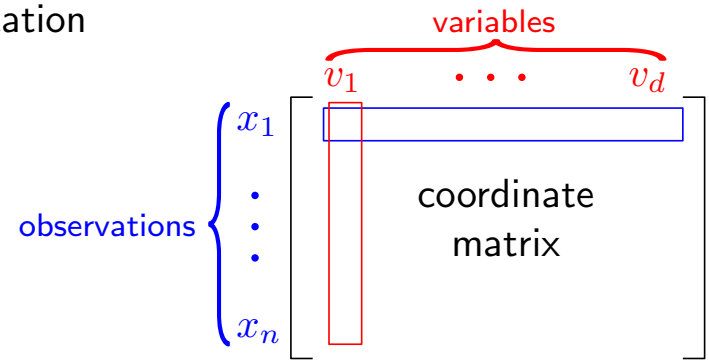
Datum ≡ observation ≡ "chunk of information"



Data?

Datum ≡ observation ≡ "chunk of information"

Vector representation



categorical variables: $1, 2, \cdots, K$ (arbitrary labels)

(e.g. "cat", "dog", "horse")

continuous variables: real or complex values

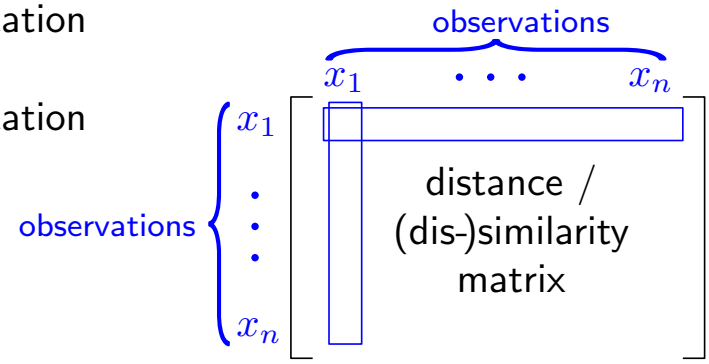
(e.g. temperature, pressure, geographic coordinate, income, amplitude/phase)

Data?

Datum ≡ observation ≡ "chunk of information"

Vector representation

Metric representation



distances: Euclidean, Hamming, geodesic, diffusion, edit, Jaccard, Wasserstein, etc.

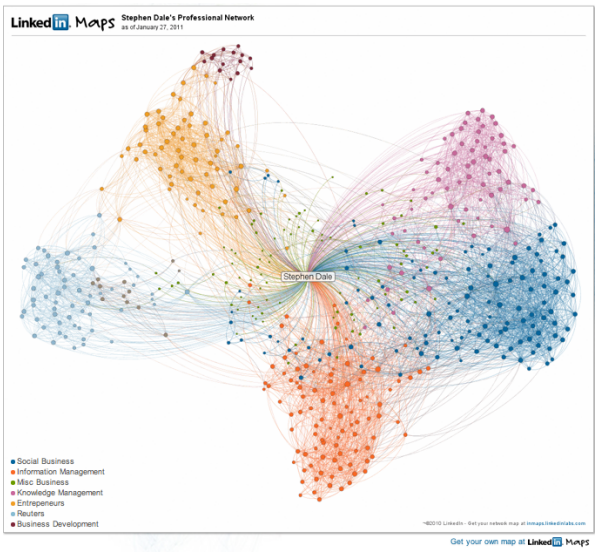
(dis-)similarity measures: cosine, Kullback-Leibler, Bregman divergences, etc.

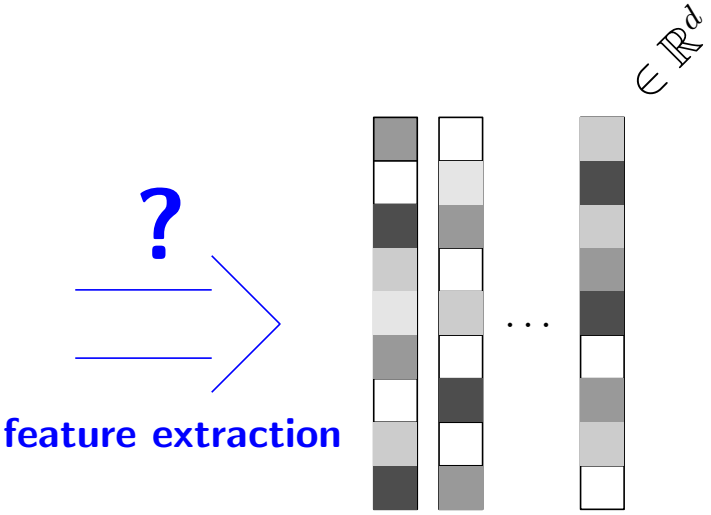
Data?

Datum ≡ observation ≡ "chunk of information"

Vector representation

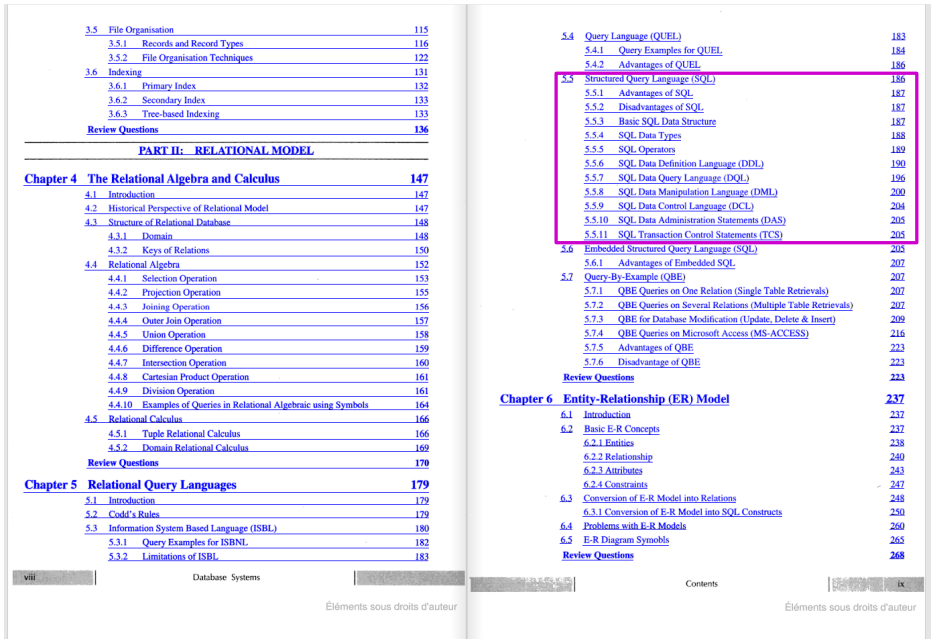
Metric representation

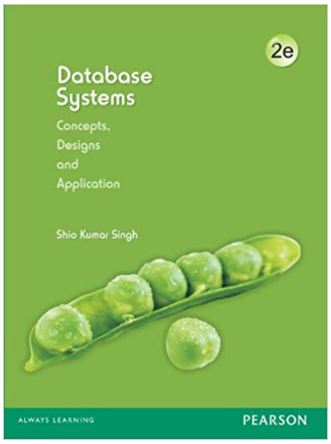




Programming languages for data science

- Databases / data manipulation: Structured Query Language (SQL)

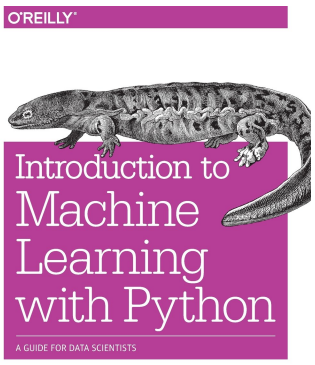


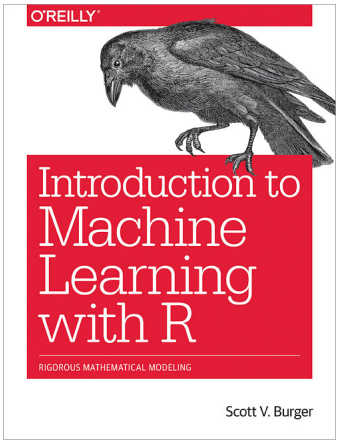


Programming languages for data science

- Databases / data manipulation: Structured Query Language (SQL)

- Data analysis: Python (CS) / R (stats)

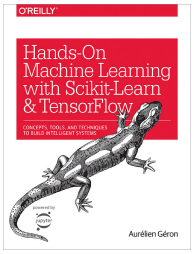


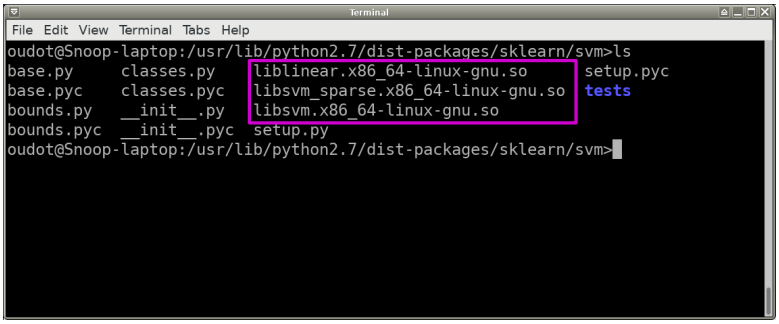


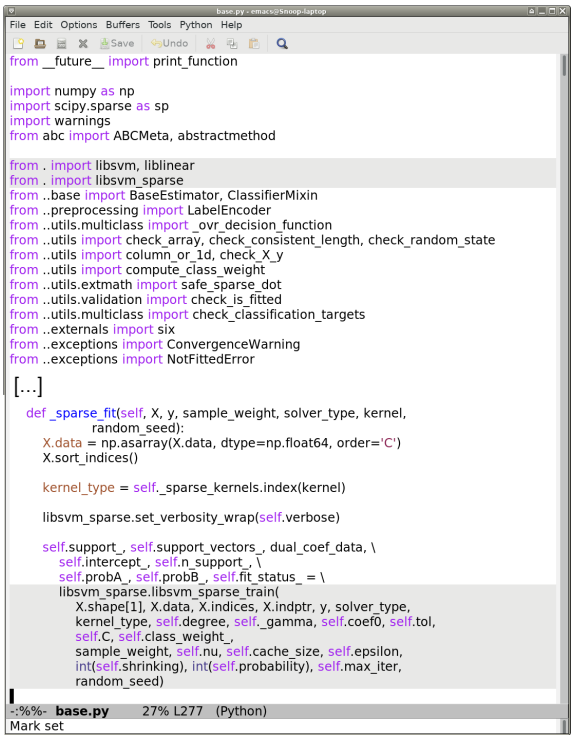
Programming languages for data science

- Databases / data manipulation: Structured Query Language (SQL)
- Data analysis: Python (CS) / R (stats)
- Effective data processing:

  C / C++ / CUDA (GPGPU)







Learning paradigms

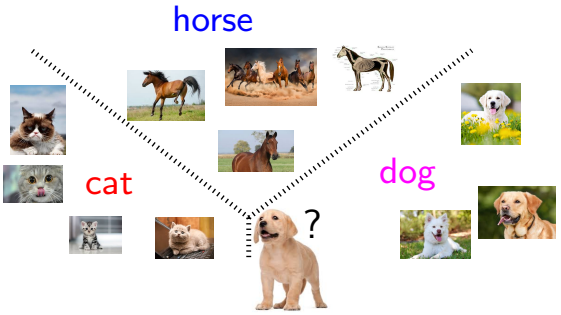
Supervised learning

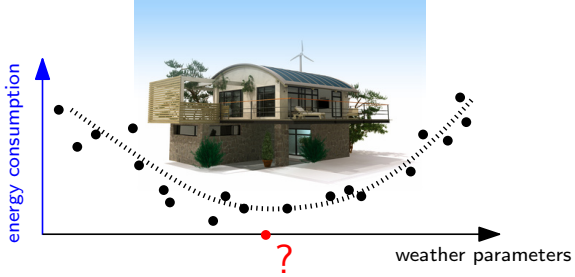
Input: data with labels (examples)

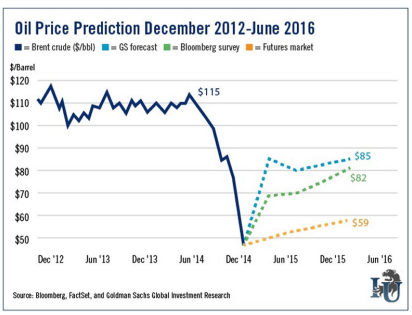
Goal: predict the labels of new data

Typical problems:

- classification (categorical labels)
- regression (continuous labels)
- forecasting (regression on time series)







Learning paradigms

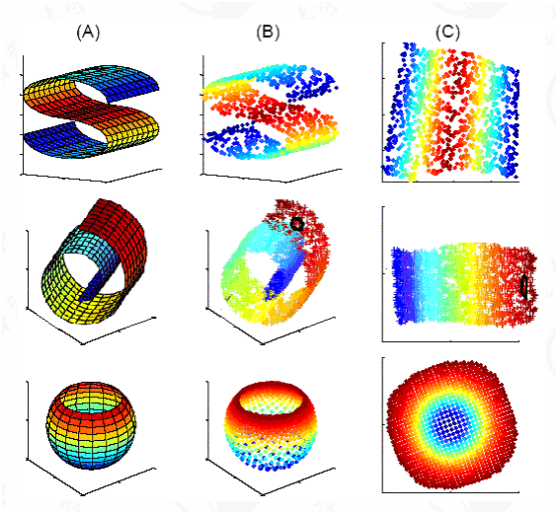
Unsupervised learning

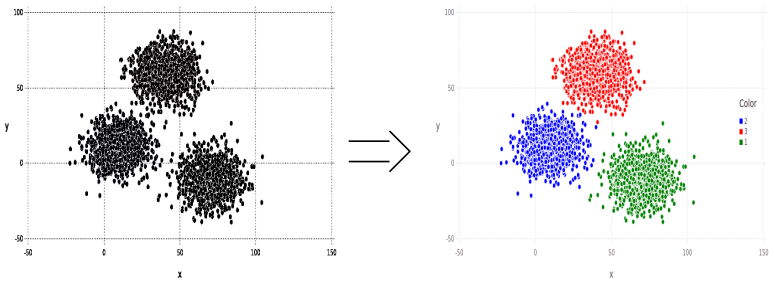
Input: data without labels

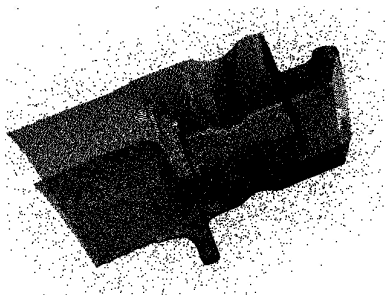
Goal: identify patterns, correlations

Typical problems:

- ▶ clustering
- ▶ dimensionality reduction
- ▶ anomaly detection / noise removal



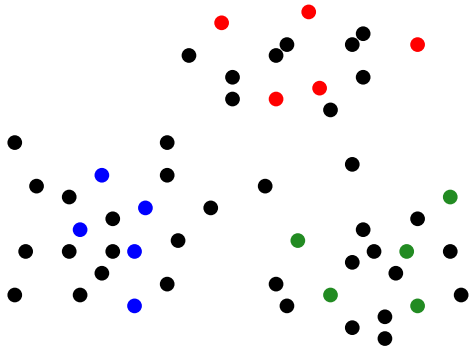




Learning paradigms

Unsupervised learning

Semi-supervised learning (only a fraction of the input data has labels)



Supervised learning

Learning paradigms

Reinforcement learning

Input: Markov decision process:

- agent & environment states, vis. rules, actions, transition probabilities, rewards

Goal: find policy that minimizes the *regret*

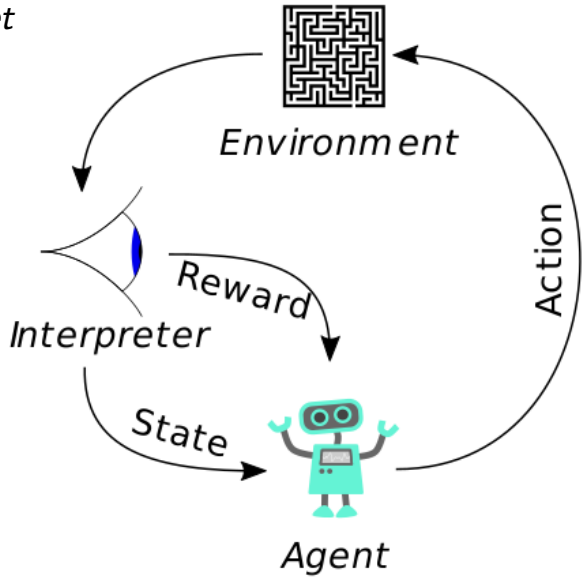
(expected loss of reward compared to optimal strategy)

Typical problems:

- exploration vs. exploitation

  (e.g. multi-armed bandit)

- control learning



Learning paradigms

Reinforcement learning

Input: Markov decision process:

- agent & environment states, vis. rules, actions, transition probabilities, rewards

Goal: find policy that minimizes the *regret*

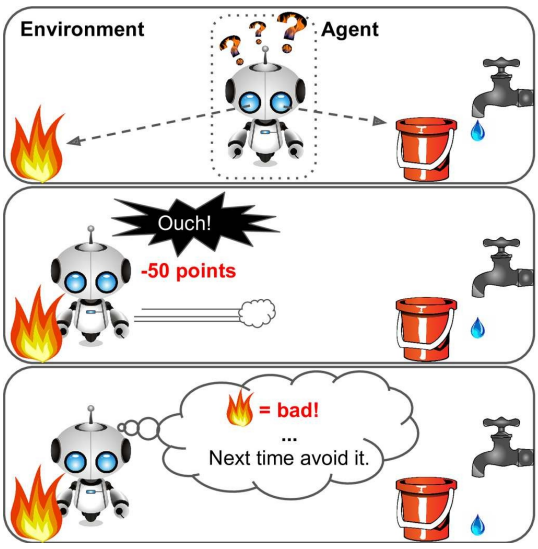
(expected loss of reward compared to optimal strategy)

Typical problems:

- exploration vs. exploitation

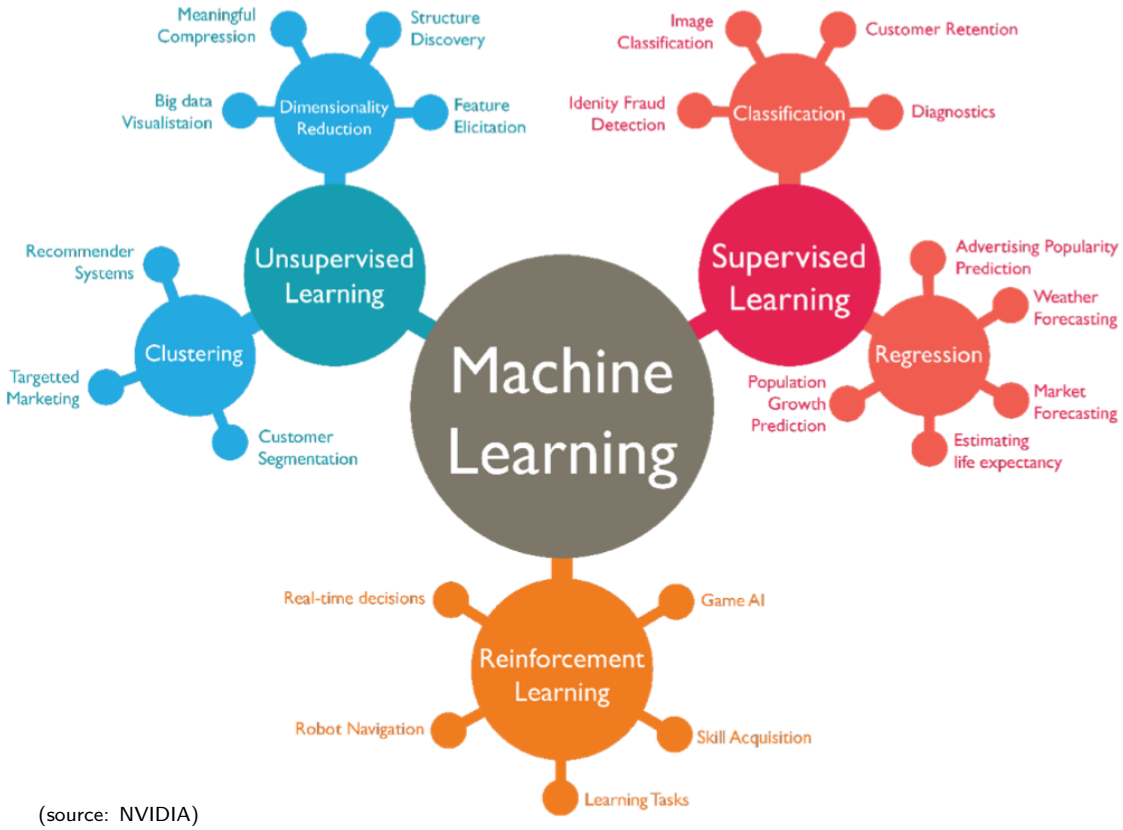
  (e.g. multi-armed bandit)

- control learning



[Géron 2017]

Learning paradigms



(source: NVIDIA)

Nearest-Neighbors Search

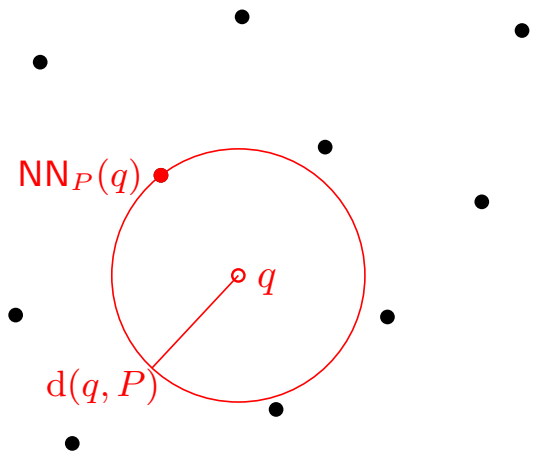
Outline:

- **Problem statement**
- **Naive approach: linear scan**
- **Challenges and popular approaches**
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Outline

- **Problem statement**
- **Naive approach: linear scan**
- **Challenges and popular approaches**
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Nearest neighbor search



pre-processing input: P

query input: q

goal: find $p \in \mathrm{NN}_P(q)$

$\mathrm{d}(q,p) = \min_{p' \in P} \mathrm{d}(q,p')$

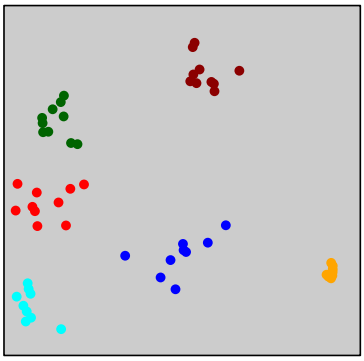
Nearest neighbor search

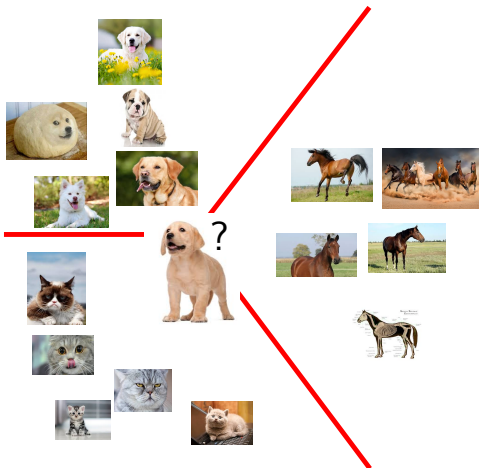
Variants:

- k-nearest neighbors: find the k points closest to q in P
- r-nearest neighbor: find a point $p \in P$ such that $\mathrm{d}(q,p) \leq r$
- metrics:
  - ℓ_2, ℓ_p, ℓ_∞
  - strings: Hamming distance
  - images: optimal transport distances
  - point clouds: (Gromov-)Hausdorff distances
  - proteins: RMSD distances
  - ...

A fundamental problem across data sciences

- clustering, e.g. k-means, mean-shift
- information retrieval in databases
- information theory, e.g. vector quantization
- supervised learning, e.g. NN-classifiers
- ...





Outline

- Problem statement
- **Naive approach: linear scan**
- Challenges and popular approaches
- k-d trees:
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Linear scan

Input: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$, $q \in \mathbb{R}^d$

$\mathrm{d}_{\min} := \infty$ (dist. to nearest neighbor among the pts viewed so far)

for $i = 1$ to n do:

$\quad \mathrm{d}_{\min} := \min\{\mathrm{d}_{\min}, \mathrm{d}(q, p_i)\}$

return $\mathrm{d}_{\min}$ or index i that achieves $\mathrm{d}_{\min}$

Complexity:

space: $O(d\,n)$ — n points, d coordinates each

time: $O(d\,n)$ — n iterations, 1 distance computation each

Outline

- **Problem statement**
- **Naive approach: linear scan**
- **Challenges and popular approaches**
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Strategy and challenges

Strategy:

- preprocess the n point of P in $\mathbb{R}^d$ into some data structure DS for fast nearest-neighbor queries answers

Ideal wish list:

- DS should have linear size in n and polynomial size in d
- a query should take sublinear time in n and polynomial time in d

  e.g. binary search trees in $d = 1$: linear size, $O(\log n)$ time

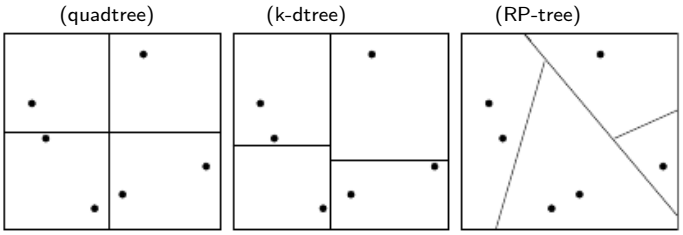
Core difficulties:

- *Curse of dimensionality*: hard to outperform linear scan in high d
- Interpretation: meaningfulness of distances in high d (concentration)

Popular approaches

- Linear scan $O(dn)$ space and time
- Voronoi diagrams
- Tree-like data structures
  - quadtrees (split at midpoint in all coordinates)
  - tries / dyadic trees (split at mean, cycle around coordinates)
  - kd-trees (split at median, cycle around coordinates)
  - Random Projection trees (split at median along random coordinates)
  - PCA trees (split at median along 1st eigenvector of covariance matrix)
  - ...
- Locality Sensitive Hashing

Binary Space Partitions

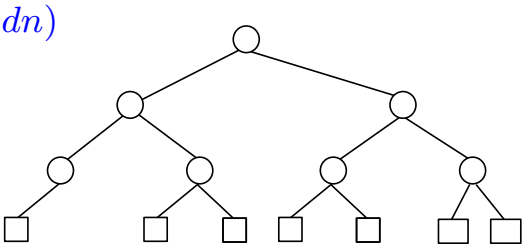


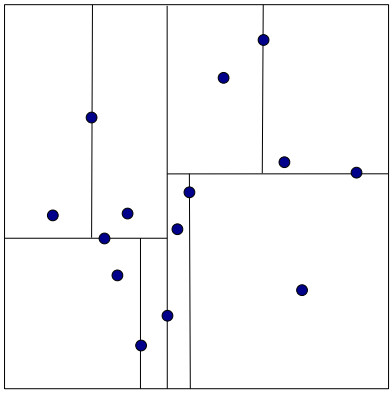
Outline

- **Problem statement**
- **Naive approach: linear scan**
- **Challenges and popular approaches**
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

kd-tree

- a binary tree encoding a hierarchy of binary space partitions
- each internal node implements a binary spatial partition induced by a hyperplane H, dividing the point cloud into three subsets:
  - ▶ node's data: a distinguished point lying on H
  - ▶ right subtree: all points lying on one side of H
  - ▶ left subtree: all other points
- subdivision stops whenever fewer than n_0 remain
  $\rightsquigarrow$ size: $O(dn)$

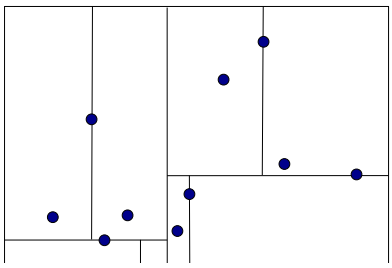




$(n_0 = 1)$

kd-tree

- a binary tree encoding a hierarchy of binary space partitions
- each internal node implements a binary spatial partition induced by a hyperplane H, dividing the point cloud into three subsets:
  - ▶ node's data: a distinguished point lying on H
  - ▶ right subtree: all points lying on one side of H
  - ▶ left subtree: all other points
- subdivision stops whenever fewer than n_0 remain
  $\rightsquigarrow$ size: $O(dn)$

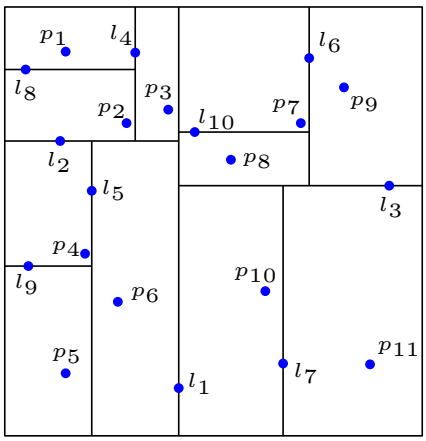


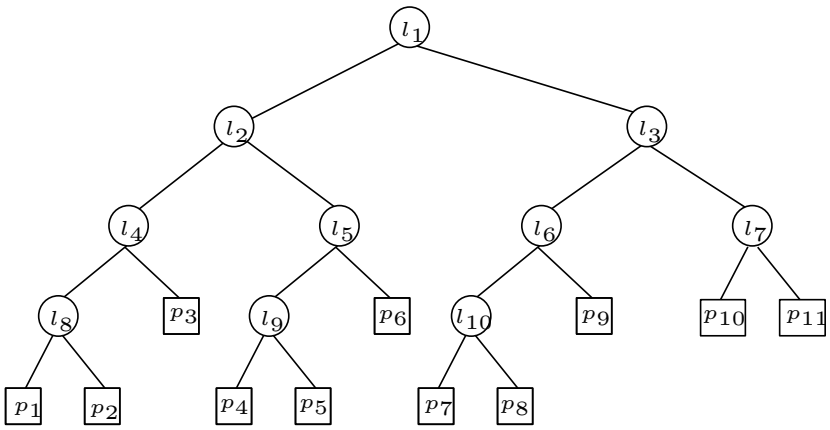
kd-tree specifics:

H is **orthogonal to coordinate axis** (possible choices: cyclic iteration, max spread)

H goes through the **median** in the considered coordinate

Example





l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Outline

- Problem statement
- Naive approach: linear scan
- Challenges and popular approaches
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Recursive construction

2 types of nodes:

- leaf: contains a batch of $\leq n_0$ points
- internal: contains a coordinate axis and value for splitting, plus a point and refs. to 2 children

build (P, c): (P = current point cloud, c = coordinate selected for splitting)

if $\#P \leq n_0$ then create a leaf storing P

else:

compute median m of $\{p[c] : p \in P\}$ and $p_* \in P$ such that $p_*[c] = m$

compute $P_l = \{p \in P \mid p[c] \leq m\} \setminus \{p_*\}$ and $P_r = \{p \in P \mid p[c] > m\}$

create an internal node storing c, m, p_*, and refs to the nodes given by

(cyclic iteration) ⟶ build (P_l, $(c+1)\%d$) and build (P_r, $(c+1)\%d$)

Recursive construction

2 types of nodes:

- leaf: contains a batch of $\leq n_0$ points
- internal: contains a coordinate axis and value for splitting, plus a point and refs. to 2 children

Complexity analysis:
1 call to build on each node of the tree
$\Rightarrow O(n)$ calls (1 point charged per node)
$O(\text{median}(\#P) + \#P)$ for each call

build (P, c): (P = current point cloud, c = coordinate selected for splitting)

if $\#P \leq n_0$ then create a leaf storing P

else:

compute median m of $\{p[c] : p \in P\}$ and $p_* \in P$ such that $p_*[c] = m$

compute $P_l = \{p \in P \mid p[c] \leq m\} \setminus \{p_*\}$ and $P_r = \{p \in P \mid p[c] > m\}$

create an internal node storing c, m, p_*, and refs to the nodes given by

(cyclic iteration) ⟶ build $(P_l, (c+1)\%d)$ and build $(P_r, (c+1)\%d)$

Recursive construction

Median computation:

- by sorting the points of the current cloud P
  - median$(\#P)$ is then in $O(\#P \log \#P) = O(\#P \log n)$
  - total complexity: $C(n) = n \log n + 2\,C(n/2) \rightsquigarrow O(n \log^2 n)$
- by pre-sorting all the points along each coordinate
  - median$(\#P)$ is then in $O(\#P)$ after $O(d\,n \log n)$ preprocessing
  - total complexity: $C(n) = d\,n + 2\,C(n/2) \rightsquigarrow O(d\,n \log n)$
- by linear median (randomized or deterministic, cf. INF562)
  - median$(\#P)$ is then in $O(\#P)$ with no preprocessing
  - total complexity: $C(n) = n + 2\,C(n/2) \rightsquigarrow O(n \log n)$

Outline

- Problem statement
- Naive approach: linear scan
- Challenges and popular approaches
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

Usage for NN search

Strategy 1: defeatist search

$\mathrm{d}_{\min} := \infty$ (min dist. to pts viewed so far)

search $(node)$: ($node = root$ initially)

if $node = leaf$:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \min_{p \in node.batch} \mathrm{d}(q, p)\}$

else:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \mathrm{d}(q, node.point)\}$

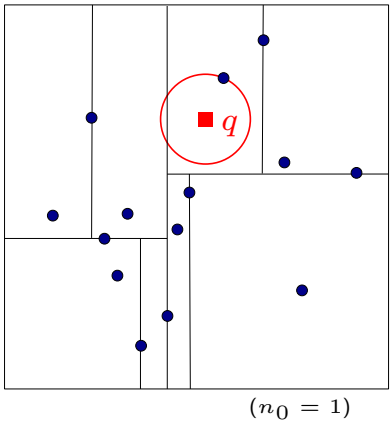
defeatist

if q on "left" side of $node.H$

search $(node.left)$

else (q on "right" side of $node.H$)

search $(node.right)$



$(n_0 = 1)$

Usage for NN search

Strategy 1: defeatist search

$\mathrm{d}_{\min} := \infty$ (min dist. to pts viewed so far) Query time: $O(d\,(n_0 + \log \frac{n}{n_0}))$

search $(node)$: ($node = root$ initially) May fail!

if $node = leaf$:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \min_{p \in node.batch} \mathrm{d}(q,p)\}$

else:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min}, \mathrm{d}(q, node.point)\}$

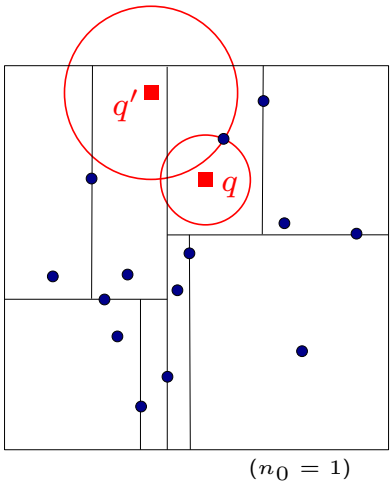
defeatist

if q on "left" side of $node.H$

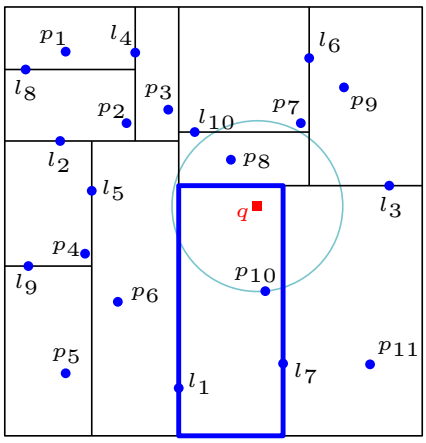
search $(node.left)$

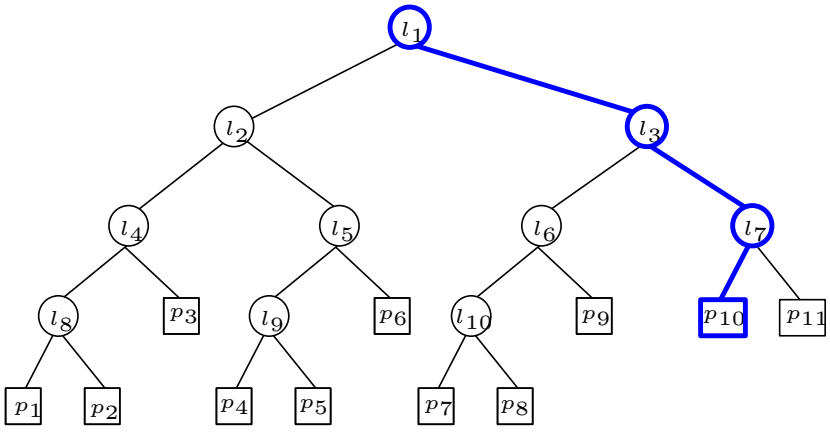
else (q on "right" side of $node.H$)

search $(node.right)$



Example





l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Usage for NN search

Strategy 2: backtracking search

$\mathrm{d}_{\min} := \infty$ (min dist. to pts viewed so far)

search $(node)$: ($node = root$ initially)

if $node = leaf$:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \min_{p \in node.batch} \mathrm{d}(q,p)\}$

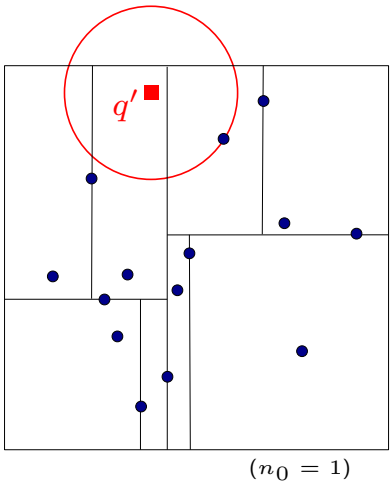
else:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \mathrm{d}(q, node.point)\}$

backtracking

if $B(q, \mathrm{d}_{min})$ intersects "left" side of $node.H$
search $(node.left)$

if $B(q, \mathrm{d}_{\min})$ intersects "right" side of $node.H$
search $(node.right)$



Usage for NN search

Strategy 2: backtracking search

$\mathrm{d}_{\min} := \infty$ (min dist. to pts viewed so far)

search $(node)$: ($node = root$ initially)

if $node = leaf$:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \min_{p \in node.batch} \mathrm{d}(q,p)\}$

else:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \mathrm{d}(q, node.point)\}$

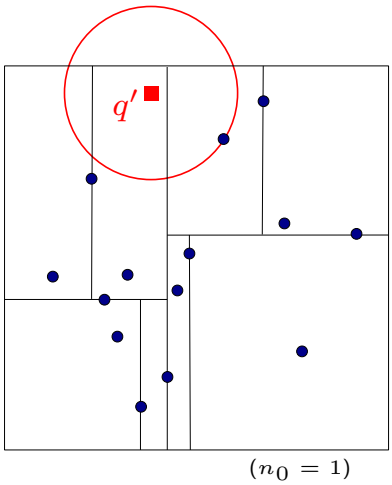
backtracking

if $B(q, \mathrm{d}_{min})$ intersects "left" side of $node.H$
search $(node.left)$

if $B(q, \mathrm{d}_{\min})$ intersects "right" side of $node.H$
search $(node.right)$

Always succeeds

$\mathrm{d}_{\min} \geq \mathrm{d}(q, \mathrm{NN}(q)) \Rightarrow B(q, \mathrm{d}_{\min})$ intersects all cells containing $\mathrm{NN}(q)$ in subdivision throughout search



Usage for NN search

Strategy 2: backtracking search

$\mathrm{d}_{\min} := \infty$ (min dist. to pts viewed so far)

search $(node)$: ($node = root$ initially)

if $node = leaf$:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min},\ \min_{p \in node.batch} \mathrm{d}(q,p)\}$

else:

$\mathrm{d}_{\min} := \min\{\mathrm{d}_{\min}, \mathrm{d}(q, node.point)\}$

backtracking

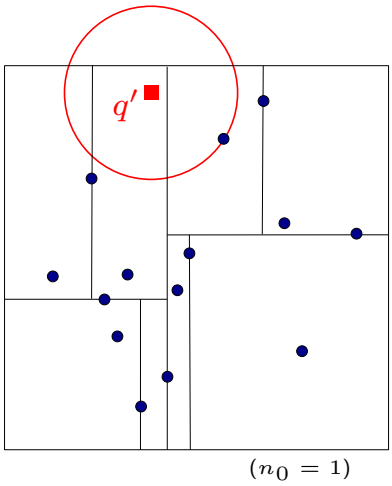
if $B(q, \mathrm{d}_{min})$ intersects "left" side of $node.H$
search $(node.left)$

if $B(q, \mathrm{d}_{\min})$ intersects "right" side of $node.H$
search $(node.right)$

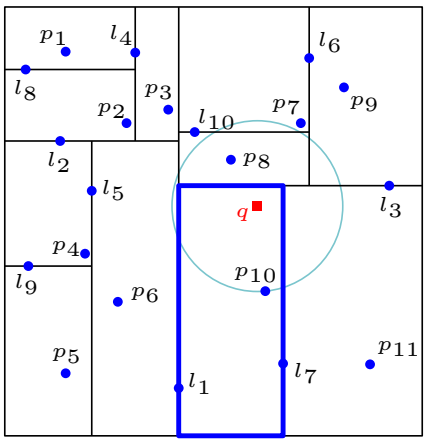
Always succeeds

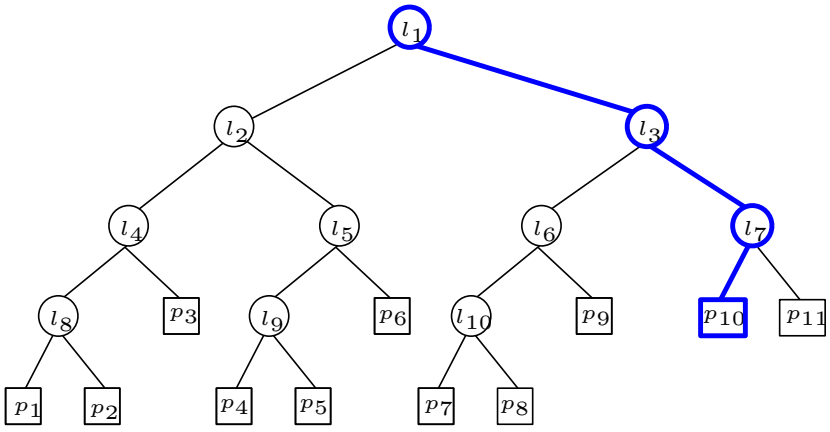
Query time may be up to linear

(all cells visited)



Example



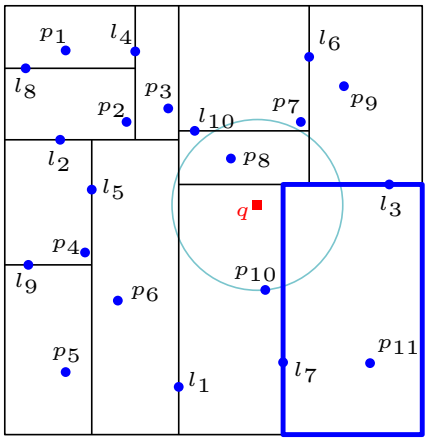


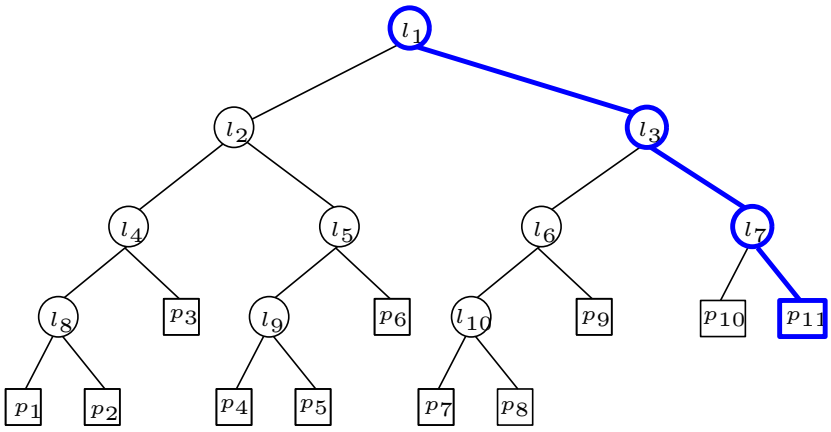
l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Example



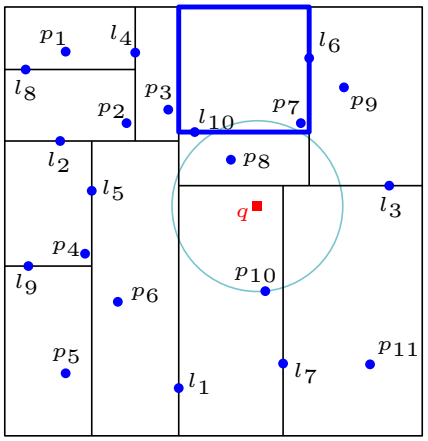


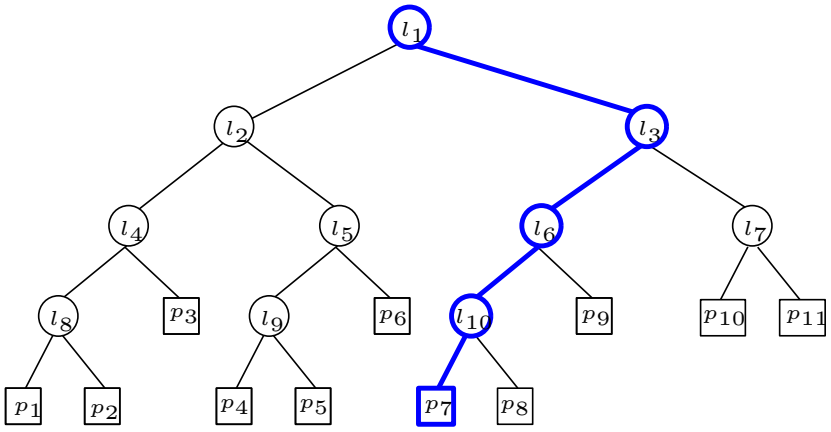
l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Example



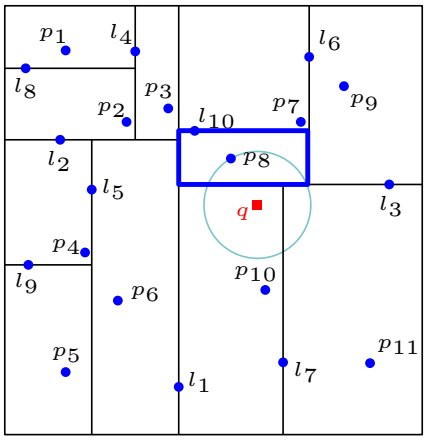


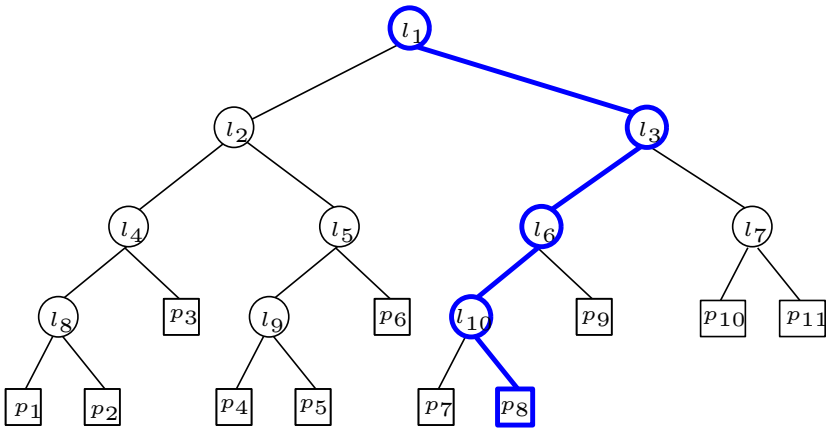
l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Example



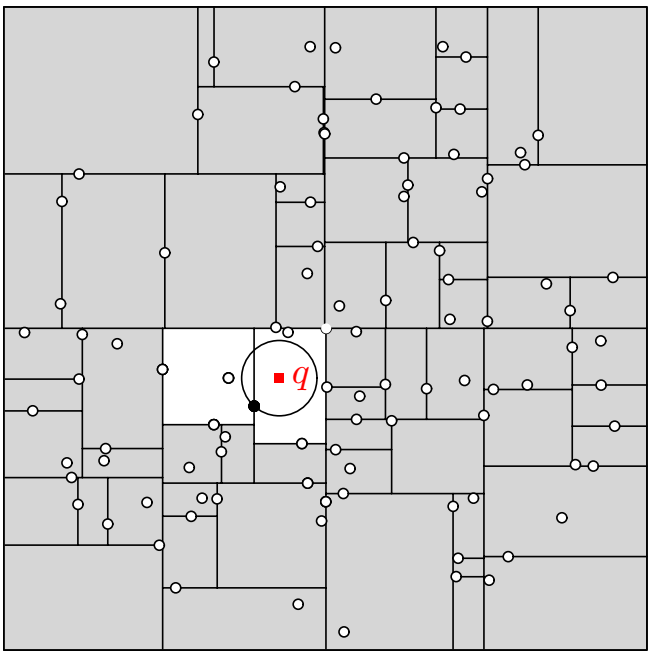


l_i: data at internal node

p_i: data at leaf node

(note: left-right labels are arbitrary)

Example



best-case input (unif. distrib.):

most cells are fat

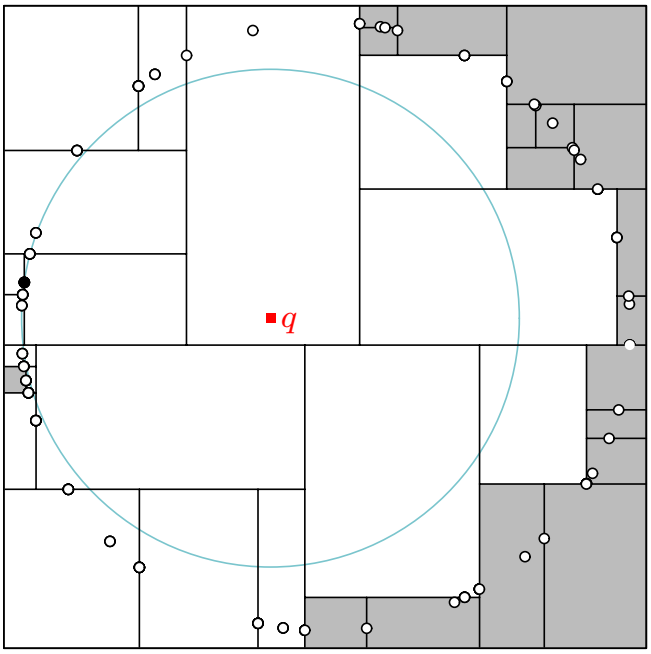
$\Downarrow$

query time $= O(c_d \log n)$

$c_d \approx 2^d$

[Friedman et al.: An algorithm for finding best matches in logarithmic expected time, 1977]

Example



worst-case input (non-unif. distrib.):

many skinny cells

$\Downarrow$

query time $= \Omega(d\,n)$

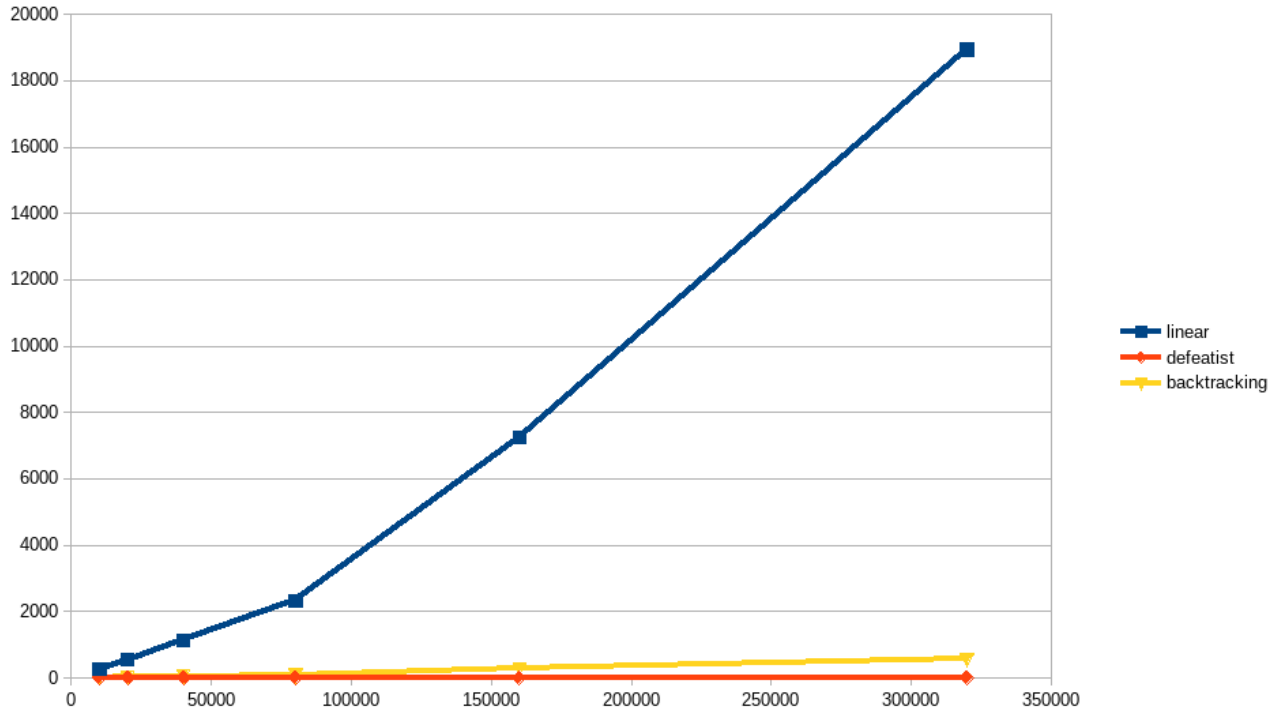
Variants: priority search, early backtracking, random cutting hyperplanes, etc.

Outline

- Problem statement
- Naive approach: linear scan
- Challenges and popular approaches
- **k-d trees:**
  - definition
  - construction
  - usage for NN search: defeatist search vs. backtracking search
  - benchmarks and curse of dimensionality

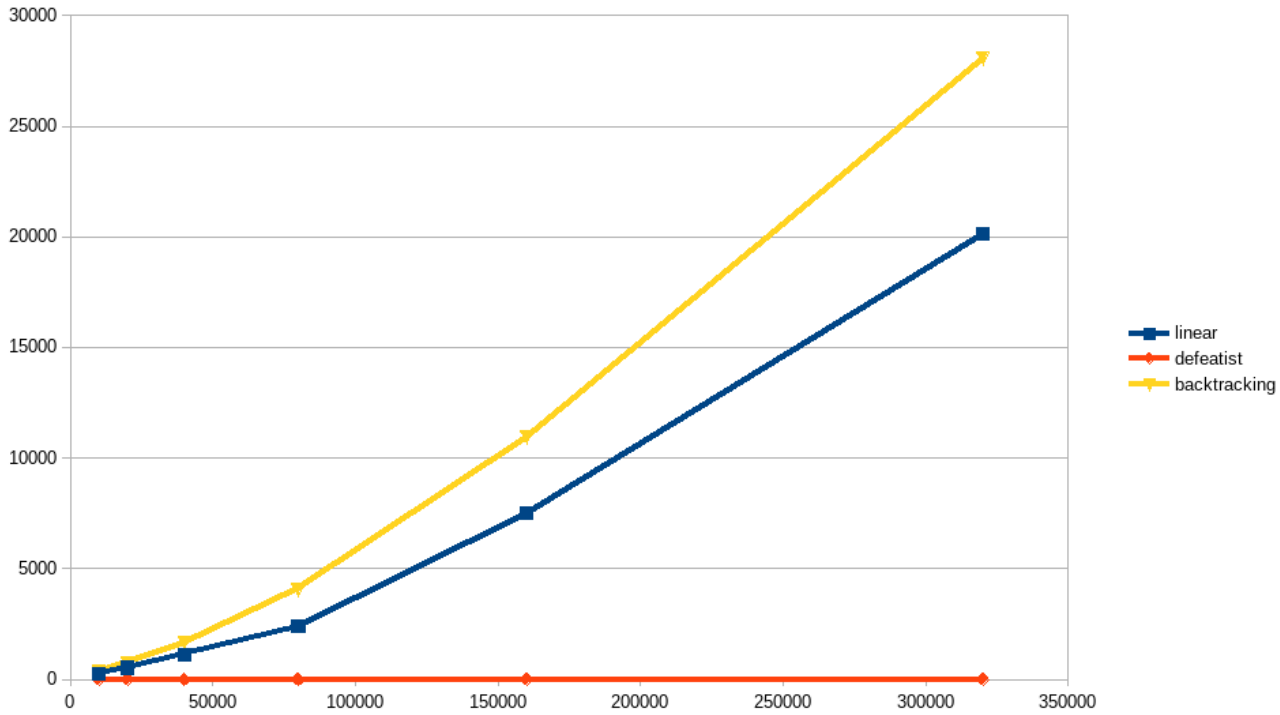
Benchmarks

avg. query time (μs) vs. # data points: (uniform measure in unit square in 2d)

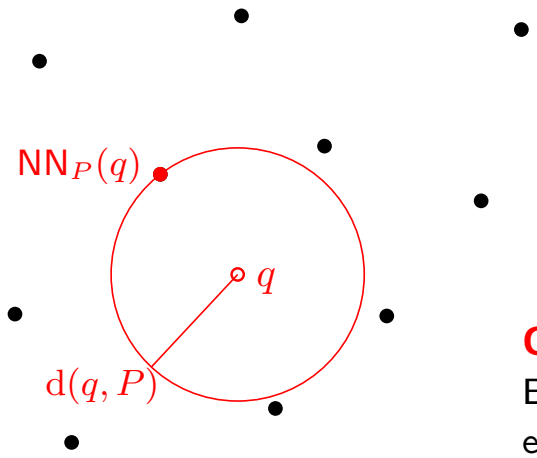


Benchmarks

avg. query time (μs) vs. # data points: (uniform measure on unit circle in 2d)



High dimensions



pre-processing input: $P \subset \mathbb{R}^d$

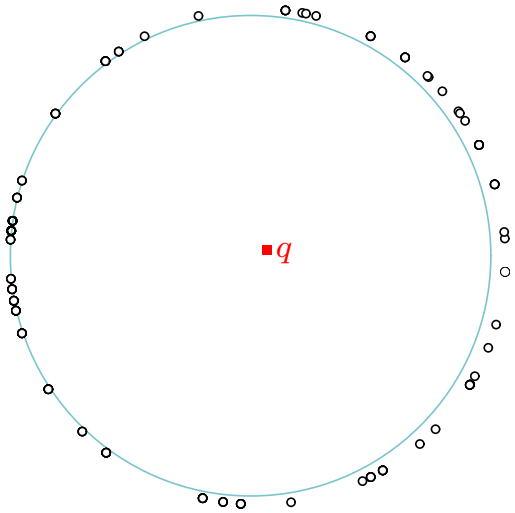
query input: q

goal: find $p \in \mathrm{NN}_P(q)$

Curse of Dimensionality:
Every data structure for NN-search has either exponential size or exponential query time (in d) in the worst case.

$\rightarrow$ **holds both in theory and in practice** [Weber et al. '98] [Arya et al. '98]

High dimensions



pre-processing input: $P \subset \mathbb{R}^d$

query input: q

goal: find $p \in \mathrm{NN}_P(q)$

Curse of Dimensionality:
Every data structure for NN-search has either exponential size or exponential query time (in d) in the worst case.

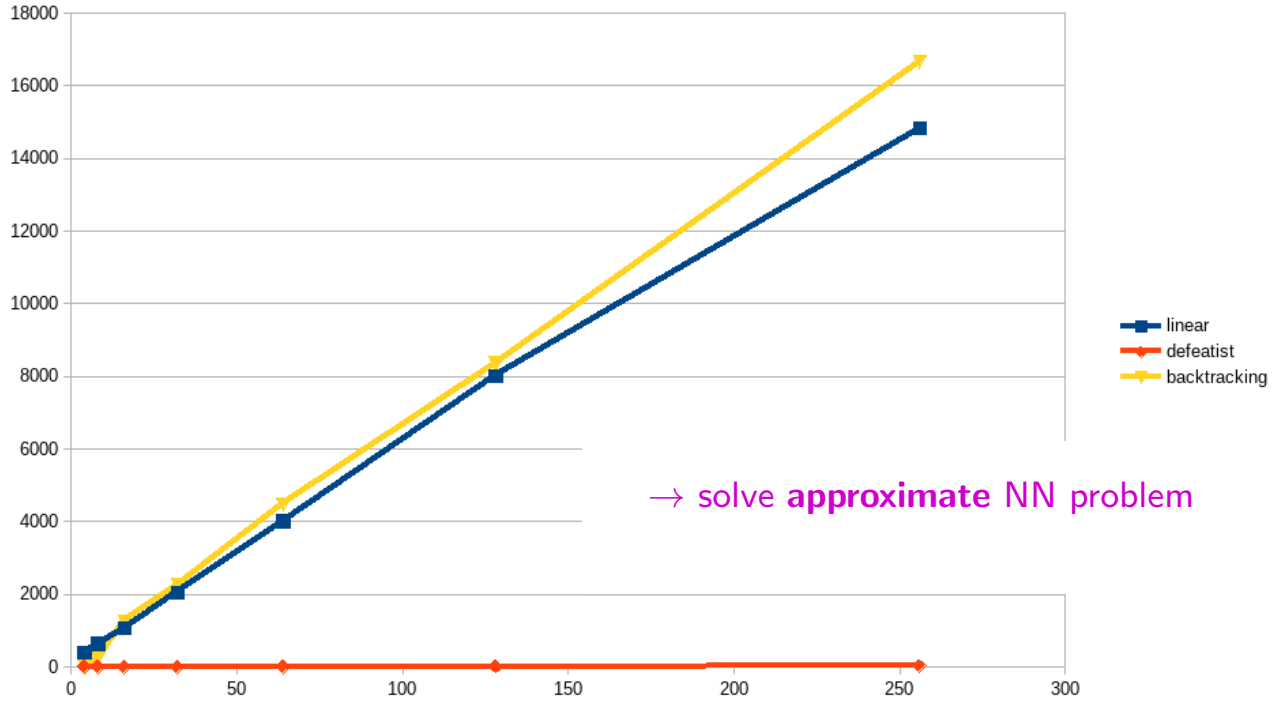
→ **holds both in theory and in practice** [Weber et al. '98] [Arya et al. '98]

→ underlying phenomenon: **concentration of measure**

(distances concentrate around mean) [Demartinez '94]

Benchmarks

avg. query time (μs) vs. dimension: (10,000 pts sampled uniformly inside unit cube)



→ solve **approximate** NN problem

What you should know

- Context and definition of NN search problem
- Linear scan
- An idea of existing approaches
- k-d trees: definition, construction, defeatist search, backtracking search
- Curse of dimensionality

Clustering with k-Means

Outline:

- **Clustering: problem statement and popular approaches**
- **k-means: minimizing intra-cluster variance**
- **Characterizing the argmin**
- **Computation: easy special cases and hard general case**
- **Lloyd's variational heuristic**
- **Initialization**
- **Choosing the number of clusters**

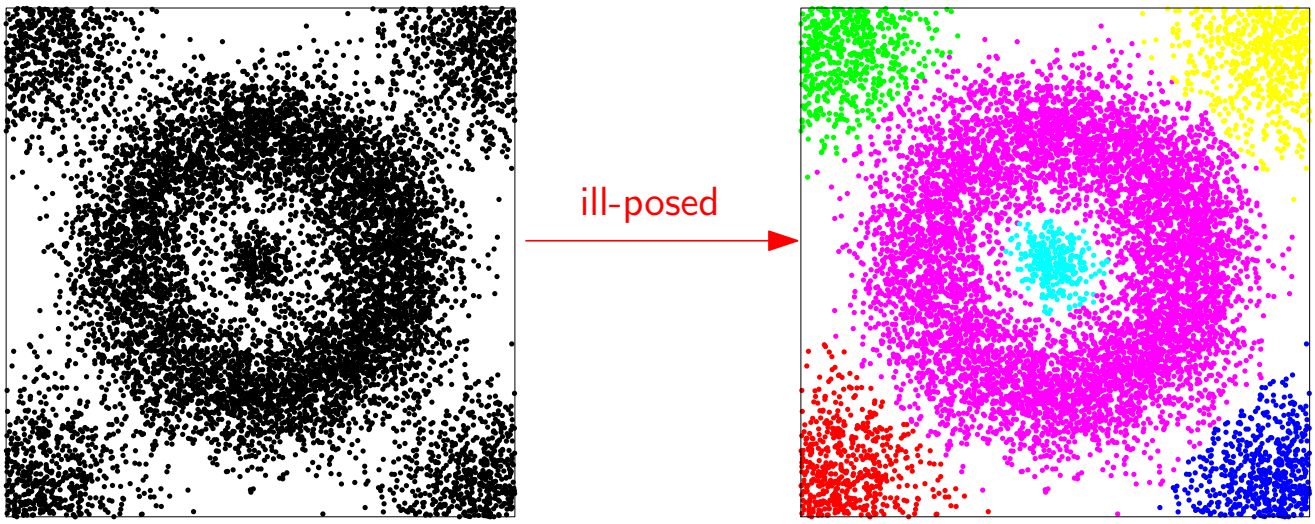
Outline

- **Clustering: problem statement and popular approaches**
- k-means: minimizing intra-cluster variance
- Characterizing the argmin
- Computation: easy special cases and hard general case
- Lloyd's variational heuristic
- Initialization
- Choosing the number of clusters

Clustering (a.k.a. unsupervised classification)

Input: a finite set of observations: - point cloud with coordinates

- distance / (dis-)similarity matrix



Task:

partition the data points into *homogeneous* subsets (clusters)

A wealth of approaches

Variational

- k-means
- k-medoids
- EM

Spectral

- Normalized Cut
- Multiway Cut

Hierarchical divisive/agglomerative

- single-linkage
- BIRCH

Density thresholding

- DBSCAN
- OPTICS

Mode seeking

- Mean/Medoid/Quick Shift
- graph-based hill climbing

Valley seeking

- [JBD'79]
- NDDs [ZZZL'07]

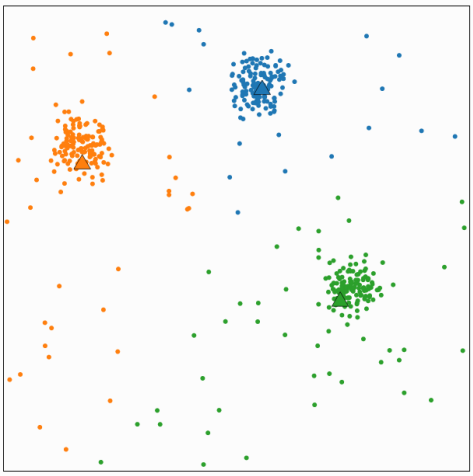
Outline

- Clustering: problem statement and popular approaches
- **k-means: minimizing intra-cluster variance**
- Characterizing the argmin
- Computation: easy special cases and hard general case
- Lloyd's variational heuristic
- Initialization
- Choosing the number of clusters

k-Means

Paradigm: cast clustering into an optimization problem

$\rightarrow$ minimize total intra-cluster variance



input: $P \subset \mathbb{R}^d$ finite $(\#P = n)$

hyper-parameter: k: number of clusters

parameters: $c_1, \cdots, c_k \in \mathbb{R}^d$: cluster centers

$\sigma : P \to \{1, \cdots, k\}$: partition

$$\min_{c_1,\cdots,c_k,\sigma} \frac{1}{n} \sum_{p \in P} \|p - c_{\sigma(p)}\|_2^2$$

(visualization tool: `http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html`)

k-Means

Paradigm: cast clustering into an optimization problem

$\rightarrow$ minimize total intra-cluster variance

$C_i := \sigma^{-1}(\{i\})$, $n_i := \#C_i$

$$\frac{1}{n} \sum_{p \in P} \|p - c_{\sigma(p)}\|_2^2$$

$$= \frac{1}{n} \sum_{i=1}^{k} \sum_{p \in C_i} \|p - c_i\|_2^2$$

$$= \sum_{i=1}^{k} \frac{n_i}{n} \operatorname{Var}(C_i, c_i)$$

(weighted sum of cluster variances)

input: $P \subset \mathbb{R}^d$ finite ($\#P = n$)

hyper-parameter: k: number of clusters

parameters: $c_1, \cdots, c_k \in \mathbb{R}^d$: cluster centers

$\sigma : P \rightarrow \{1, \cdots, k\}$: partition

$$\min_{c_1, \cdots, c_k, \sigma} \underbrace{\frac{1}{n} \sum_{p \in P} \|p - c_{\sigma(p)}\|_2^2}_{\operatorname{Var}(P, c_1, \cdots, c_k, \sigma)}$$

Outline

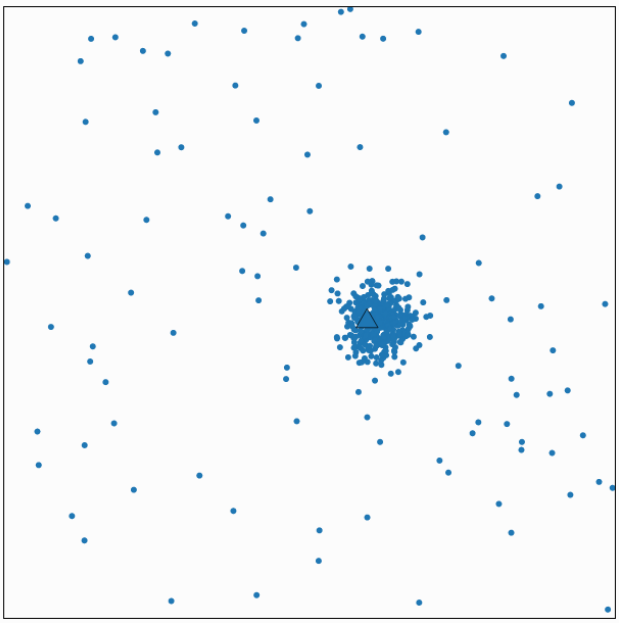
- Clustering: problem statement and popular approaches
- k-means: minimizing intra-cluster variance
- **Characterizing the argmin**
- Computation: easy special cases and hard general case
- Lloyd's variational heuristic
- Initialization
- Choosing the number of clusters

Characterizing the argmin

Fixed partition: $\sigma : P \to \{1, \cdots, k\}$

$\to$ optimize each center c_i independently, yields center of mass (centroid)

Prop: $\underbrace{\operatorname*{argmin}_{c_i} \operatorname{Var}(C_i, c_i)}_{\text{Fréchet mean}} = \underbrace{\frac{1}{n_i} \sum_{p \in C_i} p}_{\substack{\text{arithmetic} \\ \text{mean} \\ \text{(centroid)}}}$



Characterizing the argmin

Fixed partition: $\sigma : P \to \{1, \cdots, k\}$

$\to$ optimize each center c_i independently, yields center of mass (centroid)

Prop: $\operatorname*{argmin}_{c_i} \operatorname{Var}(C_i, c_i) = \dfrac{1}{n_i} \sum_{p \in C_i} p \quad =: c^*$

proof:

$$\begin{aligned}
\operatorname{Var}(C_i, c_i) = \frac{1}{n_i} \sum_{p \in C_i} \|p - c_i\|_2^2 \quad &= \frac{1}{n_i} \sum_{p \in C_i} \|(p - c^*) + (c^* - c_i)\|_2^2 \\
&= \frac{1}{n_i} \sum_{p \in C_i} \|p - c^*\|_2^2 + \frac{2}{n_i} \sum_{p \in C_i} (p - c^*) \cdot (c^* - c_i) + \frac{1}{n_i} \sum_{p \in C_i} \|c^* - c_i\|_2^2 \\
&= \operatorname{Var}(C_i, c^*) + \underbrace{\|c^* - c_i\|_2^2}_{\geq 0}
\end{aligned}$$

□

Characterizing the argmin

Fixed centers: $c_1, \cdots, c_k$

$\to$ optimize each assignment $\sigma(p)$ independently, yields Voronoi partition

Prop: $\operatorname*{argmin}_{\sigma} \operatorname{Var}(P, c_1, \cdots, c_k, \sigma) = \left(p \mapsto \mathrm{NN}_{\{c_1, \cdots, c_k\}}(p)\right) \quad =: \sigma_{\mathrm{NN}}$

proof:

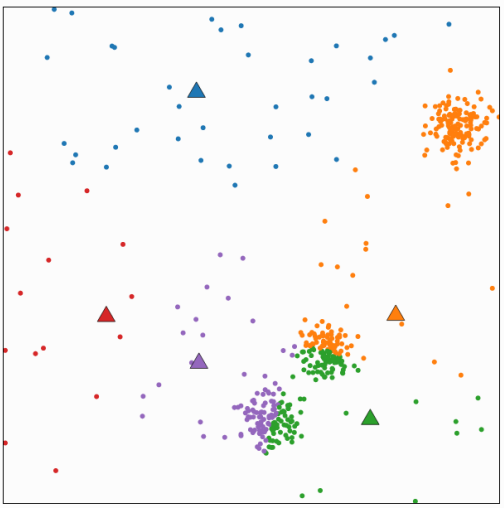
For each point $p \in P$:

$$\|p - c_{\sigma(p)}\|_2^2 \geq \|p - \mathrm{NN}_{\{c_1, \cdots, c_k\}}(p)\|_2^2$$

Hence, by termwise minimization:

$$\sum_{p \in P} \|p - c_{\sigma(p)}\|_2^2 \geq \sum_{p \in P} \|p - \mathrm{NN}_{\{c_1, \cdots, c_k\}}(p)\|_2^2$$

□



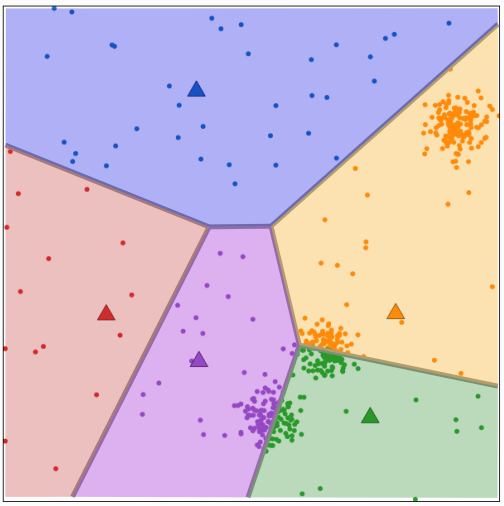
Characterizing the argmin

Fixed centers: $c_1, \cdots, c_k$

$\rightarrow$ optimize each assignment $\sigma(p)$ independently, yields Voronoi partition

Prop: $\operatorname*{argmin}_{\sigma} \operatorname{Var}(P, c_1, \cdots, c_k, \sigma) = \left(p \mapsto \operatorname{NN}_{\{c_1,\cdots,c_k\}}(p)\right) \quad =: \sigma_{\operatorname{NN}}$

$V(c_i) := \{x \in \mathbb{R}^d \mid \|x - c_i\|_2 \leq \|x - c_j\|_2 \ \forall j\}$



Prop: $V(c_i) \cap V(c_j)$ is affine

proof:

$$\begin{aligned} & \|x - c_i\|_2^2 = \|x - c_j\|_2^2 \\ \iff & \cancel{x^2} - 2\,x \cdot c_i + c_i^2 = \cancel{x^2} - 2\,x \cdot c_j + c_j^2 \\ \iff & 2\,x \cdot (c_j - c_i) + (c_i^2 - c_j^2) = 0 \end{aligned}$$

(affine equation) $\square$

Characterizing the argmin

Fixed centers: $c_1, \cdots, c_k$

$\rightarrow$ optimize each assignment $\sigma(p)$ independently, yields Voronoi partition

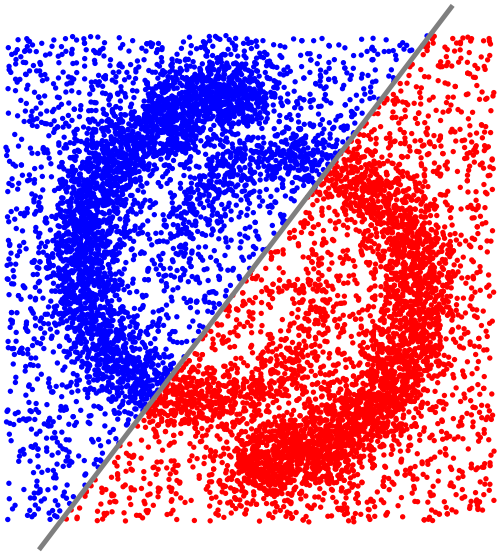
Prop: $\operatorname*{argmin}_{\sigma} \operatorname{Var}(P, c_1, \cdots, c_k, \sigma) = \left(p \mapsto \operatorname{NN}_{\{c_1,\cdots,c_k\}}(p)\right) \quad =: \sigma_{\operatorname{NN}}$

$V(c_i) := \{x \in \mathbb{R}^d \mid \|x - c_i\|_2 \leq \|x - c_j\|_2 \ \forall j\}$

Prop: $V(c_i) \cap V(c_j)$ is affine

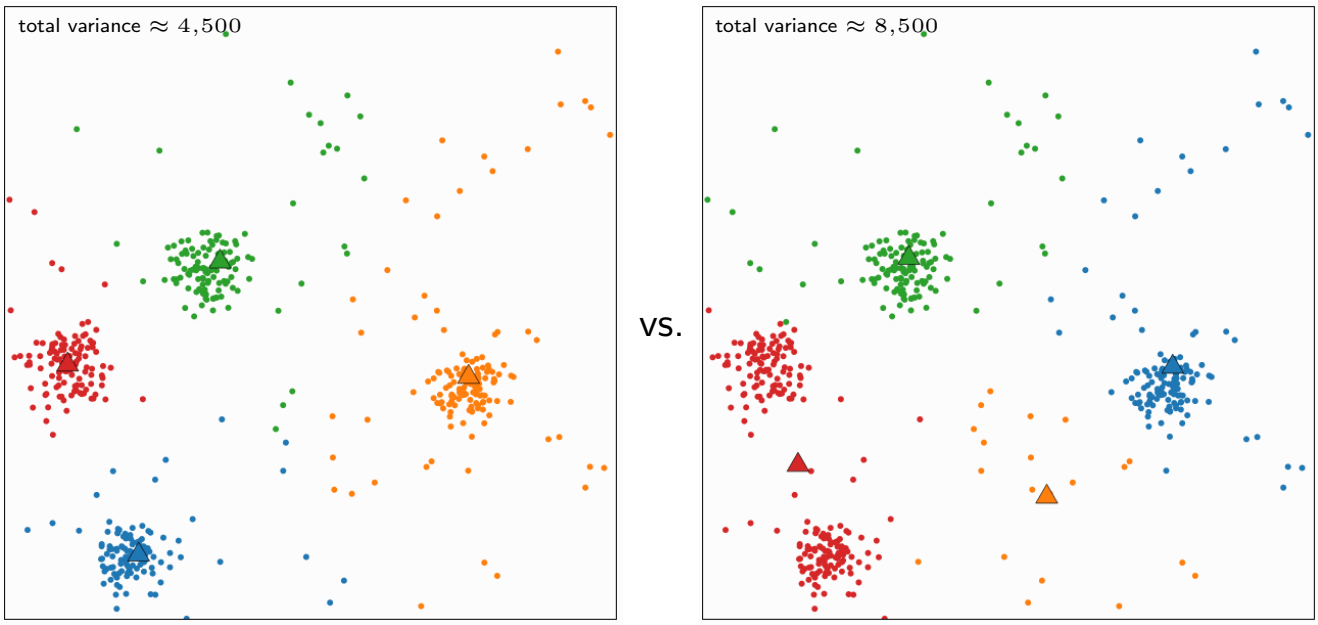
$$\Rightarrow V(c_i) = \bigcap_{j \neq i} \underbrace{\{x \in \mathbb{R}^d \mid \|x - c_i\|_2^2 \leq \|x - c_j\|_2^2\}}_{\text{half-space}}$$

$\Rightarrow$ clusters are convex



Characterizing the argmin

This characterization is **not unique:**



Both configurations are centroidal Voronoi partitions

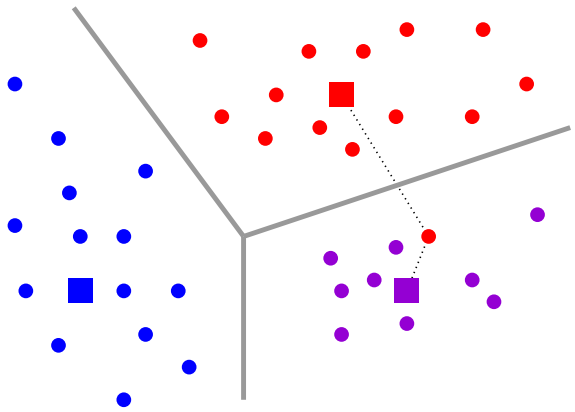
Characterizing the argmin

This characterization is **not unique:**

Prop: Every centroidal Voronoi partition $(P, c_1^*, \cdots, c_k^*, \sigma_{\text{NN}})$ such that there are no points of P on the boundaries corresponds to a **local minimum** of $\text{Var}(P, c_1, \cdots, c_k, \sigma)$.

proof:

- no pts on boundaries $\Rightarrow \text{Var}(P, c_1^*, \cdots, c_k^*, \sigma) > \text{Var}(P, c_1^*, \cdots, c_k^*, \sigma_{\text{NN}})\ \forall \sigma \neq \sigma_{\text{NN}}$



Characterizing the argmin

This characterization is **not unique:**

Prop: Every centroidal Voronoi partition $(P, c_1^*, \cdots, c_k^*, \sigma_{\mathrm{NN}})$ such that there are no points of P on the boundaries corresponds to a **local minimum** of $\mathrm{Var}(P, c_1, \cdots, c_k, \sigma)$.

proof:

- no pts on boundaries $\Rightarrow \mathrm{Var}(P, c_1^*, \cdots, c_k^*, \sigma) > \mathrm{Var}(P, c_1^*, \cdots, c_k^*, \sigma_{\mathrm{NN}})\ \forall \sigma \neq \sigma_{\mathrm{NN}}$
- for $\sigma \neq \sigma_{\mathrm{NN}}$ fixed, the map $(c_1, \cdots, c_k) \mapsto \mathrm{Var}(P, c_1, \cdots, c_k, \sigma)$ is continuous

  $\Rightarrow$ for $(c_1, \cdots, c_k)$ close enough to $(c_1^*, \cdots, c_k^*)$,

  $$\mathrm{Var}(P, c_1, \cdots, c_k, \sigma) > \mathrm{Var}(P, c_1^*, \cdots, c_k^*, \sigma_{\mathrm{NN}})$$

- $\exists$ finitely many partitions $\Rightarrow \exists$ common neighborhood for all $\sigma \neq \sigma_{\mathrm{NN}}$
- meanwhile, $(c_1^*, \cdots, c_k^*)$ are optimal for σ_{NN} itself

□

Characterizing the argmin

This characterization is **not unique:**

Prop: Every centroidal Voronoi partition $(P, c_1^*, \cdots, c_k^*, \sigma_{\mathrm{NN}})$ such that there are no points of P on the boundaries corresponds to a **local minimum** of $\mathrm{Var}(P, c_1, \cdots, c_k, \sigma)$.

$\rightarrow$ degenerate cases (points on boundaries) lead to non-minimal configurations:

Outline

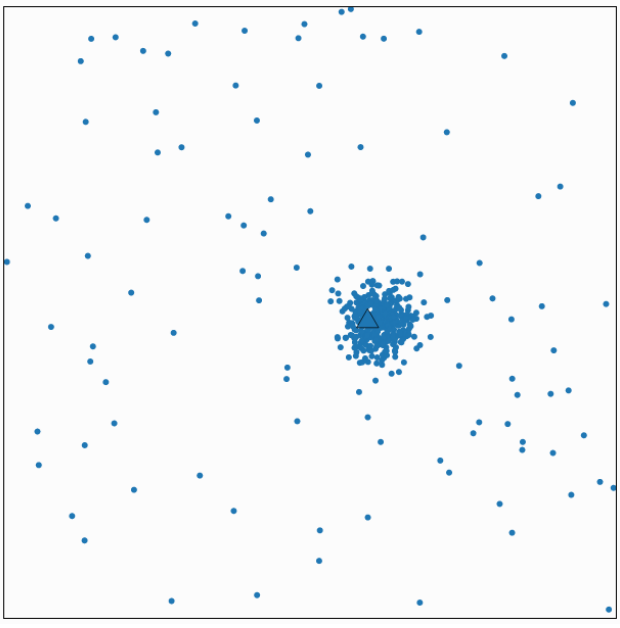
- Clustering: problem statement and popular approaches
- k-means: minimizing intra-cluster variance
- Characterizing the argmin
- **Computation: easy special cases and hard general case**
- Lloyd's variational heuristic
- Initialization
- Choosing the number of clusters

Computation: easy special cases

Case $k = 1$: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$

$\rightarrow$ put c_1 at the center of mass (arithmetic mean) of P

complexity: $O(n\,d)$



Computation: easy special cases

Case $d = 1$: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}$

$\rightarrow$ solve the problem in polynomial time by dynamic programming:

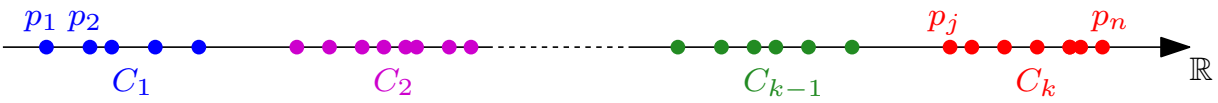
- sort and relabel the points of p so that $p_1 \leq p_2 \leq \cdots \leq p_n$

Voronoi cells are convex $\Rightarrow$ optimal clusters are contiguous: $C_1 < C_2 < \cdots < C_k$

Given $p_j := \min C_k$, clusters $C_1, \cdots, C_{k-1}$ are optimal for $\{p_1, \cdots, p_{j-1}\}$

- recurrence: $\mathrm{OPT}(n,k) := \min_{c_1,\cdots,c_k,\sigma} \sum_{i=1}^{n} |p_i - c_{\sigma(p_i)}|^2$

$$\begin{cases} \mathrm{OPT}(n,k) = \min\limits_{1 \leq j \leq n} \{\mathrm{OPT}(j-1,k-1) + \underbrace{(n+1-j)\,\mathrm{Var}(\{p_j, \cdots, p_n\})}_{\text{sum of squared distances to the mean}}\} \\ \mathrm{OPT}(n,k) = 0 \text{ if } n = 0 \text{ and } +\infty \text{ if } n \neq 0 = k \end{cases}$$



Computation: easy special cases

Case $d = 1$: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}$

$\rightarrow$ solve the problem in polynomial time by dynamic programming:

- sort and relabel the points of p so that $p_1 \leq p_2 \leq \cdots \leq p_n$

Voronoi cells are convex $\Rightarrow$ optimal clusters are contiguous: $C_1 < C_2 < \cdots < C_k$

Given $p_j := \min C_k$, clusters $C_1, \cdots, C_{k-1}$ are optimal for $\{p_1, \cdots, p_{j-1}\}$

- recurrence: $\mathrm{OPT}(n,k) := \min_{c_1,\cdots,c_k,\sigma} \sum_{i=1}^{n} |p_i - c_{\sigma(p_i)}|^2$

$$\begin{cases} \mathrm{OPT}(n,k) = \min\limits_{1 \leq j \leq n} \{\mathrm{OPT}(j-1,k-1) + \underbrace{(n+1-j)\,\mathrm{Var}(\{p_j, \cdots, p_n\})}_{\text{sum of squared distances to the mean}}\} \\ \mathrm{OPT}(n,k) = 0 \text{ if } n = 0 \text{ and } +\infty \text{ if } n \neq 0 = k \end{cases}$$

complexity: $O(n^3\,k)$ naive, or $O(n^2\,k)$ with linear-time aggregation of variances

Computation: hard general case

For arbitrary $k > 1$ and $d > 1$:

$\rightarrow$ naive: try all k^n partitions $\sigma : P \rightarrow \{1, \cdots, k\}$

complexity: $O\left(k^n \operatorname{poly}(n, k, d)\right)$

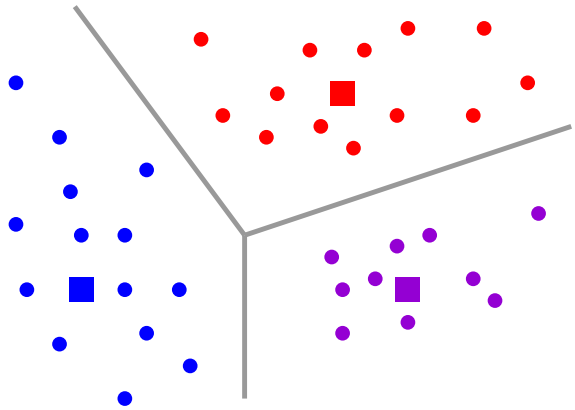
$\rightarrow$ better: try only the $O\left(n^{kd}\right)$ Voronoi partitions σ_{NN} [Inaba, Katoh, Imai 1994]

complexity: $O\left(n^{kd+1} \operatorname{poly}(n, k, d)\right)$

Problem is **NP-hard**:

- for arbitrary d, even when $k = 2$
  [Aloise et al. 2009]
- for arbitrary k, even when $d = 2$
  [Mahajan et al. 2009]

$\Rightarrow$ **no poly-time algorithm** in (n, k, d)
(unless $P = NP$)



Outline

- Clustering: problem statement and popular approaches
- k-means: minimizing intra-cluster variance
- Characterizing the argmin
- Computation: easy special cases and hard general case
- **Lloyd's variational heuristic**
- Initialization
- Choosing the number of clusters

Lloyd's variational heuristic

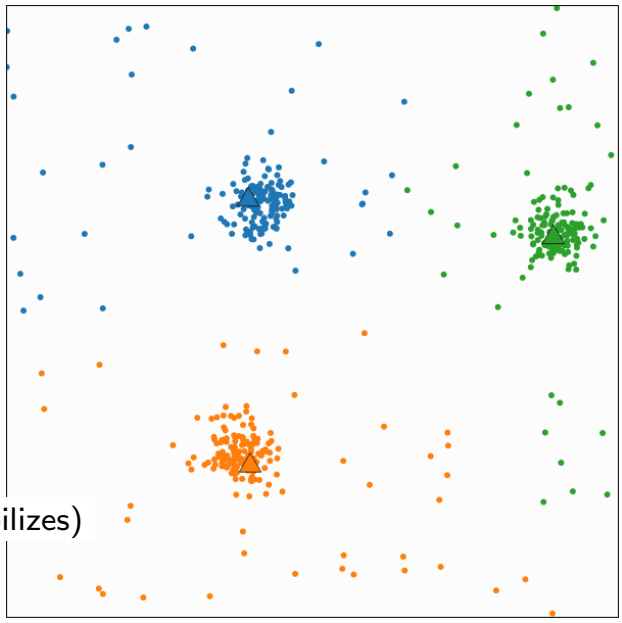
Input: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d, \quad k \geq 1$

Initialize cluster centers $c_1, \cdots, c_k \in \mathbb{R}^d$

Repeat:

- compute/update the Voronoi partition $\sigma_{\text{NN}} : P \to \{1, \cdots, k\}$ and let $\sigma = \sigma_{\text{NN}}$
- move each center c_i to the centroid of its cluster $C_i = \sigma^{-1}(\{i\})$

Until convergence ($\text{Var}(P, c_1, \cdots, c_k, \sigma)$ stabilizes)



Lloyd's variational heuristic

Input: $P = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d, \quad k \geq 1$

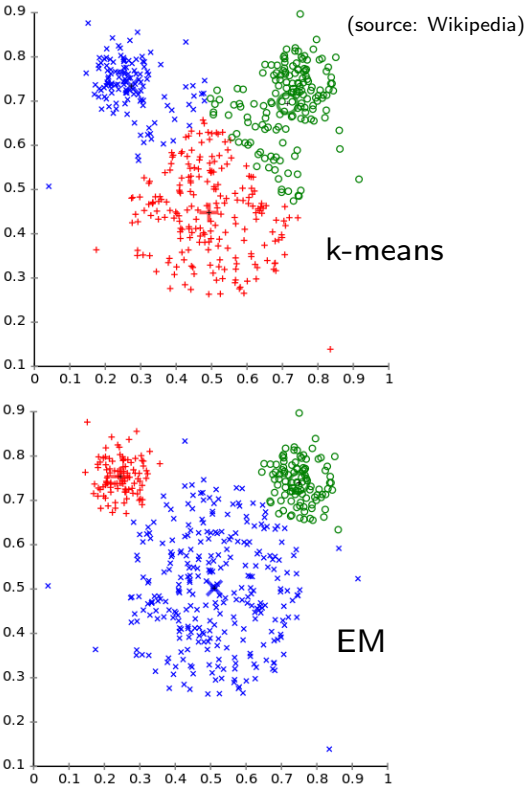
Initialize cluster centers $c_1, \cdots, c_k \in \mathbb{R}^d$

Repeat:

E-step
- compute/update the Voronoi partition $\sigma_{\mathrm{NN}} : P \to \{1, \cdots, k\}$ and let $\sigma = \sigma_{\mathrm{NN}}$

M-step
- move each center c_i to the centroid of its cluster $C_i = \sigma^{-1}(\{i\})$

Until convergence ($\mathrm{Var}(P, c_1, \cdots, c_k, \sigma)$ stabilizes)



→ special instance of the EM algorithm

EM is more general: non-uniform weights, anisotropic gaussians, soft clustering...

Lloyd's variational heuristic

Prop: The algorithm converges after finitely many iterations.

proof:

Each E-step decreases the total variance (Voronoi partition)

Each M-step decreases the variance (centroid)

$\Rightarrow$ total variance decreases **strictly** at each non-terminal iteration (stopping criterion)

After each iteration, the lowest variance is achieved for the current Voronoi partition σ

$\overset{\text{strict decrease}}{\Rightarrow}$ each Voronoi partition is considered during ≤ 1 non-terminal iteration

There are finitely many different Voronoi partitions

$\Rightarrow$ finitely many iterations in total

□

Lloyd's variational heuristic

Prop: The algorithm converges after finitely many iterations.

proof:

Each E-step decreases the total variance (Voronoi partition)

Each M-step decreases the variance (centroid)

$\Rightarrow$ total variance decreases **strictly** at each non-terminal iteration (stopping criterion)

After each iteration, the lowest variance is achieved for the current Voronoi partition σ

$\overset{\text{strict decrease}}{\Rightarrow}$ each Voronoi partition is considered during ≤ 1 non-terminal iteration

There are finitely many different Voronoi partitions ⟵ at most $O\left(n^{kd}\right)$ iterations

$\Rightarrow$ finitely many iterations in total

Complexity: $O(n^{kd}\text{poly}(n,k,d))$

□

Lloyd's variational heuristic

Prop: The algorithm converges after finitely many iterations.

Prop: The algorithm converges *generically* to a local minimum of the total variance.

Shortcomings:

- convergence may require an exponential number of iterations,
  even in the plane ($d = 2$) [Vattani 2011]
- depending on the initialization, the local minimum reached can be arbitrarily bad
  [MacQueen 1967]

(lots of room for improvement)

Note: certified $(1+\varepsilon)$-approx. algos. typically run in time $O(2^k\,\varepsilon^{-d}\,n\,\text{polylog}(n,k,d))$
[Har-Peled, Mazumdar 2004]

Outline

- Clustering: problem statement and popular approaches
- k-means: minimizing intra-cluster variance
- Characterizing the argmin
- Computation: easy special cases and hard general case
- Lloyd's variational heuristic
- **Initialization**
- Choosing the number of clusters

Initialization

Random centers:

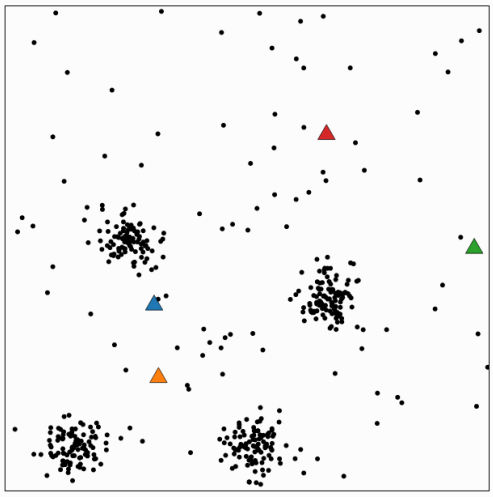
- sampled uniformly in bounding box
- sampled uniformly among the data points [Forgy 1965]
- defined as centroids of a random partition $\sigma : P \to \{1, \cdots, k\}$

(assign each data point randomly and independently to $i \in [\![1, k]\!]$)

$\to$ performances highly depend on input data

$\to$ they degrade quickly as k or d increases

$\sim 35\%$ success rate w/ unif. in bound. box on this input



Initialization

k-means++ [Arthur, Vassilvitskii 2007]

$\rightarrow$ soft (randomized) variant of furthest-point sampling

draw c_1 uniformly at random from P

$C := \{c_1\}$

for $i = 2$ to k **do**:

draw c_i at random from P according to the probability distribution:

$$\mathbb{P}(p) := \frac{\mathrm{d}(p, C)^2}{\sum_{q \in P} \mathrm{d}(q, C)^2}, \quad \text{where } \mathrm{d}(x, C) = \min_{1 \leq j < i} \|x - c_j\|_2$$

$C := C \cup \{c_i\}$

done

Initialization

k-means++ [Arthur, Vassilvitskii 2007]

$\rightarrow$ soft (randomized) variant of furthest-point sampling

$\rightarrow$ easy to implement in arbitrary dimensions

$\rightarrow$ theoretical guarantees on the resulting (initial) total variance

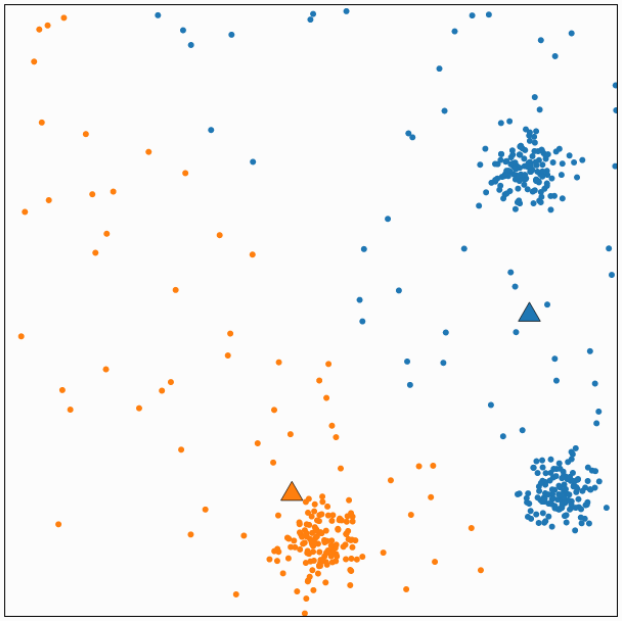
Prop: In expectation, the total variance is within a factor $O(\log k)$ of the optimal:

$$\mathbb{E}\left[\mathrm{Var}(P, c_1, \cdots, c_k, \sigma_{\mathrm{NN}})\right] \leq 8\ (2 + \ln k) \min_{c'_1, \cdots, c'_k, \sigma} \mathrm{Var}(P, c'_1, \cdots, c'_k, \sigma))$$

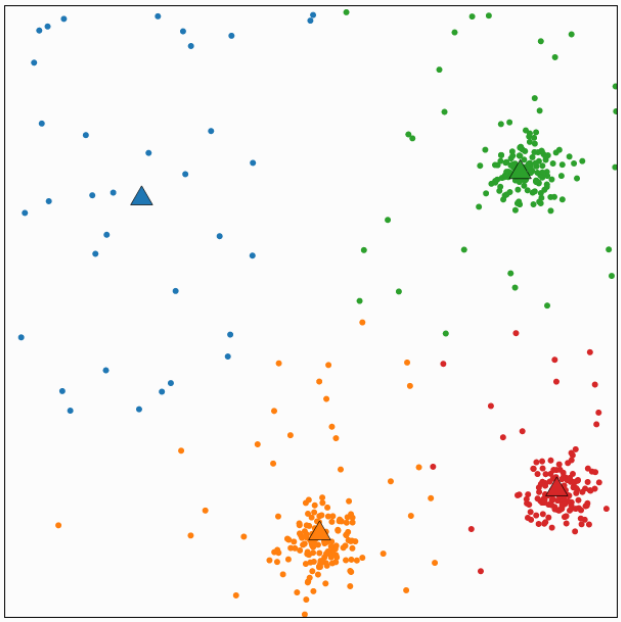
Outline

- Clustering: problem statement and popular approaches
- k-means: minimizing intra-cluster variance
- Characterizing the argmin
- Computation: easy special cases and hard general case
- Lloyd's variational heuristic
- Initialization
- **Choosing the number of clusters**

Choosing the number k of clusters



$k = 2$ (underfitting)

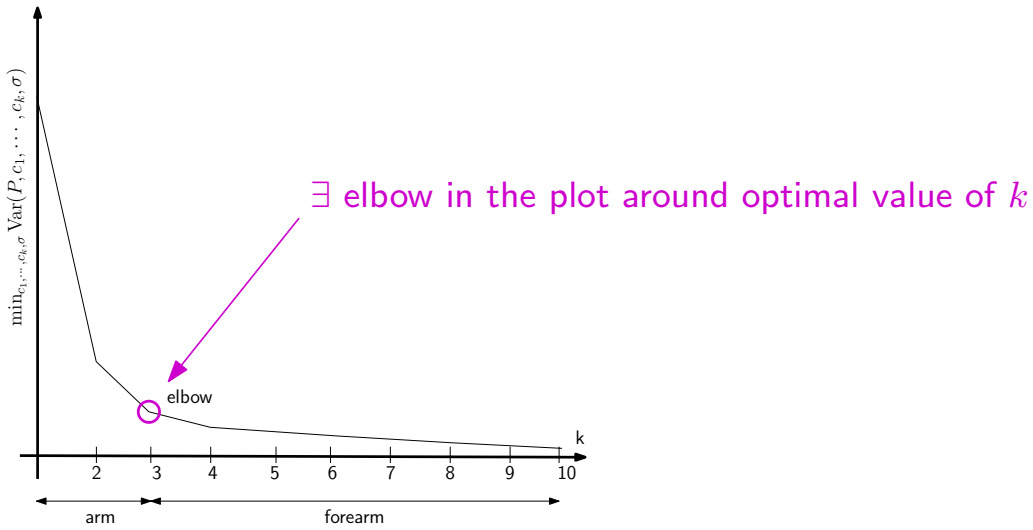


$k = 4$ (overfitting)

Choosing the number k of clusters

Elbow method (heuristic): [Thorndike 1953]

plot $\min_{c_1,\cdots,c_k,\sigma} \mathrm{Var}(P, c_1, \cdots, c_k, \sigma)$ as a function of k



Choosing the number k of clusters

Silhouette: [Rousseeuw 1987]

plot average silhouette $\frac{1}{n} \sum_{p \in P} s(p)$ as a function of k, where:

$$\text{silhouette } s(p) := \frac{b(p) - a(p)}{\max\{a(p),\, b(p)\}} \in [-1, 1]$$

$a(p) :=$ average distance of p to points in its cluster C

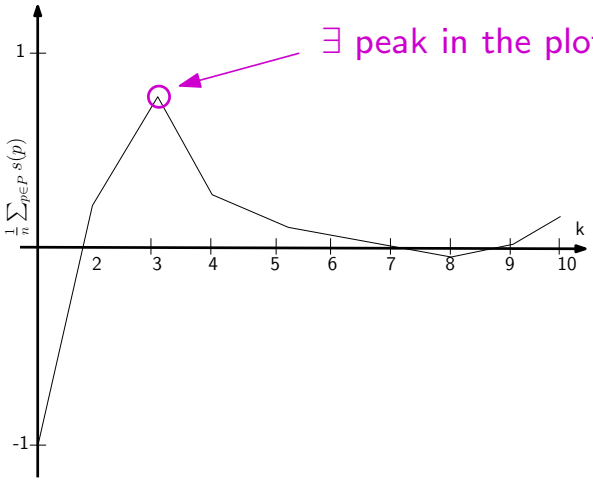
$b(p) := \min_{C' \neq C}$ average distance of p to points in cluster C'

Choosing the number k of clusters

Silhouette: [Rousseeuw 1987]

plot average silhouette $\frac{1}{n}\sum_{p\in P} s(p)$ as a function of k, where:

$$\text{silhouette } s(p) := \frac{b(p)-a(p)}{\max\{a(p),\, b(p)\}} \in [-1,1]$$



What you should know

- Context and definition of clustering
- k-means as a minimization problem (intra-cluster variance)
- argmins as centroidal Voronoi partitions
- Efficient computation in cases $k = 1$ or $d = 1$
- Lloyd's algorithm and its properties
- Initialization techniques: random, k-means++
- Number of centers selection: elbow method, silhouette

Hierarchical Clustering

Outline:

- **Principles of hierarchical clustering**
- **Agglomerative hierarchical clustering**
- **Single-linkage clustering**
- **Connection to ultrametrics and stability**
- **Hierarchical clustering and phylogenetic trees**

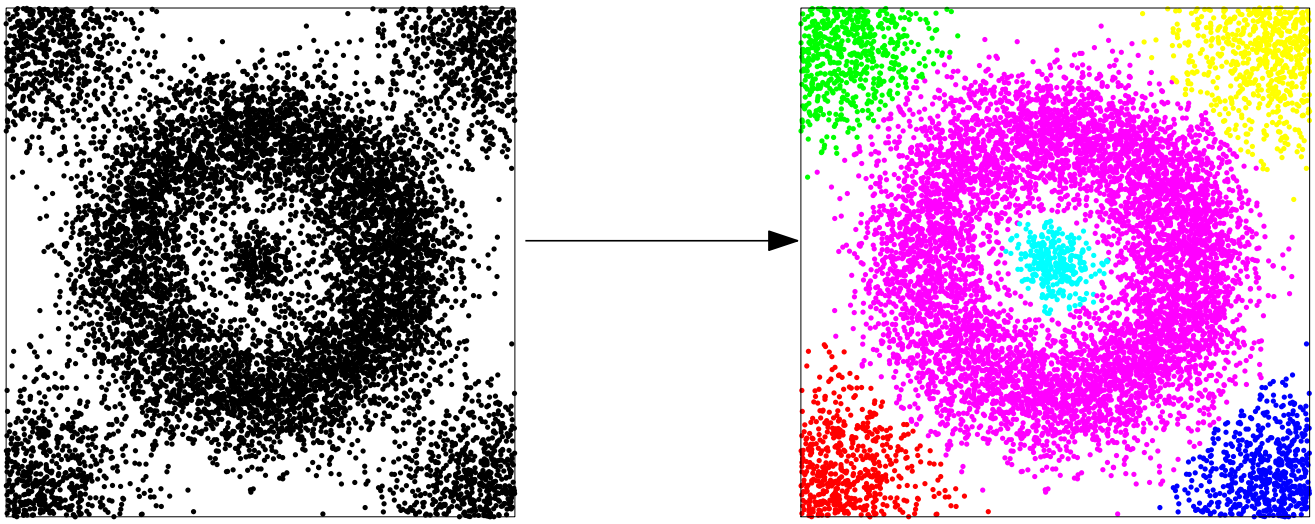
Outline

- **Principles of hierarchical clustering**
- Agglomerative hierarchical clustering
- Single-linkage clustering
- Connection to ultrametrics and stability
- Hierarchical clustering and phylogenetic trees

Hierarchical clustering

Input: a finite set of n observations: - point cloud with coordinates

- distance / (dis-)similarity matrix

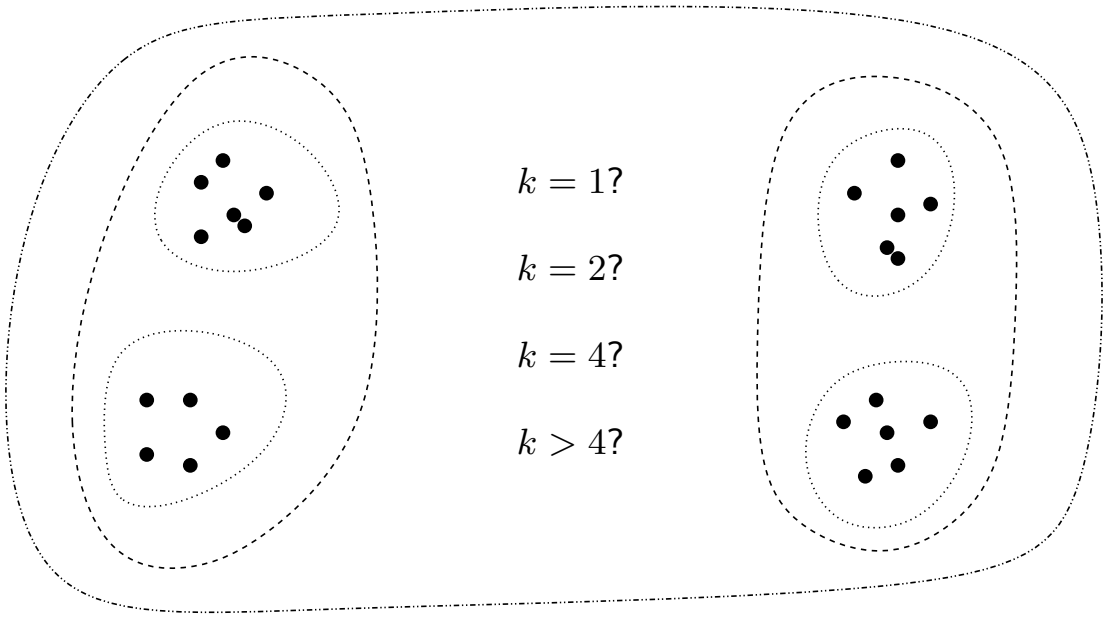


Task:

partition the data points into k *homogeneous* subsets (clusters)

Hierarchical clustering

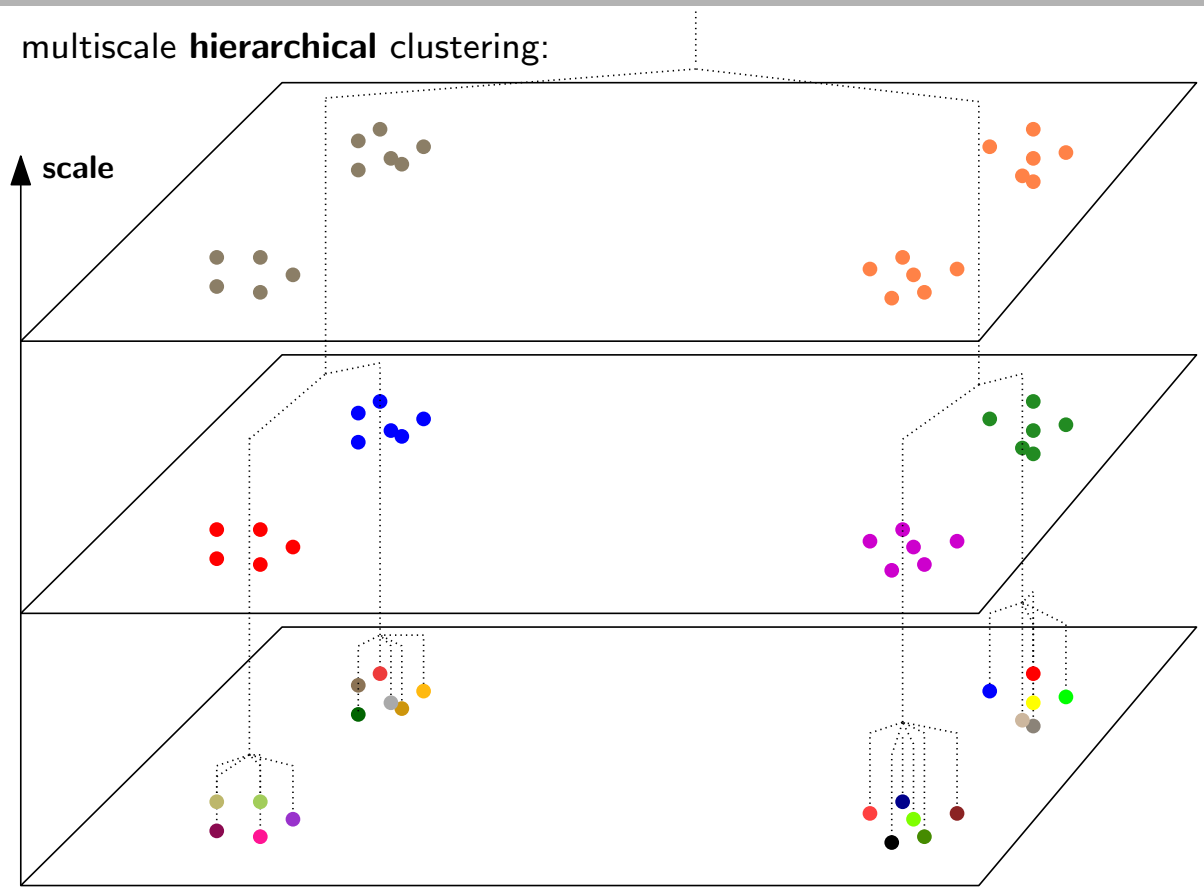
Q: what is the number k of clusters? what if there is more than one solution?



$\rightarrow$ solution depends on **scale**

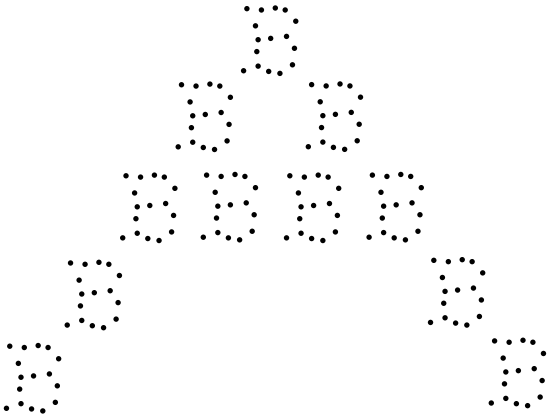
Hierarchical clustering

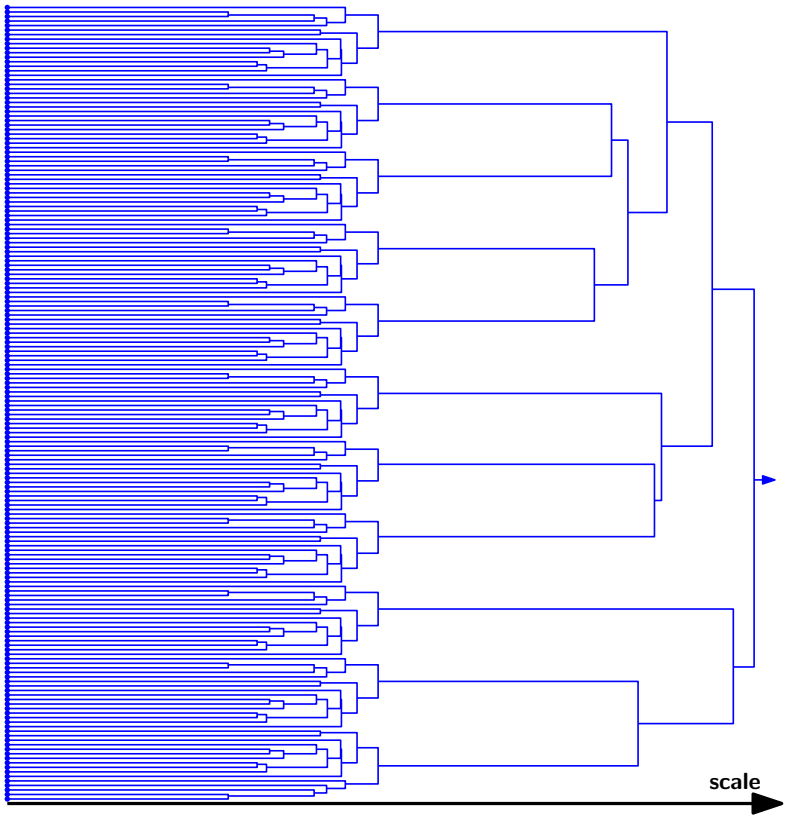
multiscale **hierarchical** clustering:



Hierarchical clustering

multiscale **hierarchical** clustering:





Hierarchical clustering

multiscale **hierarchical** clustering:

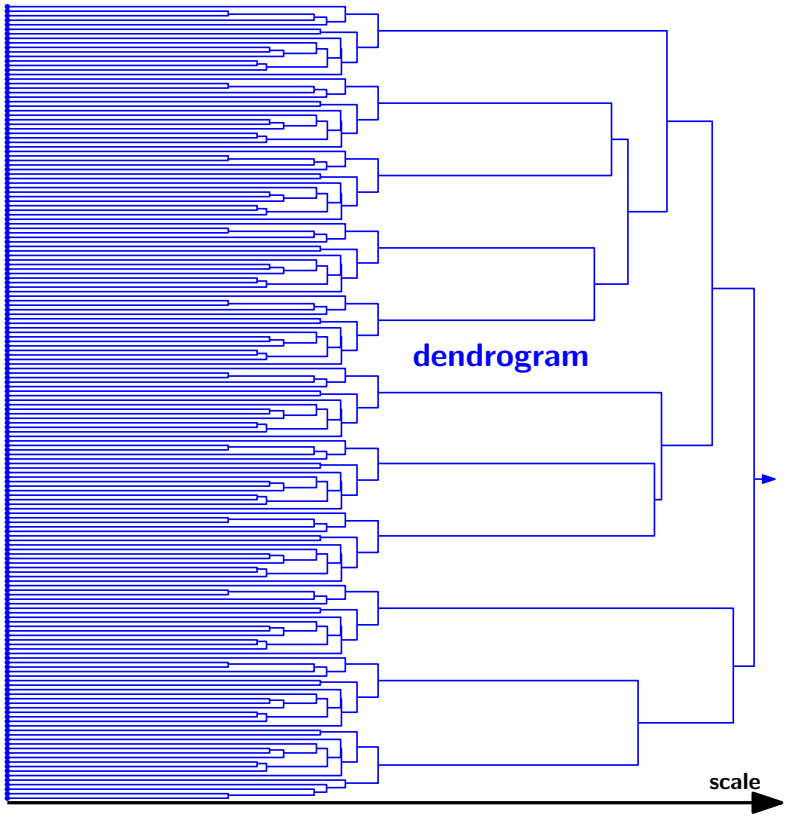
Def: (dendrogram) Given P finite, a dendrogram on P is a map

$$\theta : \mathbb{R}^+ \to \text{Partitions}(P)$$

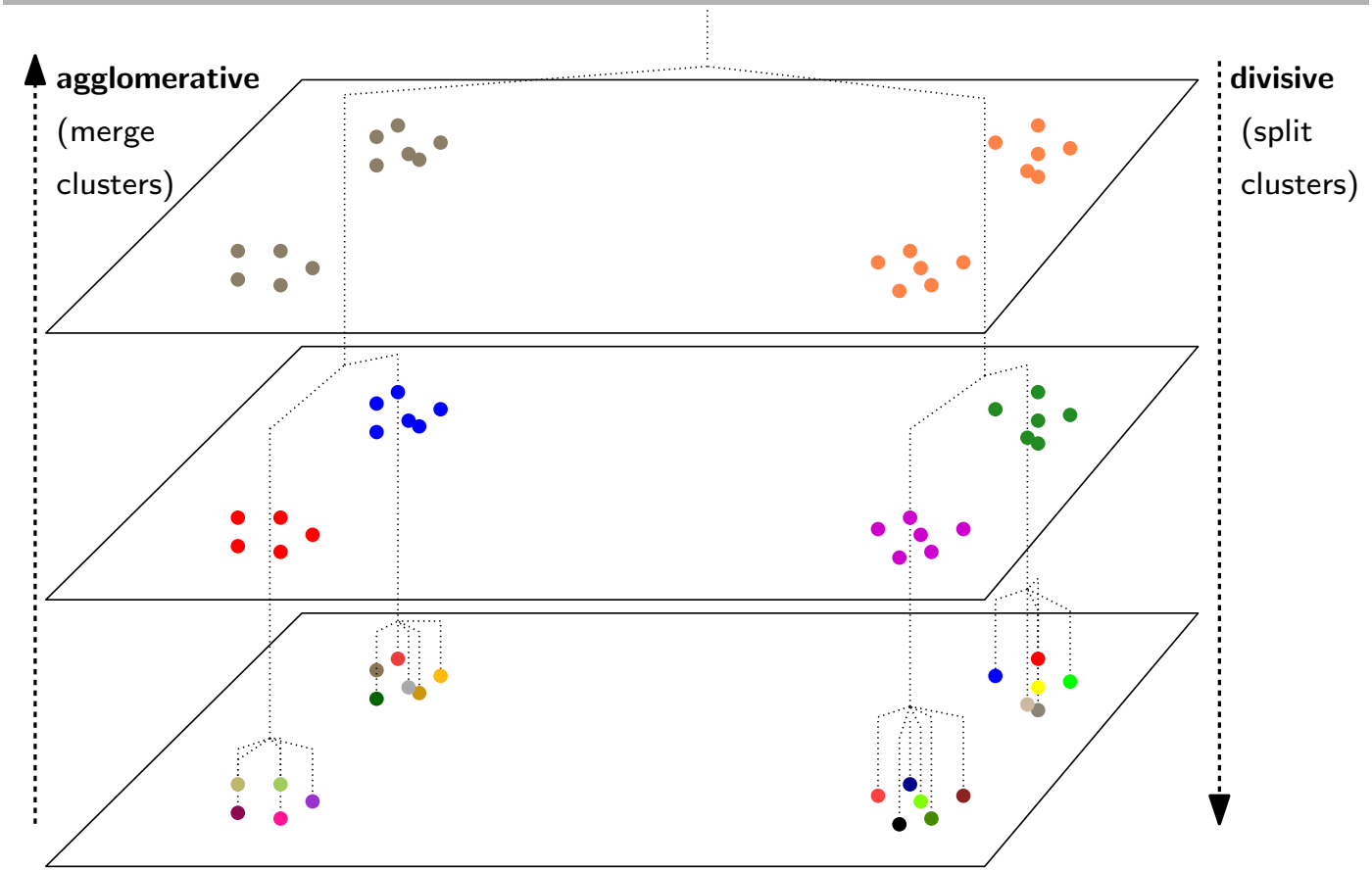
such that:

- $\theta(0) = \{\text{singletons}(P)\}$
- $\exists t_0 : \forall t \geq t_0,\ \theta(t) = \{P\}$
- $\forall t \leq t'$, $\theta(t)$ refines $\theta(t')$:

$$\forall C \in \theta(t),\ \exists C' \in \theta(t') : C \subseteq C'$$



Building the hierarchy



Building the hierarchy

Agglomerative hierarchical clustering (AHC):

- build tree from leaves to root
- start with each observation in its own cluster (leaf)
- iteratively merge clusters until only one remains (root)

(cf. Kruskal's minimum spanning tree algortihm)

Divisive hierarchical clustering (DHC):

- build tree from root to leaves
- start with all observations in the same cluster (root)
- recursively split the clusters until they become singletons (leaves)

(cf. kd-tree construction)

Building the hierarchy

Combinatorial aspects:

- $n-1$ steps for each approach (with two-fold merges or splits)
- at step k:

AHC: $n-k$ clusters and $\binom{n-k}{2}$ choices for merge

$\Rightarrow$ average size of search space: $\Theta(n^2)$

the most commonly used approach

DHC: k clusters of sizes $n_1, \cdots, n_k$, and $\sum_{i=1}^{k} \left(2^{n_i-1}-1\right)$ choices for split

$\Rightarrow$ average size of search space: $\Theta(2^n)$

the historical approach

Outline

- Principles of hierarchical clustering
- **Agglomerative hierarchical clustering**
- Single-linkage clustering
- Connection to ultrametrics and stability
- Hierarchical clustering and phylogenetic trees

Agglomerative hierarchical clustering

Meta-algorithm: input: (P, d) where $P = \{p_1, \cdots, p_n\}$

start with each data point $p_i \in P$ as its own cluster $\{p_i\}$ at height 0

for $k = 1$ to $n - 1$ **do**: // invariant: $\exists\ n - k$ clusters at end of iteration

choose the next two clusters C, C' to be merged ($C, C' \subset \{1, \cdots, n\}$)

merge C and C' by replacing them with a single cluster $C \cup C'$

record the new height and assign it to the merge of C, C'

done

→ use a union-find data structure (e.g. disjoint-set forest), as in Kruskal's algorithm

Agglomerative hierarchical clustering

Meta-algorithm: input: (P, d) where $P = \{p_1, \cdots, p_n\}$

start with each data point $p_i \in P$ as its own cluster $\{p_i\}$ at height 0

for $k = 1$ to $n - 1$ **do**: // invariant: $\exists\ n - k$ clusters at end of iteration

choose the next two clusters C, C' to be merged ($C, C' \subset \{1, \cdots, n\}$)

merge C and C' by replacing them with a single cluster $C \cup C'$

record the new height and assign it to the merge of C, C'

done

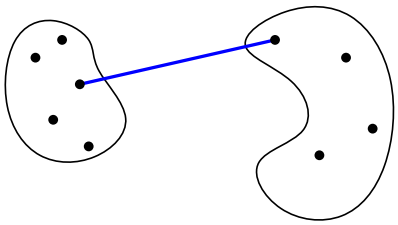
$\rightarrow$ define a distance δ between clusters and:
- merge the **closest** clusters
- their **distance** is the new height

Agglomerative hierarchical clustering

Distances δ based on ground metric d:

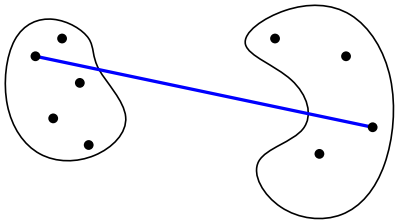
- minimum distance (**single-linkage**):

$$\delta_{\mathrm{SL}}(C, C') := \min_{p \in C, p' \in C'} \mathrm{d}(p, p')$$



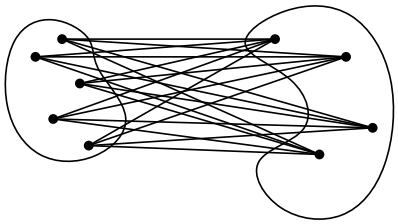
- maximum distance (**complete-linkage**):

$$\delta_{\mathrm{CL}}(C, C') := \max_{p \in C, p' \in C'} \mathrm{d}(p, p')$$



- mean distance (**average-linkage**):

$$\delta_{\mathrm{AL}}(C, C') := \frac{1}{|C|\,|C'|} \sum_{p \in C} \sum_{p' \in C'} \mathrm{d}(p, p')$$



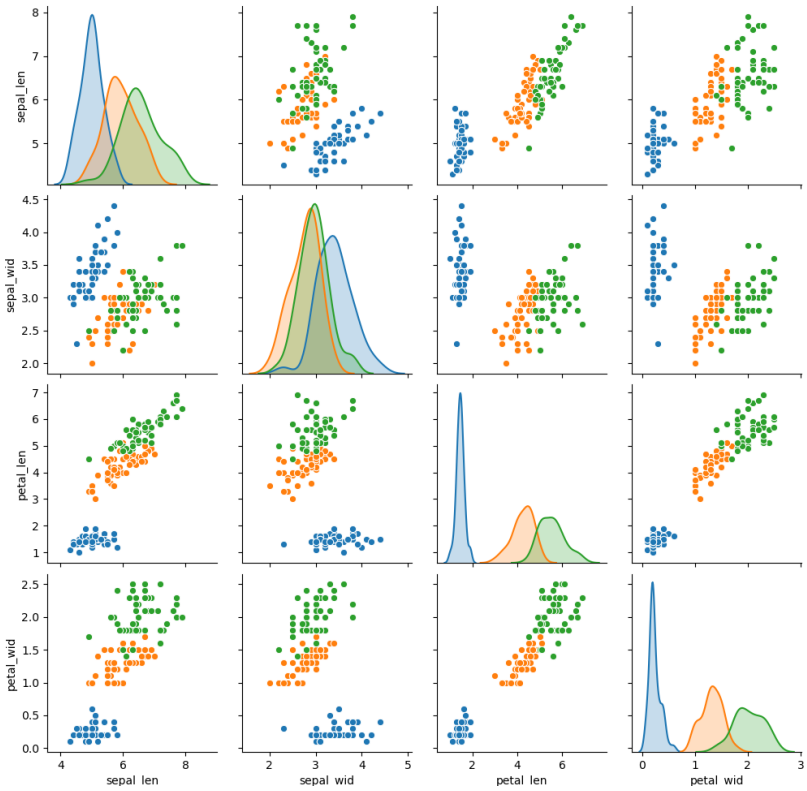
Impact of choice of distance δ

Real dataset (Fisher's *Iris* data):

- $n = 150$ observations
- $d = 4$ variables:

  sepal length/width,
  petal length/width
- $k = 3$ species:

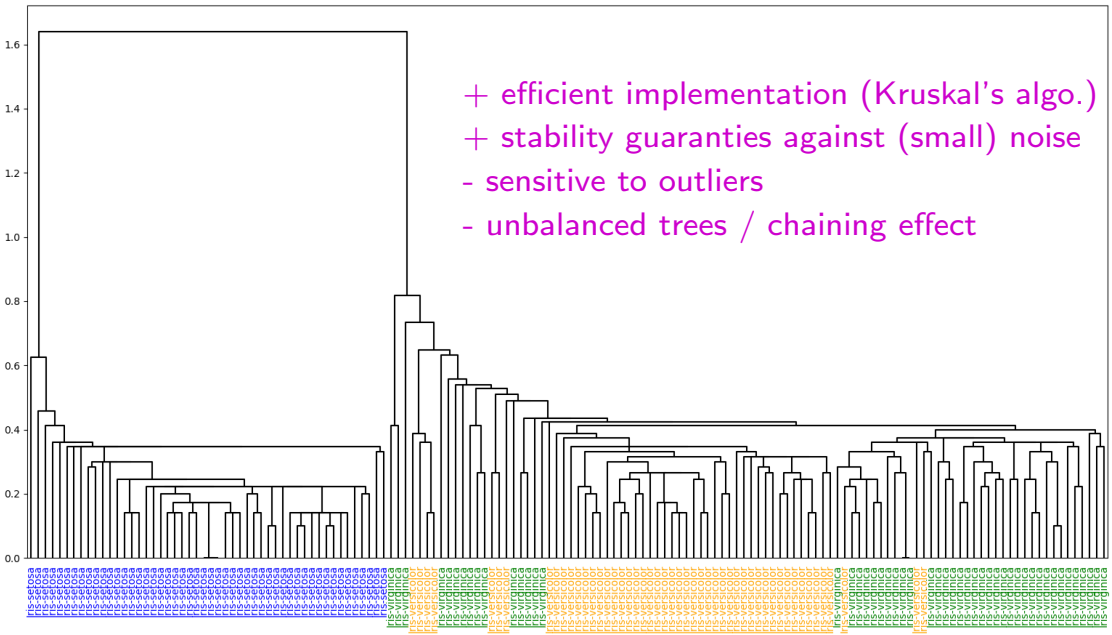
  virginica, versicolor, setosa



`http://archive.ics.uci.edu/ml/datasets/Iris`

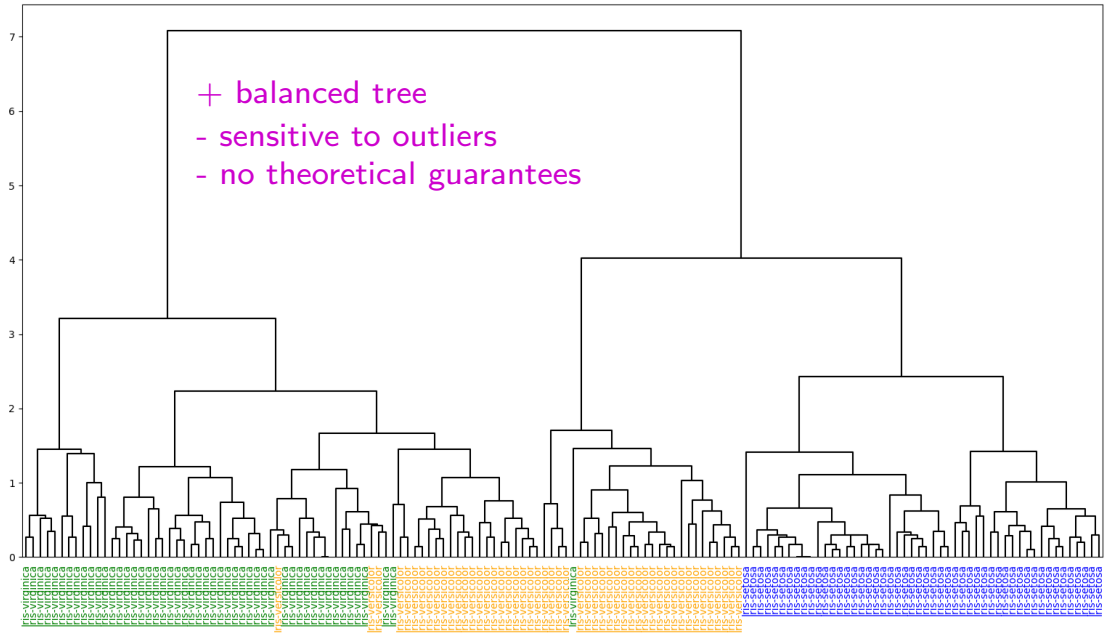
Impact of choice of distance δ

- single-linkage:



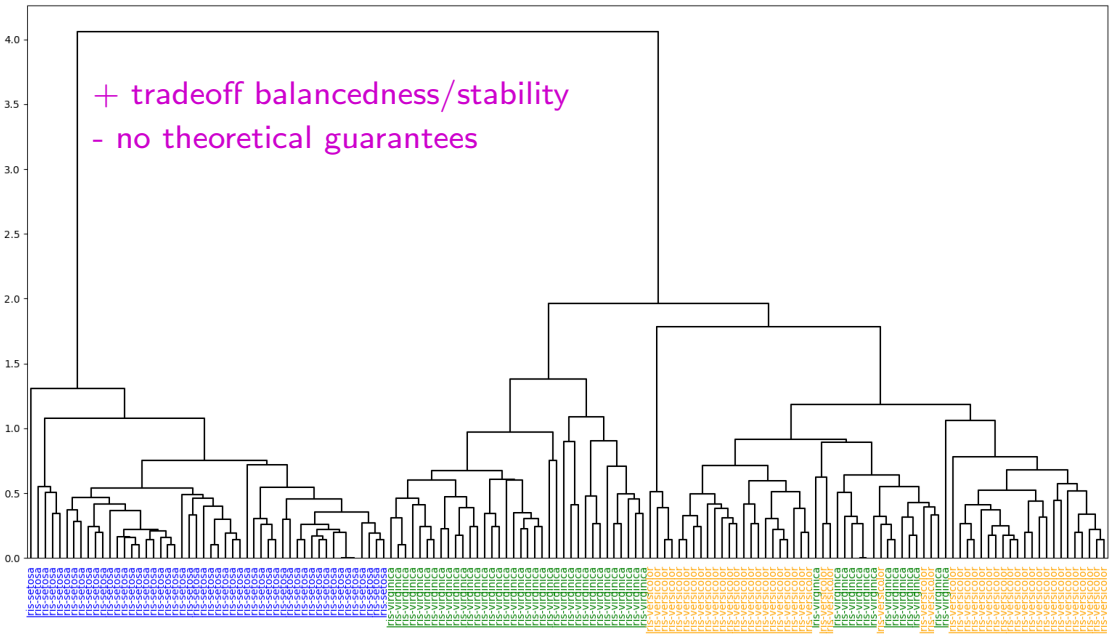
Impact of choice of distance δ

- complete-linkage:



Impact of choice of distance δ

- average-linkage:



Distances based on statistical quantities

Distance δ based on (weighted) intra-cluster variance — **Ward's criterion**:

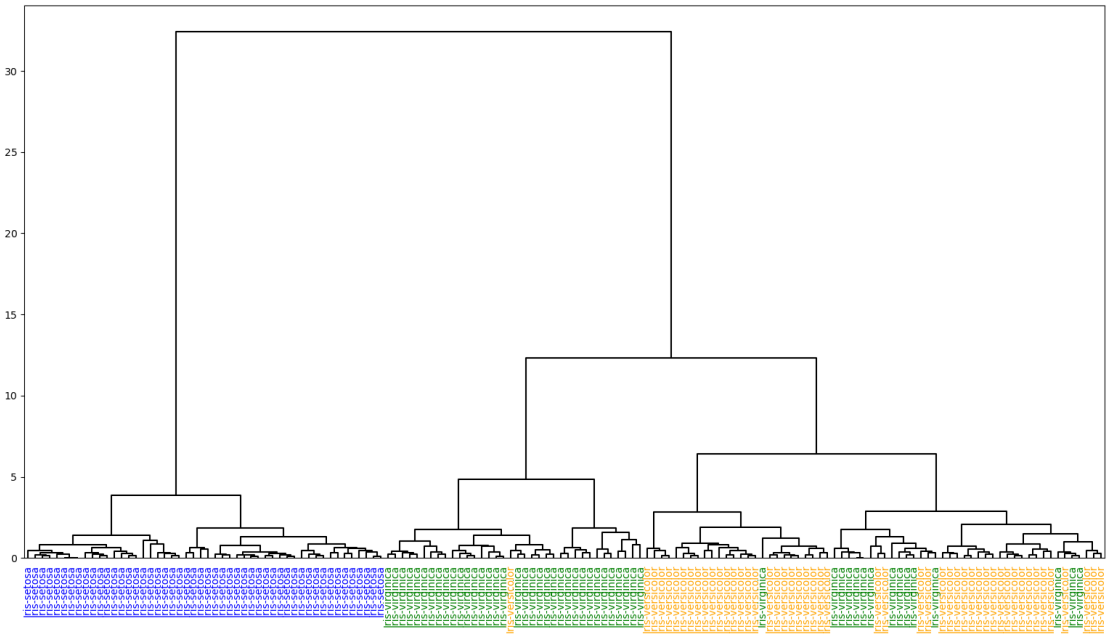
$$\delta(C, C') := |C \cup C'|\mathrm{Var}(C \cup C') - (|C|\mathrm{Var}(C) + |C'|\mathrm{Var}(C'))$$

$$= \frac{|C|\,|C'|}{|C|+|C'|}\,\|\mathbb{E}C - \mathbb{E}C'\|_2^2 \text{ in Euclidean space}$$

- next merge is the one that **least increases the** (weighted) **intra-cluster variance**
- $\delta(C, C')$ is **non-negative** and **monotonously increasing**
- greedy approach $\Rightarrow$ intra-cluster variance is **not minimal for a fixed #clusters**
  - ↳ output clusters can be optimized using k-means (same objective function)

Distances based on statistical quantities

- Ward's criterion on iris data:



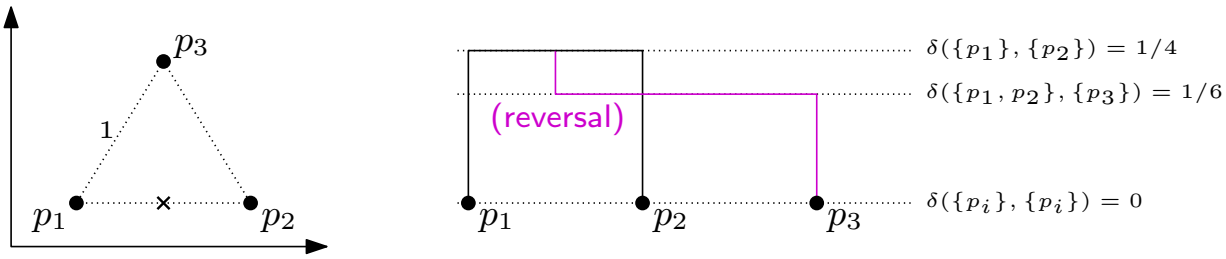
Distances based on statistical quantities

Distance δ based on ~~(weighted)~~ intra-cluster variance — ~~**Ward's criterion**~~:

$$\delta(C, C') := \cancel{|C \cup C'|}\mathrm{Var}(C \cup C') - (|C|\mathrm{Var}(C) + |C'|\mathrm{Var}(C')) \,/\, |C \cup C'|$$

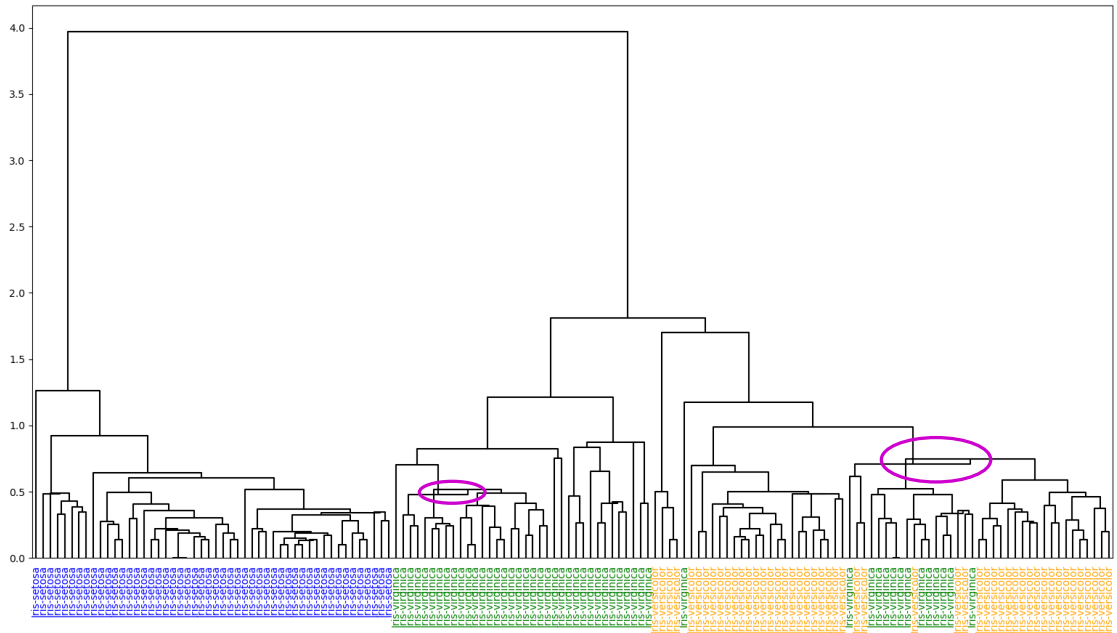
$$= \frac{|C|\,|C'|}{(|C|+|C'|)^2}\|\mathbb{E}C - \mathbb{E}C'\|_2^2 \text{ in Euclidean space}$$

- using unweighted variance may lead to **reversals** in the output dendrogram



Distances based on statistical quantities

- unweighted intra-cluster variance criterion on iris data:



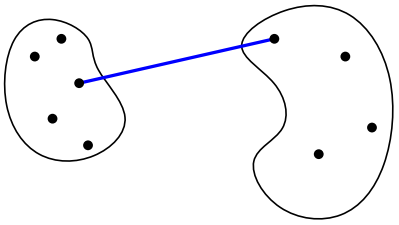
Outline

- Principles of hierarchical clustering
- Agglomerative hierarchical clustering
- **Single-linkage clustering**
- Connection to ultrametrics and stability
- Hierarchical clustering and phylogenetic trees

Single-linkage clustering

$$\delta(C, C') := \min_{p \in C, p' \in C'} \mathrm{d}(p, p')$$

(note: δ increases from 0 to ∞ during agglomerative process)

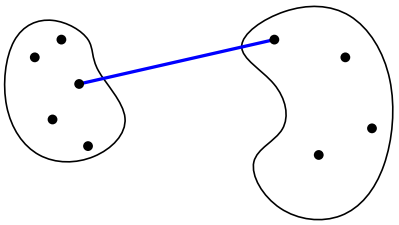


Def: given $s \geq 0$ and $p_i \in P$, call $C_s(p_i)$ the cluster containing p_i when $\delta = s$

(note: $s \leq t \Rightarrow C_s(p_i) \subseteq C_t(p_i)$ — merges only)

Single-linkage clustering

$\delta(C, C') := \min_{p \in C, p' \in C'} \mathrm{d}(p, p')$

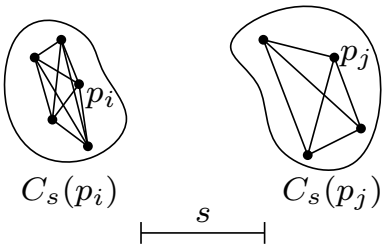


(note: δ increases from 0 to ∞ during agglomerative process)

Prop: (invariant) For any $p_i \in P$ and any $s \geq 0$,

$C_s(p_i) \equiv$ connected component (CC) of s-neighborhood graph

$(P, \{(p_k, p_l) \mid \mathrm{d}(p_k, p_l) \leq s\})$ that contains p_i

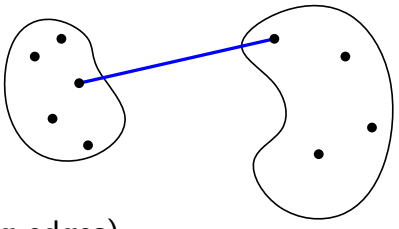


proof (by induction on s):

- true for $s = 0$ ($\delta = 0 \Rightarrow$ singleton clusters and CCs)
- if true for $\delta = s$, then still true for $\delta = t$ for any $t > s$ such that interval $(s, t]$ contains no pairwise distance $\mathrm{d}(p_k, p_l)$
- true for $s = \mathrm{d}(p_k, p_l)$:
  edge (p_k, p_l) changes the CCs of the graph iff $\mathrm{d}(p_k, p_l)$ is cluster-connecting □

Single-linkage clustering

$\delta(C, C') := \min_{p \in C, p' \in C'} \mathrm{d}(p, p')$



(note: δ increases from 0 to ∞ during agglomerative process)

Algo.: (Kruskal — minimum spanning tree $\equiv$ set of connecting edges)

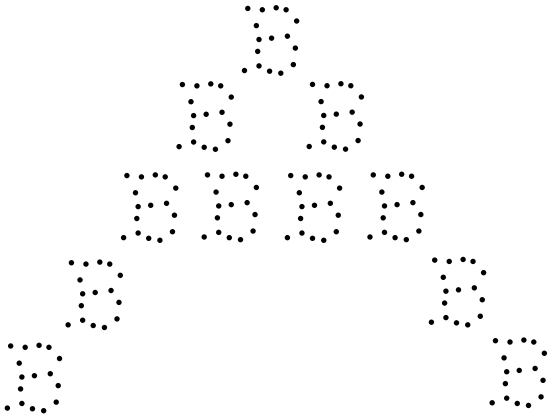
1. sort edges of complete graph on P by increasing length
2. iterate over the edges in this order:
   - ▶ if the next edge connects 2 points from different clusters C, C', then it triggers the next merge $C, C' \mapsto C \cup C'$
   - ▶ else the edge can be ignored

Details / complexity:

- sort edges in $O(n^2 \log n)$ time using merge sort
- iterate over edges in $O(n^2 \alpha(n))$ time using disjoint-set forest

inverse Ackermann function

Example

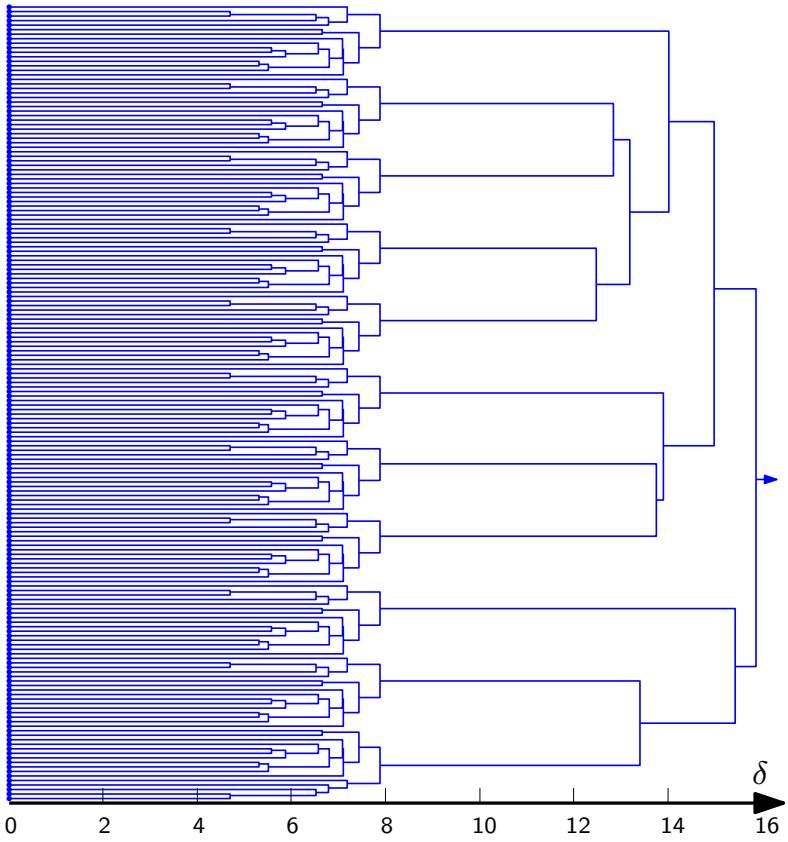


when $\delta = s$:

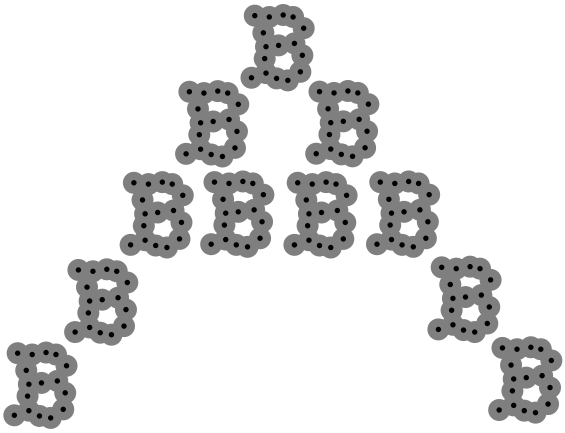
s-neighborhood graph

$\equiv$

intersection graph of balls of radius $s/2$



Example

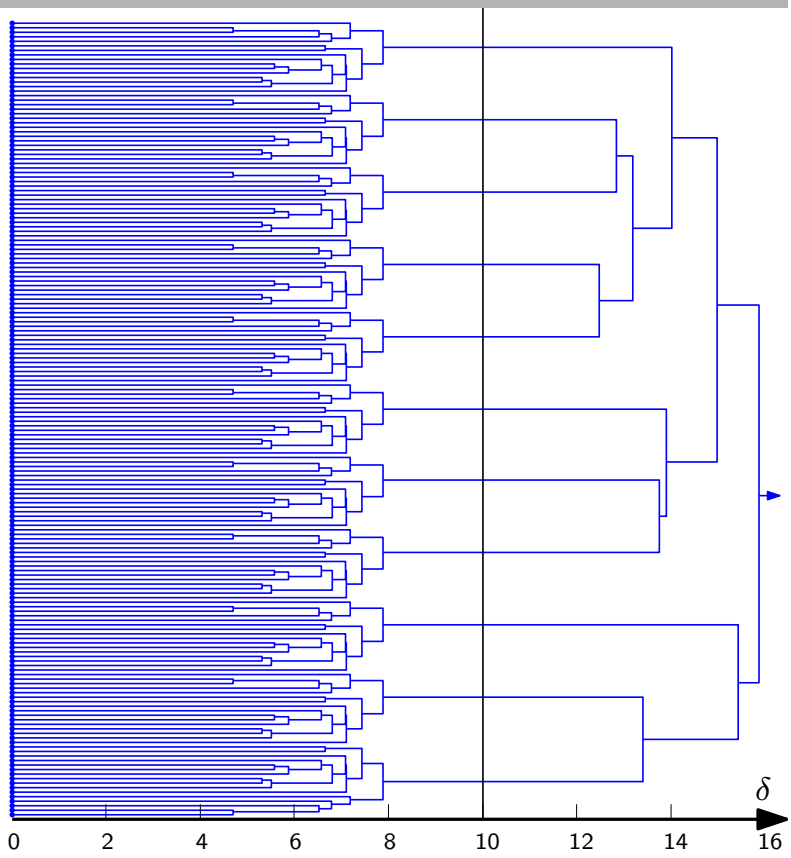


when $\delta = s$:

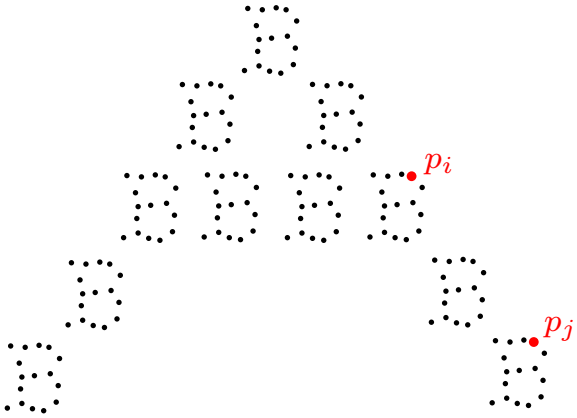
s-neighborhood graph

$\equiv$

intersection graph of balls of radius $s/2$



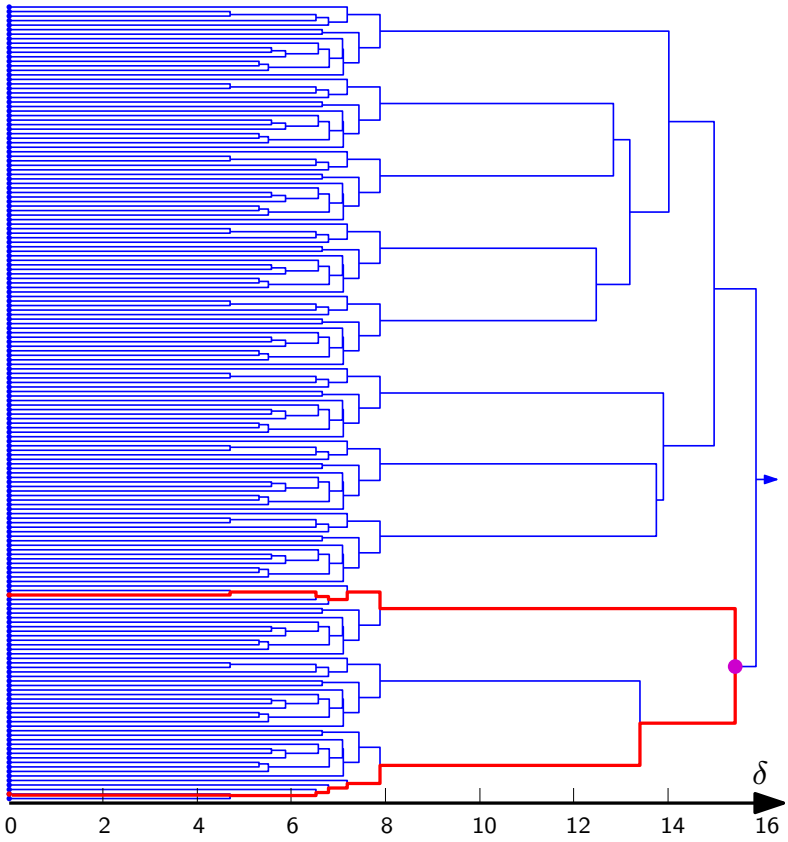
Example



height of least common ancestor (LCA)

$=$

smallest s for which $C_s(p_i) = C_s(p_j)$



Outline

- Principles of hierarchical clustering
- Agglomerative hierarchical clustering
- Single-linkage clustering
- **Connection to ultrametrics and stability**
- Hierarchical clustering and phylogenetic trees

Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}^{\theta}_{\mathrm{LCA}}$ is an **ultrametric** on P

true for any dendrogram,

not only that of single-linkage

note: $\mathrm{d} : P \times P \to \mathbb{R}$ ultrametric if:

- non-negativity: $d \geq 0$
- symmetry: $\mathrm{d}(p_i, p_j) = \mathrm{d}(p_j, p_i)$
- identity: $\mathrm{d}(p_i, p_j) = 0 \quad \Longrightarrow \quad p_i = p_j$
- ultrametric inequality: $\mathrm{d}(p_i, p_k) \leq \max\{\mathrm{d}(p_i, p_j), \mathrm{d}(p_j, p_k)\}$

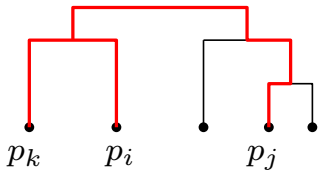
Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}^{\theta}_{\mathrm{LCA}}$ is an **ultrametric** on P

proof:

- non-negativity: height of LCA is ≥ 0
- symmetry: $\mathrm{LCA}(p_i, p_j) = \mathrm{LCA}(p_j, p_i) \quad \Longrightarrow \quad \mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) = \mathrm{d}^{\theta}_{\mathrm{LCA}}(p_j, p_i)$
- identity: $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) = 0 \quad \Longrightarrow \quad p_i = \mathrm{LCA}(p_i, p_j) = p_j$
- ultrametric inequality: $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_k) \leq \max\{\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j), \mathrm{d}^{\theta}_{\mathrm{LCA}}(p_j, p_k)\}$



height of lowest path $p_i \to p_k \leq$ height of path $p_i \to p_j \to p_k$

□

Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}^{\theta}_{\mathrm{LCA}}$ is an **ultrametric** on P

Thm: The map $\Psi : \theta \mapsto \mathrm{d}^{\theta}_{\mathrm{LCA}}$ is a bijection between the dendrograms on P and the ultrametrics on P. Its inverse Ψ^{-1} sends any ultrametric d to the dendrogram $\theta^{\mathrm{d}}_{\mathrm{SL}}$ given by single-linkage clustering on (P, d).

proof: see INF631 (lecture 3)

Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}^{\theta}_{\mathrm{LCA}}$ is an **ultrametric** on P

Thm: The map $\Psi : \theta \mapsto \mathrm{d}^{\theta}_{\mathrm{LCA}}$ is a bijection between the dendrograms on P and the ultrametrics on P. Its inverse Ψ^{-1} sends any ultrametric d to the dendrogram $\theta^{\mathrm{d}}_{\mathrm{SL}}$ given by single-linkage clustering on (P, d).

Thm: [Carlsson, Mémoli 2010]

For any metrics $\mathrm{d}, \mathrm{d}' : P \times P \to \mathbb{R}$ (possibly not ultrametrics),

$$\max_{P\times P} |\mathrm{d}^{\theta^{\mathrm{d}}_{\mathrm{SL}}}_{\mathrm{LCA}} - \mathrm{d}^{\theta^{\mathrm{d}'}_{\mathrm{SL}}}_{\mathrm{LCA}}| \leq \max_{P\times P} |\mathrm{d} - \mathrm{d}'|$$

(note: by previous theorem, inequality is an equality when d, d' are ultrametrics)

proof: see INF631 (lecture 3)

`https://www.enseignement.polytechnique.fr/informatique/INF631/`

Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}^{\theta}_{\mathrm{LCA}}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}^{\theta}_{\mathrm{LCA}}$ is an **ultrametric** on P

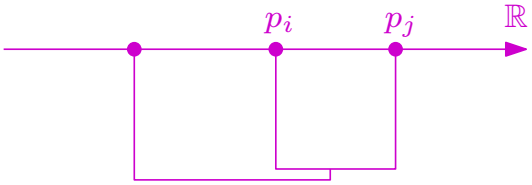
Thm: The map $\Psi : \theta \mapsto \mathrm{d}^{\theta}_{\mathrm{LCA}}$ is a bijection between the dendrograms on P and the ultrametrics on P. Its inverse Ψ^{-1} sends any ultrametric d to the dendrogram $\theta^{\mathrm{d}}_{\mathrm{SL}}$ given by single-linkage clustering on (P, d).

Thm: [Carlsson, Mémoli 2010]

For any metrics $\mathrm{d}, \mathrm{d}' : P \times P \to \mathbb{R}$ (possibly not ultrametrics),

$$\max_{P\times P} |\mathrm{d}^{\theta^{\mathrm{d}}_{\mathrm{SL}}}_{\mathrm{LCA}} - \mathrm{d}^{\theta^{\mathrm{d}'}_{\mathrm{SL}}}_{\mathrm{LCA}}| \leq \max_{P\times P} |\mathrm{d} - \mathrm{d}'|$$

$\to$ height of LCA is stable... but not LCA itself



Connection to ultrametrics and stability

Def: Given θ a dendrogram, $\mathrm{d}_{\mathrm{LCA}}^{\theta}(p_i, p_j) :=$ height of LCA in θ

Prop: $\mathrm{d}_{\mathrm{LCA}}^{\theta}$ is an **ultrametric** on P

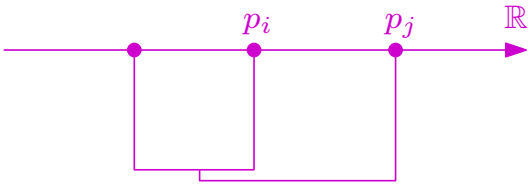
Thm: The map $\Psi : \theta \mapsto \mathrm{d}_{\mathrm{LCA}}^{\theta}$ is a bijection between the dendrograms on P and the ultrametrics on P. Its inverse Ψ^{-1} sends any ultrametric d to the dendrogram $\theta_{\mathrm{SL}}^{\mathrm{d}}$ given by single-linkage clustering on (P, d).

Thm: [Carlsson, Mémoli 2010]

For any metrics $\mathrm{d}, \mathrm{d}' : P \times P \to \mathbb{R}$ (possibly not ultrametrics),

$$\max_{P\times P} |\mathrm{d}_{\mathrm{LCA}}^{\theta_{\mathrm{SL}}^{\mathrm{d}}} - \mathrm{d}_{\mathrm{LCA}}^{\theta_{\mathrm{SL}}^{\mathrm{d}'}}| \leq \max_{P\times P} |\mathrm{d} - \mathrm{d}'|$$

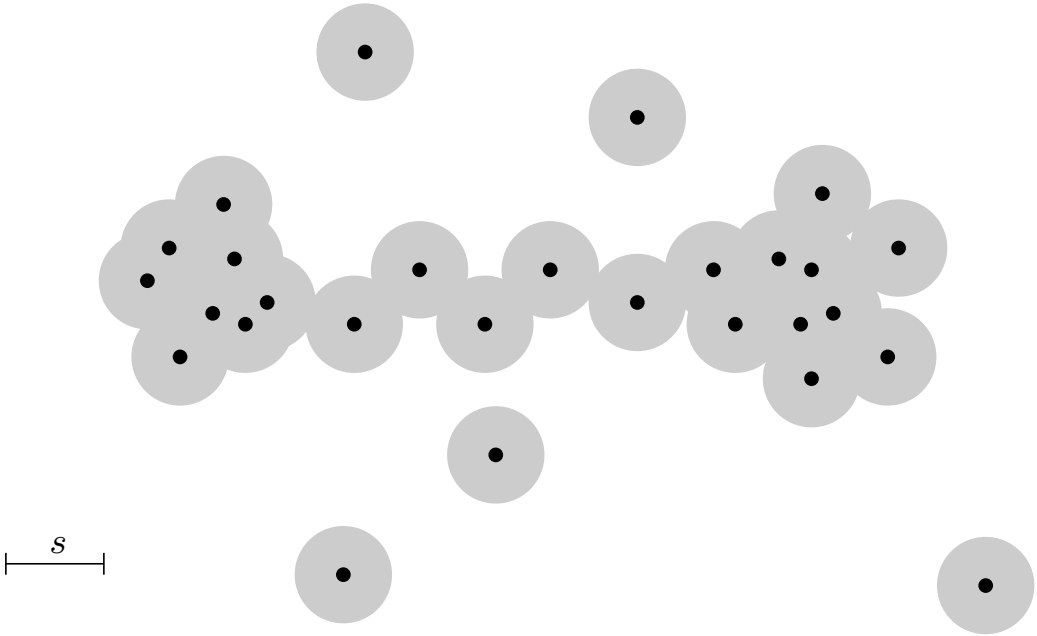
$\to$ height of LCA is stable... but not LCA itself



Connection to ultrametrics and stability

Chaining effect: a few outliers distort the metric in worst case

$\implies$ clusters get merged far earlier than expected

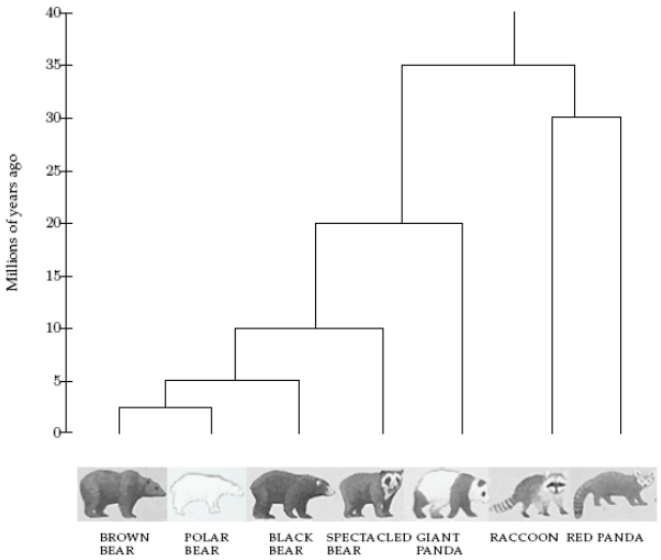


Outline

- Principles of hierarchical clustering
- Agglomerative hierarchical clustering
- Single-linkage clustering
- Connection to ultrametrics and stability
- **Hierarchical clustering and phylogenetic trees**

Hierarchical clustering and phylogenetic trees

A (rooted) **phylogenetic tree** shows the evolutionary relationships among various biological species, based upon some measure of (dis-)similarity between species.

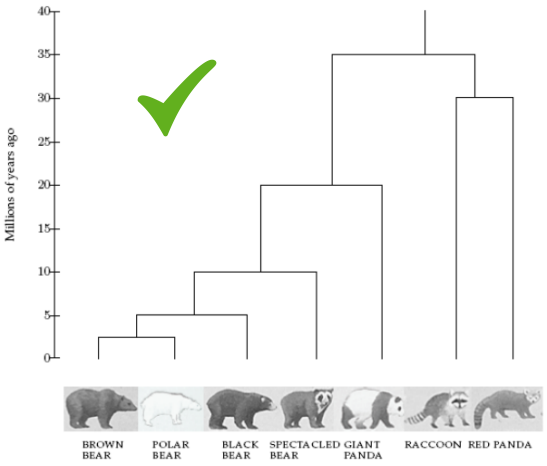


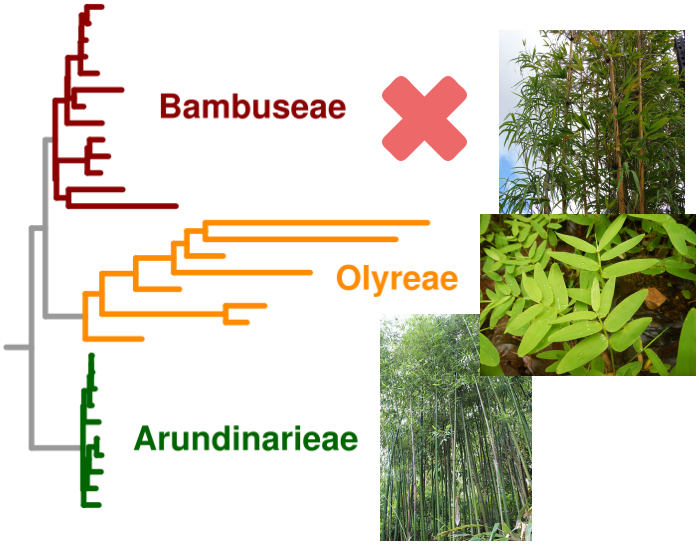
▶ **Goal:** given a collection of species together with a distance/dissimilarity measure d, build a tree whose LCA distance is as close to d as possible.

Hierarchical clustering and phylogenetic trees

Common approach: build a dendrogram θ from d

- approximate d by an ultrametric d^{θ}
- all species lie at the same level / evolve at the same rate (**molecular clock**)





Hierarchical clustering and phylogenetic trees

UPGMA algorithm:

Input: set P of species, distance $\mathrm{d} : P \times P \to \mathbb{R}$ given as an $n \times n$ matrix M

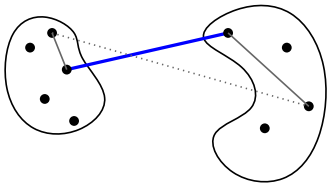
Process:

- run average-linkage clustering on $M \rightsquigarrow$ dendrogram $\theta^{\mathrm{d}}_{\mathrm{AL}}$
- scale $\theta^{\mathrm{d}}_{\mathrm{AL}}$ by a factor of $1/2 \rightsquigarrow$ dendrogram $\theta'^{\mathrm{d}}_{\mathrm{AL}}$:
  - each leaf is assigned height 0
  - each internal node (corresponding to a merge $C \cup C'$) is assigned height $\frac{1}{2}\,\delta_{\mathrm{AL}}(C, C')$

Output: $\theta'^{\mathrm{d}}_{\mathrm{AL}}$

Thm: If d is an ultrametric then $\theta^{\mathrm{d}}_{\mathrm{AL}} = \theta^{\mathrm{d}}_{\mathrm{SL}}$.

Therefore, $d = \mathrm{d}^{\theta^{\mathrm{d}}_{\mathrm{SL}}}_{\mathrm{LCA}} = \mathrm{d}^{\theta^{\mathrm{d}}_{\mathrm{AL}}}_{\mathrm{LCA}} = 2\,\mathrm{d}^{\theta'^{\mathrm{d}}_{\mathrm{AL}}}_{\mathrm{LCA}}$.



What you should know

- Concepts: dendrograms, divisive vs. agglomerative
- Meta-algorithm for agglomerative hierarchical clustering
- Geometric and statistical distances, and their impact
- Single-linkage clustering: algorithm, invariant, complexity
- Bijection dendrograms $\longleftrightarrow$ ultrametrics
- Stability result
- Optional: UPGMA algorithm to build phylogenetic trees

Density Estimation

Outline:

- **Mathematical formulation**
- **Quality criteria for density estimators**
- **Parametric vs. non-parametric estimators**
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Outline

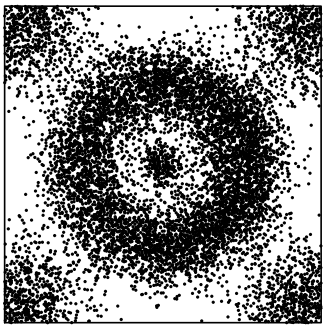
- **Mathematical formulation**
- **Quality criteria for density estimators**
- **Parametric vs. non-parametric estimators**
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Mathematical formulation

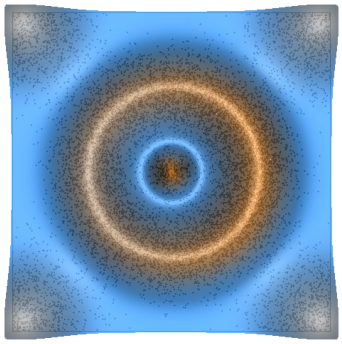
Input: $P_n = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$

Prior: $p_i \overset{\text{iid}}{\sim} \nu$ for some unknown probability measure ν with density $f : \mathbb{R}^d \to \mathbb{R}$

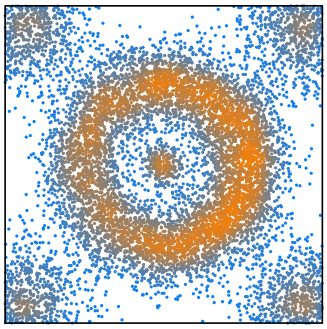
Goal: estimate f from P_n, i.e. build an estimator $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}$



Input: $P_n \subset \mathbb{R}^2$



Prior: $\exists \nu$ w/ density f



Output: $\hat{f}_n$

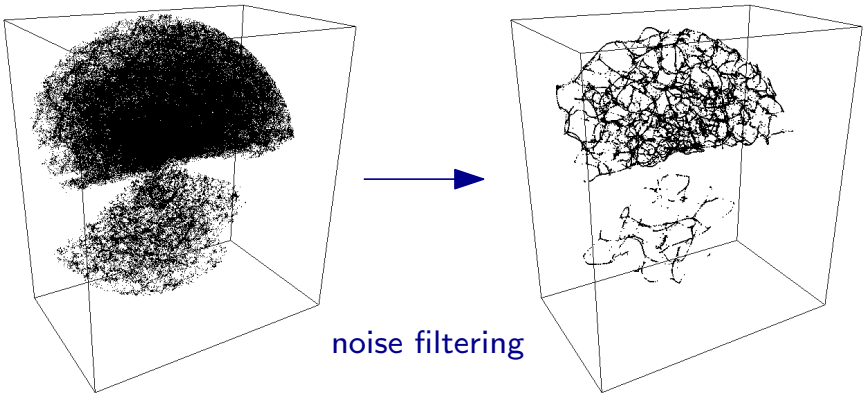
Mathematical formulation

Input: $P_n = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$

Prior: $p_i \overset{\text{iid}}{\sim} \nu$ for some unknown probability measure ν with density $f : \mathbb{R}^d \to \mathbb{R}$

Goal: estimate f from P_n, i.e. build an estimator $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}$

Sample applications:



noise filtering

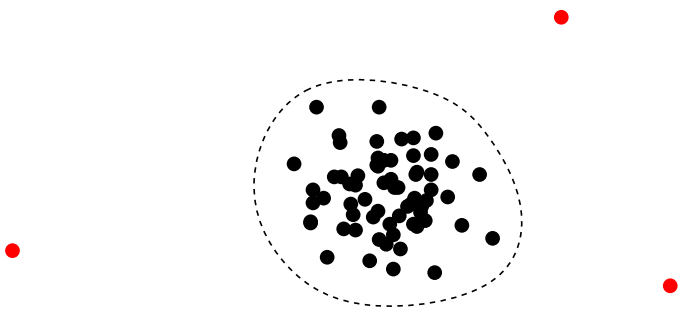
Mathematical formulation

Input: $P_n = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$

Prior: $p_i \overset{\text{iid}}{\sim} \nu$ for some unknown probability measure ν with density $f : \mathbb{R}^d \to \mathbb{R}$

Goal: estimate f from P_n, i.e. build an estimator $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}$

Sample applications:



outlier detection/removal

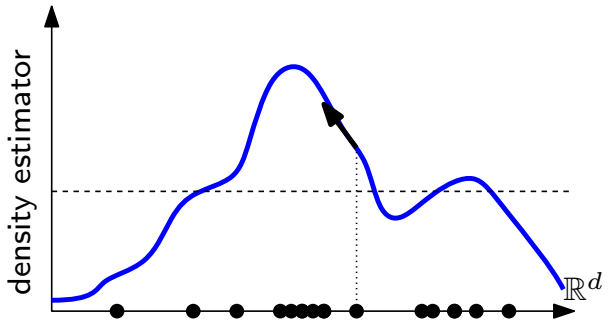
Mathematical formulation

Input: $P_n = \{p_1, \cdots, p_n\} \subset \mathbb{R}^d$

Prior: $p_i \overset{\text{iid}}{\sim} \nu$ for some unknown probability measure ν with density $f : \mathbb{R}^d \to \mathbb{R}$

Goal: estimate f from P_n, i.e. build an estimator $\hat{f}_n : \mathbb{R}^d \to \mathbb{R}$

Sample applications:



clustering (DBSCAN, mean-shift, etc.)

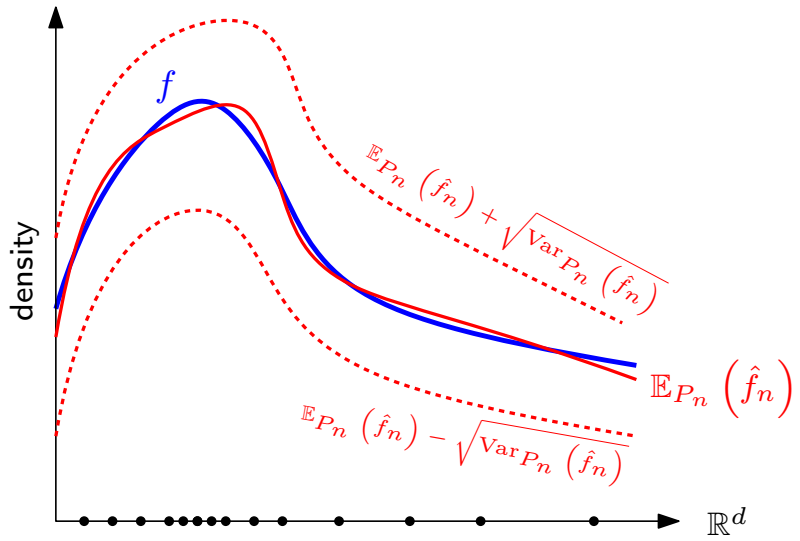
Outline

- Mathematical formulation
- **Quality criteria for density estimators**
- Parametric vs. non-parametric estimators
- Parametric estimators:
  - Gaussian model
  - Gaussian mixture models (GMMs)
- Non-parametric estimators:
  - histograms
  - kernel density estimators

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$



Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

$\rightarrow$ $\hat{f}_n(x)$ is biased if $\mathrm{Bias}_{P_n}(\hat{f}_n(x)) \neq 0$

$\rightarrow$ $\hat{f}_n(x)$ is unbiased if $\mathrm{Bias}_{P_n}(\hat{f}_n(x)) = 0$

$\rightarrow$ $\hat{f}_n(x)$ is asymptotically unbiased if $\lim_{n\to\infty} \mathrm{Bias}_{P_n}(\hat{f}_n(x)) = 0$

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

$\rightarrow$ bias-variance decomposition (allows to measure the bias-variance tradeoff):

Thm: $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) = \mathrm{Bias}_{P_n}(\hat{f}_n(x))^2 + \mathrm{Var}_{P_n}(\hat{f}_n(x))$

Proof sketch: decompose $\hat{f}_n(x) - f(x) = \hat{f}_n(x) - \mathbb{E}_{P_n}\hat{f}_n(x) + \mathbb{E}_{P_n}\hat{f}_n(x) - f(x)$ then follow your intuition... $\square$

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

▶ Proof of bias-variance decomposition:

$$\begin{aligned}
\mathbb{E}_{P_n}\left(\hat{f}_n(x) - f(x)\right)^2 &= \mathbb{E}_{P_n}\left(\hat{f}_n(x) - \mathbb{E}_{P_n}\hat{f}_n(x) + \mathbb{E}_{P_n}\hat{f}_n(x) - f(x)\right)^2 \\
&= \mathbb{E}_{P_n}\left(\hat{f}_n(x) - \mathbb{E}_{P_n}\hat{f}_n(x)\right)^2 + \mathbb{E}_{P_n}\left(\mathbb{E}_{P_n}\hat{f}_n(x) - f(x)\right)^2 \\
&\quad + 2\,\mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - \mathbb{E}_{P_n}\hat{f}_n(x)\right)\left(\mathbb{E}_{P_n}\hat{f}_n(x) - f(x)\right)\right) \\
&= \mathrm{Var}_{P_n}\hat{f}_n(x) + \mathrm{Bias}_{P_n}(\hat{f}_n(x))^2 + \mathrm{cst}\cdot\mathbb{E}_{P_n}\left(\hat{f}_n(x) - \mathbb{E}_{P_n}\hat{f}_n(x)\right) \\
&= \mathrm{Var}_{P_n}\hat{f}_n(x) + \mathrm{Bias}_{P_n}(\hat{f}_n(x))^2. \qquad \square
\end{aligned}$$

cst.

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

Convergence (consistency): $\underbrace{\operatorname{plim}_{n\to\infty}}\, \hat{f}_n(x) = f(x)$

convergence in probability: $\forall \varepsilon > 0 \quad \lim_{n\to\infty} \mathbb{P}_{P_n}(|\hat{f}_n(x) - f(x)| > \varepsilon) = 0$

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

Convergence (consistency): $\operatorname{plim}_{n\to\infty} \hat{f}_n(x) = f(x)$

Prop: $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) \overset{n\to\infty}{\longrightarrow} 0 \implies \hat{f}_n(x)$ is consistent

Proof sketch: let $X_n = \left|\hat{f}_n(x) - f(x)\right|$. Using Chebyshev's inequality, convergence in L^2 (i.e. $\mathbb{E}\, X_n^2 \overset{n\to\infty}{\longrightarrow} 0$) implies convergence in probability. $\square$

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

Convergence (consistency): $\operatorname{plim}_{n\to\infty} \hat{f}_n(x) = f(x)$

▶ Proof of consistency: let $\varepsilon > 0$ and define $X_n = \left|\hat{f}_n(x) - f(x)\right|$.

$$\mathbb{E}X_n^2 \overset{n\to\infty}{\longrightarrow} 0 \implies \mathrm{Var}X_n + (\mathbb{E}X_n)^2 \longrightarrow 0 \implies \mathrm{Var}X_n \longrightarrow 0 \text{ and } |\mathbb{E}X_n| \longrightarrow 0.$$

$$\exists N \in \mathbb{N} \text{ s.t. } \forall n > N,\ |\mathbb{E}X_n| \le \varepsilon/2$$

$$\implies \forall n > N,\ \mathbb{P}(|X_n| \ge \varepsilon) \le \mathbb{P}(|X_n - \mathbb{E}X_n| \ge \varepsilon/2) \le \tfrac{4}{\varepsilon^2}\mathrm{Var}X_n \longrightarrow 0.$$

(Chebyshev)

Quality of a density estimator

Note: For a fixed $x \in \mathbb{R}^d$, the estimate $\hat{f}_n(x)$ is itself a random variable in $\mathbb{R}$

$\rightarrow$ mean/expectation $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right)$, variance $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right)$

Bias: $\mathrm{Bias}_{Pn}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)$

Mean Squared Error (MSE): $\mathrm{MSE}_{P_n}(\hat{f}_n(x)) := \mathbb{E}_{P_n}\left(\left(\hat{f}_n(x) - f(x)\right)^2\right)$

Convergence (consistency): $\operatorname{plim}_{n\to\infty} \hat{f}_n(x) = f(x)$

Robustness: (sloppy def.)

expectation, variance, bias, MSE of estimator are not perturbed too much by adding outliers (e.g. consistency and asymptotic unbiasedness preserved)

Outline

- **Mathematical formulation**
- **Quality criteria for density estimators**
- **Parametric vs. non-parametric estimators**
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Parametric vs. non-parametric estimation

Parametric estimator: assumes that ν belongs to a known parametrized family

$\rightarrow$ approach: compute estimate(s) of parameter(s) then take corresponding $\hat{f}_n$

$\rightarrow$ examples

- Normal/Gaussian, Poisson, exponential families, etc.
- mixture models

Nonparametric estimator: no underlying parametrized family assumed

$\rightarrow$ approach: estimate f pointwise directly

$\rightarrow$ examples

- histograms
- kernel density estimators
- k-NN estimator

Outline

- Mathematical formulation
- Quality criteria for density estimators
- Parametric vs. non-parametric estimators
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Gaussian model

Univariate case ($d = 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}(\mu, \sigma^2)$

$\rightarrow$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\sigma}_n^2$ of mean and variance, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{2\pi\hat{\sigma}_n^2}} \exp\left(-\frac{(x-\hat{\mu}_n)^2}{2\hat{\sigma}_n^2}\right)$$

consistent
(continuous mapping theorem)

Estimators:

- empirical mean: $\hat{\mu}_n := \frac{1}{n}\sum_{i=1}^{n} p_i$ $\rightarrow$ unbiased, consistent (law of large numbers)
- empirical variance: $\hat{\sigma}_n^2 := \frac{1}{n}\sum_{i=1}^{n}(p_i - \hat{\mu}_n)^2$ $\rightarrow$ negatively biased, consistent
  (Cochran's theorem)
- corrected empirical variance: $\hat{\sigma}_n^2 := \frac{1}{n-1}\sum_{i=1}^{n}(p_i - \hat{\mu}_n)^2$ $\rightarrow$ unbiased, consistent

Gaussian model

Univariate case ($d = 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}(\mu, \sigma^2)$

$\rightarrow$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\sigma}_n^2$ of mean and variance, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{2\pi\hat{\sigma}_n^2}} \exp\left(-\frac{(x-\hat{\mu}_n)^2}{2\hat{\sigma}_n^2}\right)$$

Rates of convergence:

- empirical mean: $|\hat{\mu}_n - \mu| = \underbrace{O_{P_n}}\left(1/\sqrt{n}\right)$ (Berry-Esseen theorem)
  stochastic boundedness: $\forall \varepsilon > 0\ \exists \Delta, N > 0$ s.t. $\mathbb{P}_{P_n}\left(|\hat{\mu}_n - \mu|\sqrt{n} > \Delta\right) < \varepsilon$, $\forall n > N$
- empirical variance (corrected or not): $|\hat{\sigma}_n^2 - \sigma^2| = O_{P_n}\left(1/\sqrt{n}\right)$
- density estimator: $|\hat{f}_n(x) - f(x)| = O_{P_n}\left(1/\sqrt{n}\right)$ (follows from a calculation)

PARAMETRIC

Gaussian model

Univariate case ($d = 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}(\mu, \sigma^2)$

$\rightarrow$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\sigma}_n^2$ of mean and variance, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{2\pi\hat{\sigma}_n^2}} \exp\left(-\frac{(x-\hat{\mu}_n)^2}{2\hat{\sigma}_n^2}\right)$$

Non-Gaussian case: (ν has mean μ and variance σ^2 but $\nu \neq \mathcal{N}(\mu, \sigma^2)$)

- $\hat{f}_n(x)$ still converges to $\bar{f}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ at rate $1/\sqrt{n}$
- $|\bar{f}(x) - f(x)| \neq 0$ (constant bias)

Gaussian model

Multivariate case ($d > 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}_d(\mu, \Sigma)$

→ hypothesis: covariance matrix Σ is **non-singular**

→ density function: $f(x) = \dfrac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\dfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$

PARAMETRIC

Gaussian model

Multivariate case ($d > 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}_d(\mu, \Sigma)$

→ approach: compute estimates $\hat{\mu}_n$ and $\hat{\Sigma}_n$ of μ and Σ, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{(2\pi)^d \det \hat{\Sigma}_n}} \exp\left(-\frac{1}{2}(x-\hat{\mu}_n)^T \hat{\Sigma}_n^{-1}(x-\hat{\mu}_n)\right)$$

Estimators:

- empirical mean: $\hat{\mu}_n := \dfrac{1}{n}\sum_{i=1}^{n} p_i$ → unbiased, consistent
- empirical covariance: $\hat{\Sigma}_n := \dfrac{1}{n}\sum_{i=1}^{n}(p_i - \hat{\mu}_n)(p_i - \hat{\mu}_n)^T$ → biased, consistent
- corr. emp. cov.: $\hat{\Sigma}_n := \dfrac{1}{n-1}\sum_{i=1}^{n}(p_i - \hat{\mu}_n)(p_i - \hat{\mu}_n)^T$ → unbiased, consistent

Gaussian model

Multivariate case ($d > 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}_d(\mu, \Sigma)$

$\to$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\Sigma}_n$ of μ and Σ, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{(2\pi)^d \det \hat{\Sigma}_n}} \exp\left(-\frac{1}{2}(x - \hat{\mu}_n)^T \hat{\Sigma}_n^{-1}(x - \hat{\mu}_n)\right)$$

consistent
(continuous mapping theorem)

Estimators:

- empirical mean: $\hat{\mu}_n := \frac{1}{n}\sum_{i=1}^n p_i$ $\to$ unbiased, consistent
- empirical covariance: $\hat{\Sigma}_n := \frac{1}{n}\sum_{i=1}^n (p_i - \hat{\mu}_n)(p_i - \hat{\mu}_n)^T$ $\to$ biased, consistent
- corr. emp. cov.: $\hat{\Sigma}_n := \frac{1}{n-1}\sum_{i=1}^n (p_i - \hat{\mu}_n)(p_i - \hat{\mu}_n)^T$ $\to$ unbiased, consistent

PARAMETRIC

Gaussian model

Multivariate case ($d > 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}_d(\mu, \Sigma)$

$\to$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\Sigma}_n$ of μ and Σ, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{(2\pi)^d \det \hat{\Sigma}_n}} \exp\left(-\frac{1}{2}(x - \hat{\mu}_n)^T \hat{\Sigma}_n^{-1}(x - \hat{\mu}_n)\right)$$

Rates of convergence:

- empirical mean: $\|\hat{\mu}_n - \mu\|_2 = O_{P_n}\left(\sqrt{d/n}\right)$ (mean is defined coordinate-wise)
- empirical covariance (corrected or not): $\|\hat{\Sigma}_n - \Sigma\| = O_{P_n}\left(\sqrt{d/n}\right)$

operator norm

- density estimator: $|\hat{f}_n(x) - f(x)| = O_{P_n}\left(\sqrt{d/n}\right)$ (same calculation as in 1-d)

Gaussian model

Multivariate case ($d > 1$): $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \nu = \mathcal{N}_d(\mu, \Sigma)$

$\rightarrow$ approach: compute estimates $\hat{\mu}_n$ and $\hat{\Sigma}_n$ of μ and Σ, then define:

$$\hat{f}_n(x) := \frac{1}{\sqrt{(2\pi)^d \det \hat{\Sigma}_n}} \exp\left(-\frac{1}{2}(x - \hat{\mu}_n)^T \hat{\Sigma}_n^{-1}(x - \hat{\mu}_n)\right)$$

Non-Gaussian case: (ν has mean μ and covariance matrix Σ but $\nu \neq \mathcal{N}(\mu, \Sigma)$)

- $\hat{f}_n(x)$ still converges to $\bar{f}(x) := \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$ at rate $\sqrt{d/n}$
- $|\bar{f}(x) - f(x)| \neq 0$ (constant bias)

Outline

- Mathematical formulation
- Quality criteria for density estimators
- Parametric vs. non-parametric estimators
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Gaussian mixture models (GMMs)

Aim: remove bias $|\bar{f}(x) - f(x)|$ when f is not Gaussian

Target density: mixture density $\bar{f}(x) = \sum_{l=1}^{r} \xi_l \, \phi_l(x)$, where:

- each ϕ_l is a pobability density function (pdf)
- each weight ξ_l is ≥ 0
- the weights ξ_l sum up to 1: $\sum_{l=1}^{r} \xi_l = 1$

Our setup: Gaussian mixture: each ϕ_l is the pdf of some $\mathcal{N}_d(\mu_l, \Sigma_l)$:

$$\phi_l(x) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_l}} \exp\left(-\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1} (x-\mu_l)\right) =: \Phi_{\mu_l, \Sigma_l}(x)$$

Underlying generative model: $p_1, \cdots, p_n \overset{\text{iid}}{\sim} \sum_{l=1}^{r} \xi_l \, \mathcal{N}_d(\mu_l, \Sigma_l)$ (GMM)

Gaussian mixture models (GMMs)

Aim: remove bias $|\bar{f}(x) - f(x)|$ when f is not Gaussian

Target density: mixture density $\bar{f}(x) = \sum_{l=1}^{r} \xi_l \, \phi_l(x)$, where:

- each ϕ_l is a pobability density function (pdf)
- each weight ξ_l is ≥ 0
- the weights ξ_l sum up to 1: $\sum_{l=1}^{r} \xi_l = 1$

Our setup: Gaussian mixture: each ϕ_l is the pdf of some $\mathcal{N}_d(\mu_l, \Sigma_l)$:

Thm: [Bacharoglou 2010]
For every continuous, compactly supported, pdf f on $\mathbb{R}$, there is a sequence $(\bar{f}_n)_{n\geq 1}$ of Gaussian mixtures (of finite length $r_n \to \infty$) s.t. $\|\bar{f}_n - f\|_\infty \to 0$.

PARAMETRIC

Gaussian mixture models (GMMs)

Approach: for a fixed r, estimate the weights ξ_l and parameters μ_l, Σ_l, then define:

$$\hat{f}_n(x) := \sum_{l=1}^{r} \hat{\xi}_l \, \Phi_{\hat{\mu}_l, \hat{\Sigma}_l}(x)$$

Maximum likelihood estimation (MLE):

$$(\hat{\xi}_l, \hat{\mu}_l, \hat{\Sigma}_l)_{l=1}^{r} := \underset{(\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}}{\operatorname{argmax}} \underbrace{\sum_{i=1}^{n} \log \left(\sum_{l=1}^{r} \xi_l \, \Phi_{\mu_l, \Sigma_l}(p_i) \right)}_{\text{log-likelihood of } (p_i)_{i=1}^{n} \text{ given } (\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}}$$

- likelihood of parameters $(\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}$:= probability of observing the sample P_n

$$\mathcal{L}(p_i;\, (\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}) = \sum_{l=1}^{r} \xi_l \, \Phi_{\mu_l, \Sigma_l}(p_i) \qquad \text{(mixture density)}$$

$$\mathcal{L}((p_i)_{i=1}^{n};\, (\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}) = \prod_{i=1}^{n} \mathcal{L}(p_i;\, (\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}) \qquad \text{(independence)}$$

Gaussian mixture models (GMMs)

Approach: for a fixed r, estimate the weights ξ_l and parameters μ_l, Σ_l, then define:

$$\hat{f}_n(x) := \sum_{l=1}^{r} \hat{\xi}_l \, \Phi_{\hat{\mu}_l, \hat{\Sigma}_l}(x)$$

Maximum likelihood estimation (MLE):

$$(\hat{\xi}_l, \hat{\mu}_l, \hat{\Sigma}_l)_{l=1}^{r} := \operatorname*{argmax}_{(\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}} \underbrace{\sum_{i=1}^{n} \log \left(\sum_{l=1}^{r} \xi_l \, \Phi_{\mu_l, \Sigma_l}(p_i) \right)}_{\text{log-likelihood of } (p_i)_{i=1}^{n} \text{ given } (\xi_l, \mu_l, \Sigma_l)_{l=1}^{r}}$$

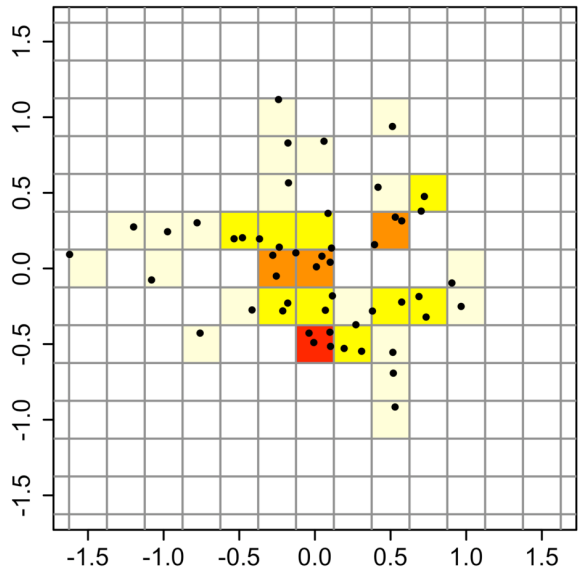
- good asympt. behavior w.r.t. $\bar{f}$: consistency, achieves Cramér–Rao lower bound
- no closed-form solution
- variational solvers (gradient ascent, **E**xpectation-**M**aximization)
- non-concave functional $\Rightarrow$ local maxima, non-unique global maximum
- choice of mixture size r: large bias (small r) vs. large variance (large r)

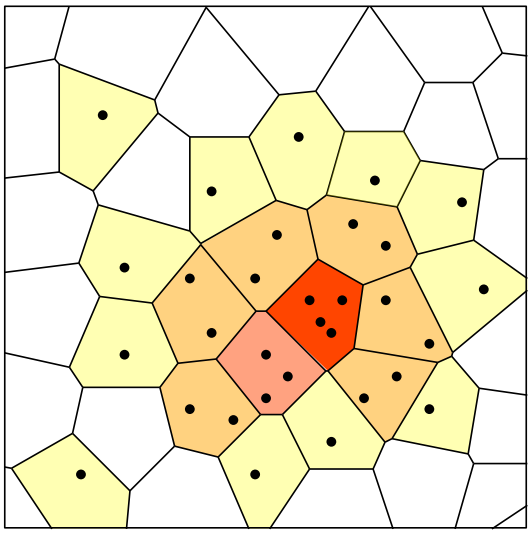
Outline

- Mathematical formulation
- Quality criteria for density estimators
- Parametric vs. non-parametric estimators
- Parametric estimators:
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

Histograms

Principle: Build a tessellation of $\mathbb{R}^d$ (grid, Voronoi diagram, etc.), then record the number of observations in each cell





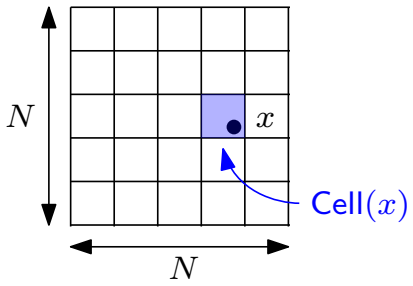
(image source: http://www.wikiwand.com/en/Multivariate_kernel_density_estimation)

Histograms

Principle: Build a tessellation of $\mathbb{R}^d$ (grid, Voronoi diagram, etc.), then record the number of observations in each cell

Uniform grid:

- assume wlog that $P_n \subset [0,1)^d$
- tessellate $[0,1)^d$ with uniform grid of size N^d
- For any $x \in [0,1)^d$, let $\hat{f}_n(x) := \dfrac{\#\text{observations in Cell}(x)}{n \operatorname{Vol}(\text{Cell}(x))}$



$$= \frac{N^d}{n} \left(\sum_{i=1}^{n} \mathbb{1}_{p_i \in \text{Cell}(x)} \right)$$

NONPARAMETRIC

Histograms

(hypothesis: f Lipschitz-continuous)

Bias:

$$\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) = \frac{N^d}{n} \sum_{i=1}^{n} \mathbb{P}\left(p_i \in \text{Cell}(x)\right) = N^d \, \mathbb{P}\left(p_1 \in \text{Cell}(x)\right)$$

$$= N^d \int_{\text{Cell}(x)} f(u)\, du \in [f(y),\ f(z)] \text{ for some } y, z \in \overline{\text{Cell}(x)}$$

(f continuous, $\overline{\text{Cell}(x)}$ compact)

$$= f(x^*) \text{ for some } x^* \in \overline{\text{Cell}(x)}$$

(intermediate value theorem, $\overline{\text{Cell}(x)}$ path-connected)

$$\left|\text{Bias}\left(\hat{f}_n(x)\right)\right| = \left|\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x)\right| = |f(x^*) - f(x)| \leq \text{Lip}_f \frac{\sqrt{d}}{N}$$

(diameter of Cell(x))

Note: bias decreases with N

Histograms

(hypothesis: f Lipschitz-continuous)

Variance:

$$\begin{aligned}\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right) &= \frac{N^{2d}}{n^2}\sum_{i=1}^{n}\mathrm{Var}\left(\mathbb{1}_{p_i\in\mathrm{Cell}(x)}\right) = \frac{N^{2d}}{n}\mathrm{Var}\left(\mathbb{1}_{p_1\in\mathrm{Cell}(x)}\right)\\ &= \frac{N^{2d}}{n}\left(\mathbb{P}\left(p_1\in\mathrm{Cell}(x)\right)-\mathbb{P}\left(p_1\in\mathrm{Cell}(x)\right)^2\right)\\ &= \frac{N^d}{n}\left(f(x^*)-\frac{1}{N^d}f(x^*)^2\right)\end{aligned}$$

Note: variance increases with N

$$\mathrm{MSE}_{P_n}\left(\hat{f}_n(x)\right) \le \mathrm{Lip}_f^2\frac{d}{N^2}+\frac{N^d f(x^*)}{n} \qquad N_{\mathrm{OPT}} = \left(\frac{2n\,\mathrm{Lip}_f^2}{f(x^*)}\right)^{\frac{1}{d+2}}$$

$$\mathrm{MSE}_{\mathrm{OPT}}\left(\hat{f}_n(x)\right) = O\left(n^{-\frac{2}{d+2}}\right)$$

NONPARAMETRIC

Histograms

- N_{OPT} is unknown in practice (must be inferred)
- the grid does not adapt to the shape of the support of ν
- the tessellation can become costly to maintain as d increases

$$\mathrm{MSE}_{P_n}\left(\hat{f}_n(x)\right) \le \mathrm{Lip}_f^2\frac{d}{N^2}+\frac{N^d f(x^*)}{n} \qquad N_{\mathrm{OPT}} = \left(\frac{2n\,\mathrm{Lip}_f^2}{f(x^*)}\right)^{\frac{1}{d+2}}$$

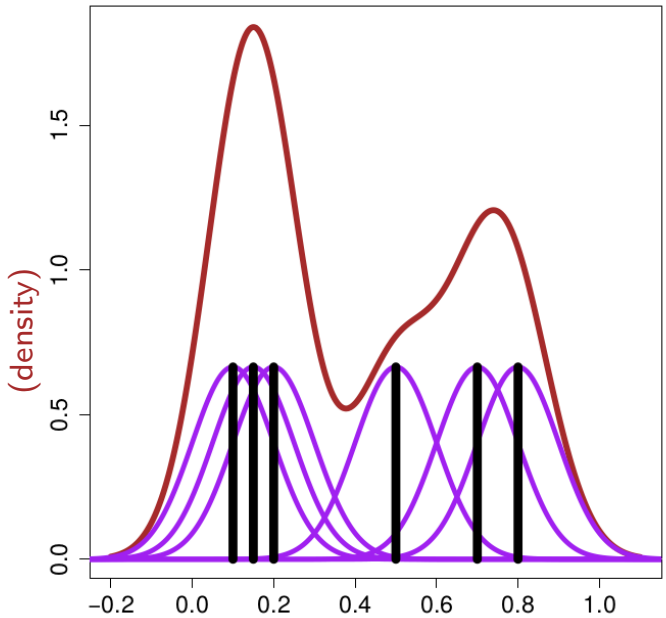
$$\mathrm{MSE}_{\mathrm{OPT}}\left(\hat{f}_n(x)\right) = O\left(n^{-\frac{2}{d+2}}\right)$$

Outline

- **Mathematical formulation**
- **Quality criteria for density estimators**
- **Parametric vs. non-parametric estimators**
- **Parametric estimators:**
  - Gaussian model
  - Gaussian mixture models (GMMs)
- **Non-parametric estimators:**
  - histograms
  - kernel density estimators

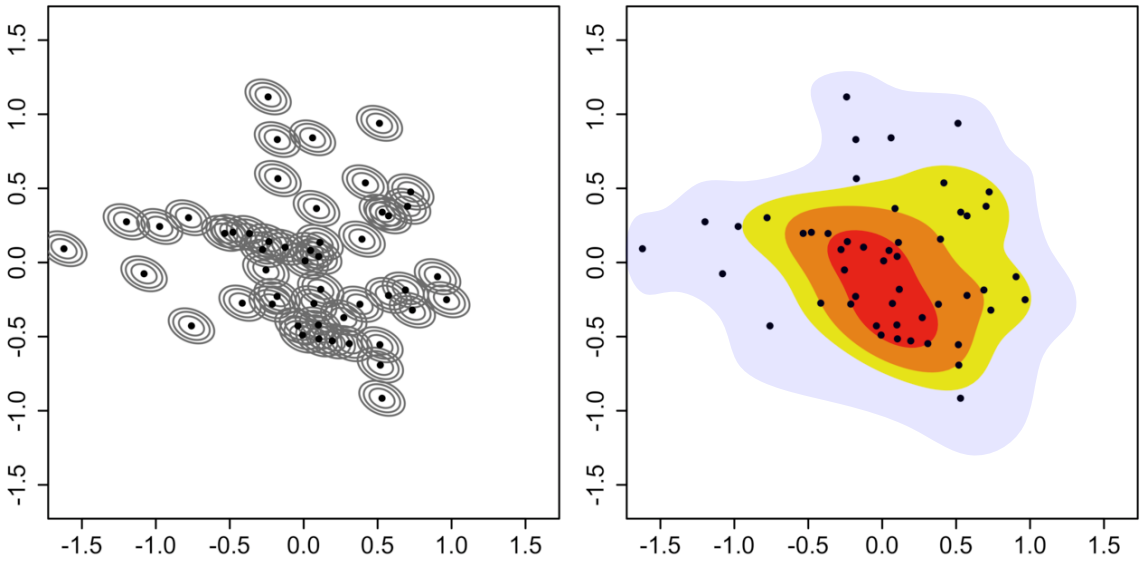
Kernel density estimators

Principle: make $\hat{f}_n$ a mixture of copies of an 'elementary' density (kernel), anchored at each observation



Kernel density estimators

Principle: make $\hat{f}_n$ a mixture of copies of an 'elementary' density (kernel), anchored at each observation



(image source: http://www.wikiwand.com/en/Multivariate_kernel_density_estimation)

NONPARAMETRIC

Kernel density estimators

Principle: make $\hat{f}_n$ a mixture of copies of an 'elementary' density (kernel), anchored at each observation

General formula: (convolution)

$$\hat{f}_n(x) := \frac{1}{n} \sum_{i=1}^{n} K_H(x - p_i), \text{ where } K_H(u) := (\det H)^{-1/2} K(H^{-1/2}u)$$

- H: inner-product (positive-definite) $d \times d$ matrix (adds scaling / anisotropy)
- $K : \mathbb{R}^d \to \mathbb{R}^+$: d-variate kernel:

$$\int_{\mathbb{R}^d} K(u)\, du = 1 \quad \text{(normalized)} \qquad \int_{\mathbb{R}^d} u\, K(u)\, du = 0 \quad \text{(centered at origin)}$$

$$\lim_{\|u\| \to \infty} K(u) = 0 \quad \text{(vanishes at infinity)} \qquad \int_{\mathbb{R}^d} uu^T K(u)\, du = c_K\, I_d \quad \text{(isotropic)}$$

Kernel density estimators

Specialization 1: take $H = \sigma^2 I_d$ (isotropic kernel)

bandwidth / window

Specialization 2: take $K(u) \propto k(\|u\|_2^2)$ for some $k : \mathbb{R}^+ \to \mathbb{R}^+$

(radially-symmetric kernel)

kernel profile

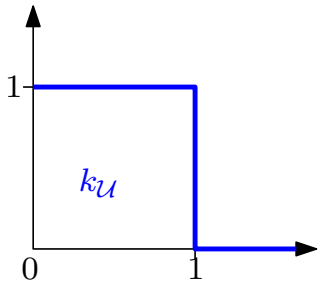
normalizing factor: $c_{k,d} := \left(\int_{\mathbb{R}^d} k(\|u\|_2^2)\, du \right)^{-1}$

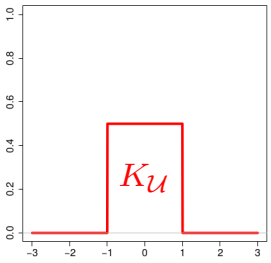
$$\rightsquigarrow \hat{f}_n(x) := \frac{c_{k,d}}{n\,\sigma^d} \sum_{i=1}^{n} k\left(\frac{\|x - p_i\|_2^2}{\sigma^2} \right)$$

Common kernels

Flat / Uniform: $k_{\mathcal{U}}(t) := \begin{cases} 1 \text{ if } t \leq 1 \\ 0 \text{ if } t > 1 \end{cases}$

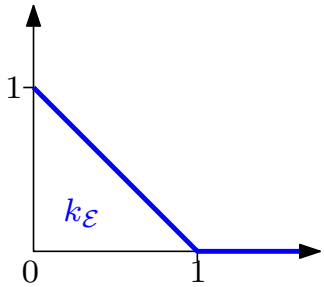
$$\rightsquigarrow c_{k,d} = 1/\mathrm{Vol}\, B_d(0,1) = \frac{\Gamma(d/2+1)}{\pi^{d/2}}$$

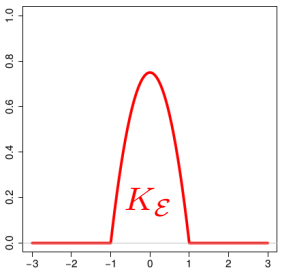




Epanechnikov: $k_{\mathcal{E}}(t) := \begin{cases} 1 - t \text{ if } t \leq 1 \\ 0 \text{ if } t > 1 \end{cases}$

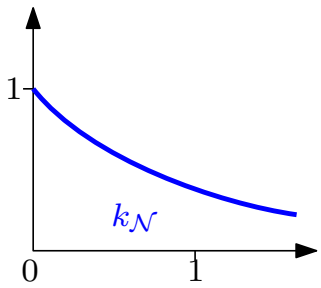
$$\rightsquigarrow c_{k,d} = \frac{d+2}{2\,\mathrm{Vol}\, B_d(0,1)}$$

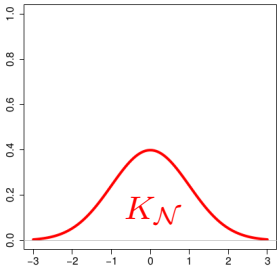




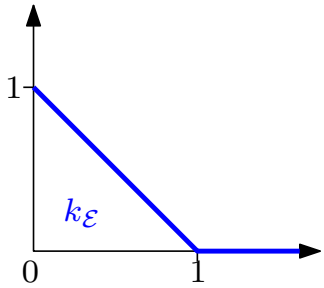
Common kernels

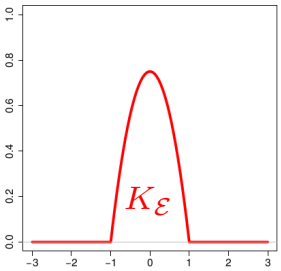
Gaussian: $k_{\mathcal{N}}(t) := \exp(-t/2)$ $\rightsquigarrow c_{k,d} = (2\pi)^{-d/2}$





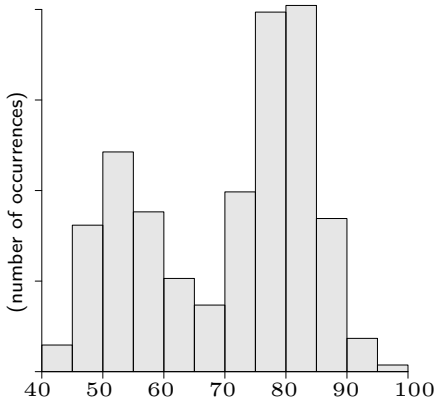
Epanechnikov: $k_{\mathcal{E}}(t) := \begin{cases} 1-t \text{ if } t \leq 1 \\ 0 \text{ if } t > 1 \end{cases}$ $\rightsquigarrow c_{k,d} = \dfrac{d+2}{2\,\mathrm{Vol}\,B_d(0,1)}$





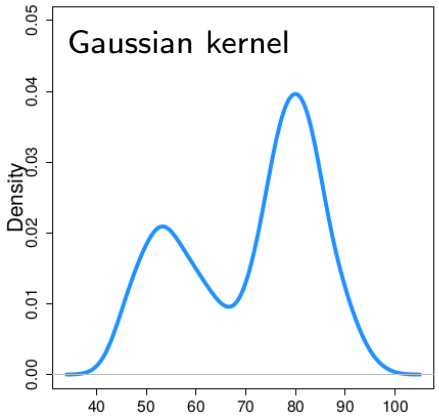
NONPARAMETRIC

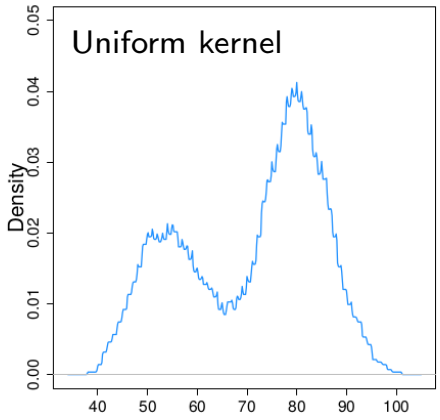
Common kernels

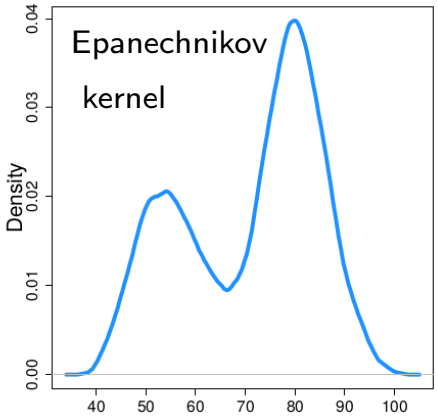


Old faithful geyser dataset (available in R):

- 1st coordinate: waiting time (sec.) between eruptions
- 2nd coordinate (unused): eruptions duration (sec.)

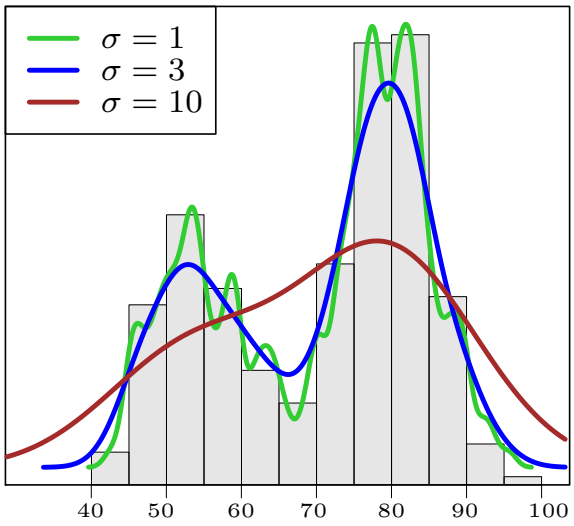






Influence of the bandwidth

- small σ (*undersmoothing*): small bias (sensitivity), large variance (instability)
- large σ (*oversmoothing*): large bias (insensitivity), small variance (stability)



Old geyser dataset

Convergence rates

Radially-symmetric Gaussian kernel in $\mathbb{R}^d$:

$$\hat{f}_n(x) := \frac{1}{(2\pi)^{d/2}\, n\, \sigma^d} \sum_{i=1}^{n} \exp\left(-\frac{\|x - p_i\|_2^2}{2\sigma^2}\right)$$

Bias: $\mathbb{E}_{P_n}\left(\hat{f}_n(x)\right) - f(x) \;=\; O\left(\sigma^2\right)$ (decreases as $\sigma \to 0$)

Variance: $\mathrm{Var}_{P_n}\left(\hat{f}_n(x)\right) \;=\; O\left(\frac{1}{n\,\sigma^d}\right)$ (increases as $\sigma \to 0$)

Mean squared error: $\mathrm{MSE}_{P_n}\left(\hat{f}_n(x)\right) \;=\; O\left(\sigma^4 + \frac{1}{n\,\sigma^d}\right)$

$$\sigma_{\mathrm{OPT}} = n^{-\frac{1}{d+4}} \implies \mathrm{MSE}_{\mathrm{OPT}}\left(\hat{f}_n(x)\right) \;=\; O\left(n^{-\frac{4}{d+4}}\right)$$

Summary

	Method	cvgence rate	parameter(s)	limitation(s)
PARAMETRIC	Gaussian model	$O\left(\sqrt{d/n}\right)$	N/A	bias for non-Gaussians
	GMMs	$O\left(\sqrt{d/n}\right)$	mixture size r	local maxima computation cost
NONPARAMETRIC	Histogram	$O\left(n^{-\frac{1}{d+2}}\right)$	number N of bins	curse of dimensionality computation cost
	Kernel density	$O\left(n^{-\frac{2}{d+4}}\right)$	bandwidth σ	curse of dimensionality

What you should know

- Concepts: density estimator, parametric vs. non-parametric, MSE and bias-variance decomposition, convergence
- Gaussian model: definition, fitting, convergence rates
- Gaussian mixture models: definition, fitting by MLE, convergence rates
- Histograms: definition, bias-variance vs. grid size, convergence rates
- Kernel density estimators: definition, common kernels, bias-variance vs. bandwidth, convergence rates

Supervised Learning and k-NN Prediction

Outline:

- **Foundations of supervised learning**
- **Regression with squared error and k-NN regression**
- **Classification with 0-1 loss and k-NN classification**
- **Advantages and drawbacks of k-NN predictors in practice**
- **Cross-validation**
- **Evaluating a classifier's performance**

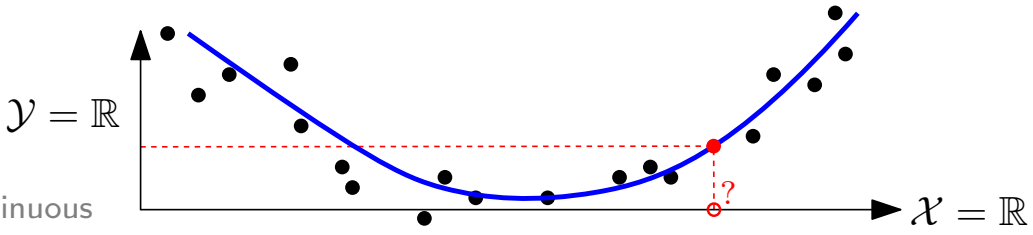
Outline

- **Foundations of supervised learning**
- Regression with squared error and k-NN regression
- Classification with 0-1 loss and k-NN classification
- Advantages and drawbacks of k-NN predictors in practice
- Cross-validation
- Evaluating a classifier's performance

Supervised learning

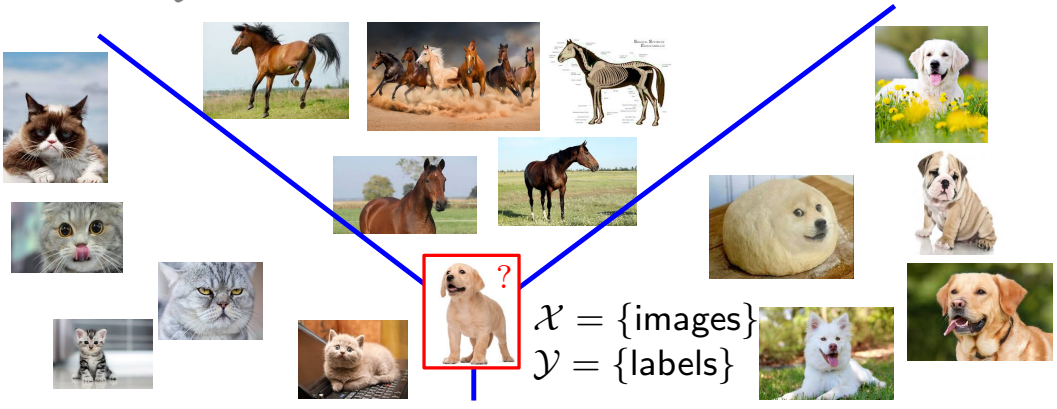
Input: n observations + responses $(x_1, y_1), \cdots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$

Goal: build a predictor $f : \mathcal{X} \to \mathcal{Y}$ from $(x_1, y_1), \cdots, (x_n, y_n)$
whose mean prediction error on new query observations is minimal



regression: $\mathcal{Y}$ continuous

classification: $\mathcal{Y}$ discrete

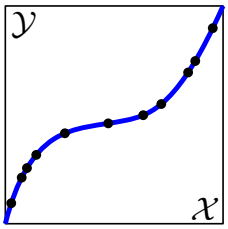


Statistical framework

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

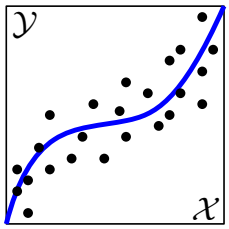
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression) or $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

$\rightarrow$ the **joint distribution** $\Pr(X, Y)$ encodes the complexity of the problem:



X, Y perfectly dependent $\Rightarrow \exists$ perfect predictor

X, Y imperfectly dependent $\Rightarrow \nexists$ perfect predictor



Statistical framework

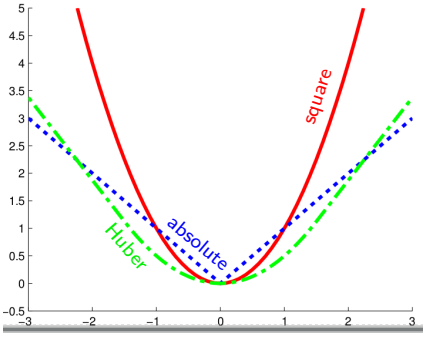
Prediction error is measured by a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

$\rightarrow$ goal: minimize **risk** (expected prediction error): $\mathbb{E}_{(X,Y)}\, L(Y,\, f(X))$

$\rightarrow$ in practice: minimize **empirical risk**: $\dfrac{1}{n} \displaystyle\sum_{i=1}^{n} L(y_i,\, f(x_i))$

Popular loss functions for regression ($\mathcal{Y} = \mathbb{R}$):

$L(y_i, f(x_i))$	Name
$(y_i - f(x_i))^2$	squared error (MSE)
$\lvert y_i - f(x_i)\rvert$	absolute error (MAE)
$\begin{cases} \frac{1}{2}(y_i - f(x_i))^2 \text{ if } \lvert y_i - f(x_i)\rvert < \delta \\ \delta \lvert y_i - f(x_i)\rvert - \frac{1}{2}\delta^2 \text{ otherwise} \end{cases}$	Huber loss



Statistical framework

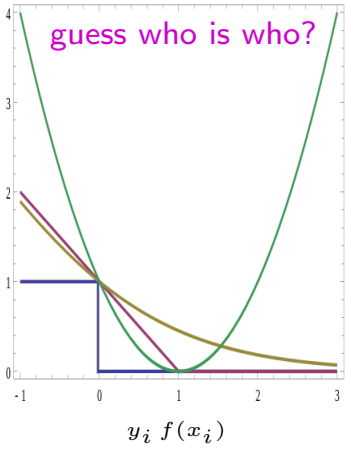
Prediction error is measured by a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

$\to$ goal: minimize **risk** (expected prediction error): $\mathbb{E}_{(X,Y)}\, L(Y,\, f(X))$

$\to$ in practice: minimize **empirical risk**: $\dfrac{1}{n} \sum_{i=1}^{n} L(y_i,\, f(x_i))$

Popular loss functions for binary classification ($\mathcal{Y} = \{-1, 1\}$):

$L(y_i, f(x_i))$	Name	Classifier
$\mathbb{1}_{y_i \neq f(x_i)}$	zero-one	Bayes / k-NN
$(1 - y_i f(x_i))^2$	square	(noname)
$\max\{0, 1 - y_i f(x_i)\}$	hinge	SVM
$\exp(-y_i f(x_i))$	exponential	Boosting
$\log(1 + \exp(-y_i f(x_i)))$	logistic	Logistic reg.



Outline

- Foundations of supervised learning
- **Regression with squared error and k-NN regression**
- Classification with 0-1 loss and k-NN classification
- Advantages and drawbacks of k-NN predictors in practice
- Cross-validation
- Evaluating a classifier's performance

Regression with squared error

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)

Risk: $R(f) = \mathbb{E}_{(X,Y)} \left(Y - f(X)\right)^2$

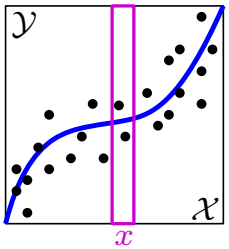
$= \mathbb{E}_X \, \mathbb{E}_{(Y|X)} \left[(Y - f(X))^2 \mid X\right]$ (conditioning on X: $\mathbb{P}(X, Y) = \mathbb{P}(Y \mid X)\, \mathbb{P}(X)$)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} \mathbb{E}_{(Y|X)} \left[(Y - y)^2 \mid X = x\right]$$

$\rightarrow$ minimizer: $f^*(x) = \mathbb{E}_{(Y|X)} \left[Y \mid X = x\right]$ (regression function)

(best prediction of Y at point $X = x$ is the **conditional mean**)



Regression with squared error

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)

Risk: $R(f) = \mathbb{E}_{(X,Y)} \left(Y - f(X)\right)^2$

$= \mathbb{E}_X \, \mathbb{E}_{(Y|X)} \left[(Y - f(X))^2 \mid X\right]$ (conditioning on X: $\mathbb{P}(X,Y) = \mathbb{P}(Y \mid X)\,\mathbb{P}(X)$)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} \mathbb{E}_{(Y|X)} \left[(Y - y)^2 \mid X = x\right]$$

no control over the regularity of f^*

$\rightarrow$ minimizer: $f^*(x) = \mathbb{E}_{(Y|X)} \left[Y \mid X = x\right]$ (regression function)

(best prediction of Y at point $X = x$ is the **conditional mean**)

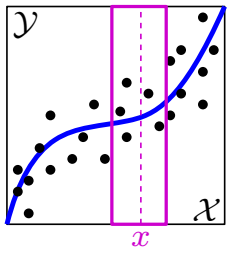
k-NN regression

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)
— unknown probability distributions

$$f^*(x) = \mathbb{E}_{(Y|X)} \left[Y \mid X = x\right]$$

expectation estimated by averaging over samples

conditioning on k-NNs of x ($\mathbb{P}(\exists$ sample at $x) = 0$)



$$\hat{f}_{n,k}(x) := \frac{1}{k} \sum_{x_i \in \mathrm{NN}_k(x)} y_i$$

(regression estimator)

(variant: responses weighted by inverse distances to x)

Thm: (universal consistency) [Stone 1977] [Devroye 1982]
Suppose Y is a bounded random variable. Then, the estimator $\hat{f}_{n,k}$ is consistent if and only if the choice of $k = k(n)$ satisfies $k \to \infty$ and $k/n \to 0$ as $n \to \infty$.

Note: $\hat{f}_{n,k}$ is consistent if: $\forall x \in \mathcal{X}$, $\operatorname*{plim}_{n \to \infty} \hat{f}_{n,k}(x) = f^*(x)$.

Outline

- Foundations of supervised learning
- Regression with squared error and k-NN regression
- **Classification with 0-1 loss and k-NN classification**
- Advantages and drawbacks of k-NN predictors in practice
- Cross-validation
- Evaluating a classifier's performance

Classification with 0-1 loss

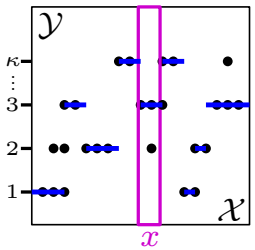
Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

Risk: $R(f) = \mathbb{E}_{(X,Y)} \mathbb{1}_{Y \neq f(X)}$

$= \mathbb{E}_X \mathbb{E}_{(Y|X)} \left[\mathbb{1}_{Y \neq f(X)} \mid X\right]$ (conditioning on X)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}} \mathbb{E}_{(Y|X)} \left[\mathbb{1}_{Y \neq y} \mid X = x\right]$$



$$= \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}} \sum_{r=1}^{\kappa} \mathbb{1}_{r \neq y} \mathbb{P}(Y = r \mid X = x) \quad (Y \text{ categorical variable})$$

$$= \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}} 1 - \mathbb{P}(Y = y \mid X = x) = \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmax}} \mathbb{P}(Y = y \mid X = x)$$

(best prediction at x **maximizes the posterior probability** $\mathbb{P}(Y \mid X)$ (Bayes classifier))

Classification with 0-1 loss

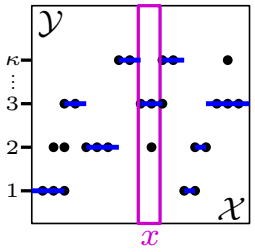
Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

Risk: $R(f) = \mathbb{E}_{(X,Y)}\, \mathbb{1}_{Y \neq f(X)}$

$= \mathbb{E}_X\, \mathbb{E}_{(Y|X)}\, \left[\mathbb{1}_{Y \neq f(X)} \mid X\right]$ (conditioning on X)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}}\ \mathbb{E}_{(Y|X)}\, [\mathbb{1}_{Y \neq y} \mid X = x]$$



$$= \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}} \sum_{r=1}^{\kappa} \mathbb{1}_{r \neq y}\, \mathbb{P}(Y = r \mid X = x) \quad (Y \text{ categorical variable})$$

$$= \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmin}}\ 1 - \mathbb{P}(Y = y \mid X = x) = \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmax}}\ \mathbb{P}(Y = y \mid X = x)$$

$\Rightarrow$ Bayes error rate $R(f^*)$ is zero when X, Y are perfectly dependent

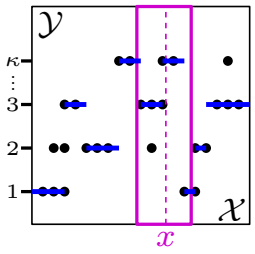
k-NN classification

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$
} unknown probability distributions

$$f^*(x) = \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmax}}\ \mathbb{P}(Y = y \mid X = x)$$

argmax determined by majority vote

conditioning on k-NNs of x ($\mathbb{P}(\exists$ sample at $x) = 0$)



$$\hat{f}_{n,k}(x) := \underset{y \in \{1,\cdots,\kappa\}}{\operatorname{argmax}}\ \#\{i : y_i = y \text{ and } x_i \in \mathrm{NN}_k(x)\}$$

Thm: (1-NN optimality) [Cover, Hart 1967]
For classification into κ classes:

$\Rightarrow$ 1-NN estimator is consistent when X, Y are perfectly dependent or independent

$$0 \leq R(f^*) \leq \lim_{n \to \infty} R(\hat{f}_{n,1}) \leq R(f^*)\left(2 - \frac{\kappa}{\kappa - 1} R(f^*)\right) \leq \frac{\kappa - 1}{\kappa}$$

Outline

- Foundations of supervised learning
- Regression with squared error and k-NN regression
- Classification with 0-1 loss and k-NN classification
- **Advantages and drawbacks of k-NN predictors in practice**
- Cross-validation
- Evaluating a classifier's performance

Advantages and drawbacks of k-NN in practice

- high flexibility:
  - ▶ little prior on the fitting model
  - ▶ method based on distances / (dis-)similarities (no need for coordinates)
- easiness of implementation:
  - ▶ only a few lines of code for NN-search via linear scan
- extends naturally to other problems:
  - ▶ density estimation:

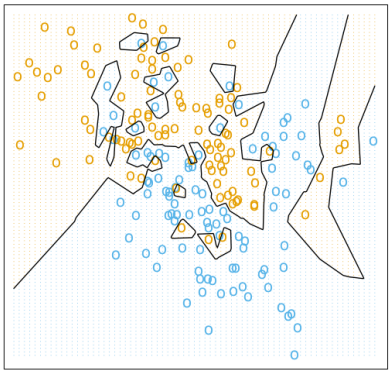
$$\hat{f}_{n,k}(x) := \frac{k}{n} \frac{1}{V_d \, \|x - \mathrm{NN}_k(x)\|_2^d}$$

volume of unit ball: $V_d = \dfrac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$

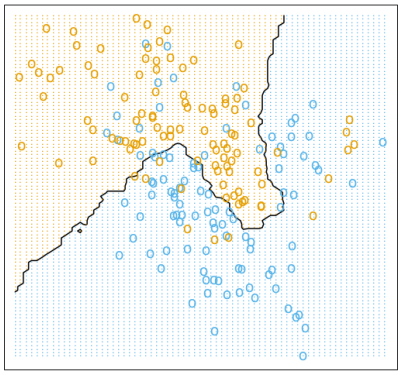
k-th nearest neighbor of x

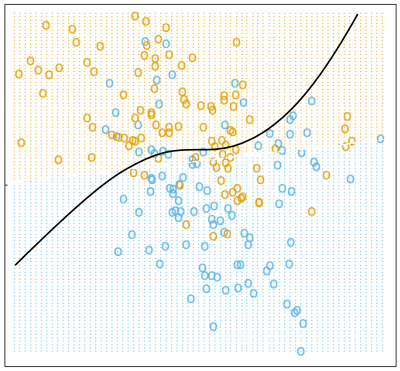
Advantages and drawbacks of k-NN in practice

- algorithmic cost of prediction:
  - linear scan in $\Theta(nd)$
  - sublinear methods become (close to) linear in high dimensions
- slow convergence in high dimensions (**curse of dimensionality**):
  - asymptotic regime often not attained in practice $\rightsquigarrow$ **need to select k**:



small k leads to overfitting





large k leads to underfitting

Outline

- Foundations of supervised learning
- Regression with squared error and k-NN regression
- Classification with 0-1 loss and k-NN classification
- Advantages and drawbacks of k-NN predictors in practice
- **Cross-validation**
- Evaluating a classifier's performance

Cross-validation

A **general method** to select hyperparameters in supervised learning algorithms.

Principle:

- ▶ explore the hyperparameter space or a subset thereof (e.g. via sampling)
- ▶ for each choice of hyperparameter value(s):
  - train the classifier with these values
  - test its performance empirically
- ▶ keep the value(s) that yield the best performance

In practice: 

- ▶ partition initial dataset into two subsets: T (training), and V (validation/test)
- ▶ do the training on T, then test hyperparameter values on V
- ▶ average its performance over some subset of all partitions $T \sqcup V$

Cross-validation

Examples of methods:

Exhaustive cross-validation:

- leave-one-out: 1 observation is reserved for validation $\rightsquigarrow n$ partitions
- leave-p-out: p observations are reserved for validation $\rightsquigarrow \binom{n}{p}$ partitions

Non-exhaustive cross-validation:

- r-fold: randomly partition the observations into r subsets of size n/r
  apply leave-one-out on the r subsets (most commonly $r = 10$)
- holdout: use single random partition $T \sqcup V$ (each pt assigned independently)
- Monte-Carlo: repeatedly use random partitions $T \sqcup V$

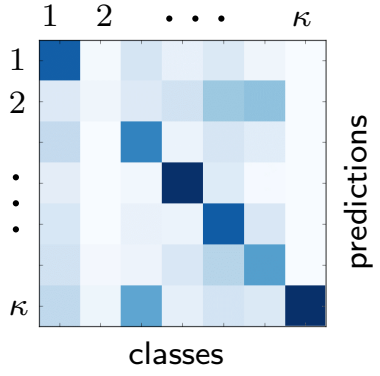
Outline

- Foundations of supervised learning
- Regression with squared error and k-NN regression
- Classification with 0-1 loss and k-NN classification
- Advantages and drawbacks of k-NN predictors in practice
- Cross-validation
- **Evaluating a classifier's performance**

Evaluating a classifier's performance

Given a test set $V = \{x'_1, \cdots, x'_m\}$ and known responses $\{y'_1, \cdots, y'_m\}$:

- **error rate**: $\tau_{\text{err}} := \frac{1}{m} \#\{\text{misclassified points}\}$
- **accuracy**: $\tau_{\text{acc}} := 1 - \tau_{\text{err}} = \frac{1}{m} \#\{\text{correctly classified points}\}$
  - ▶ biased when classes have significantly different sizes (e.g. $1/10^6$ for sick vs. healthy)
- **confusion matrix**: $C_{i,j} := \frac{1}{m} \#\{\text{points of class } j \text{ predicted as being in } i\}$



Evaluating a classifier's performance

Given a test set $V = \{x'_1, \cdots, x'_m\}$ and known responses $\{y'_1, \cdots, y'_m\}$:

- **error rate**: $\tau_{\text{err}} := \frac{1}{m} \#\{\text{misclassified points}\}$
- **accuracy**: $\tau_{\text{acc}} := 1 - \tau_{\text{err}} = \frac{1}{m} \#\{\text{correctly classified points}\}$
  - ▶ biased when classes have significantly different sizes (e.g. $1/10^6$ for sick vs. healthy)
- **confusion matrix**: $C_{i,j} := \frac{1}{m} \#\{\text{points of class } j \text{ predicted as being in } i\}$
  - ▶ true positives (TP) for class i: points of this class correctly predicted in i
  - ▶ false positives (FP) for class i: points of other classes incorrectly predicted in i
  - ▶ true negatives (TN) for class i: points of $j \neq i$ predicted in $l \neq i$ (possibly with $l \neq j$)
  - ▶ false negatives (FN) for class i: points of i predicted in some $j \neq i$

Evaluating a classifier's performance

Given a test set $V = \{x'_1, \cdots, x'_m\}$ and known responses $\{y'_1, \cdots, y'_m\}$:

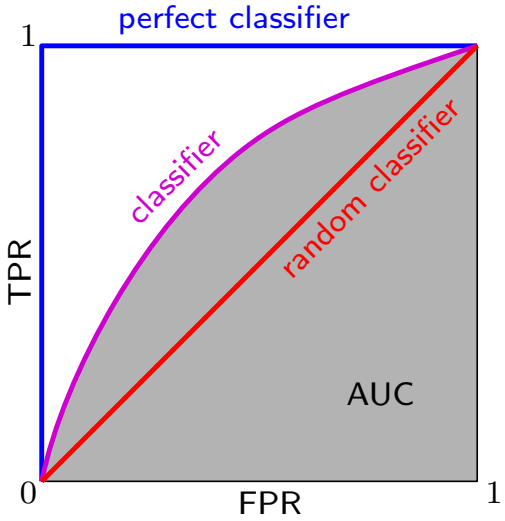
- **error rate**: $\tau_{\text{err}} := \frac{1}{m} \#\{\text{misclassified points}\}$
- **accuracy**: $\tau_{\text{acc}} := 1 - \tau_{\text{err}} = \frac{1}{m} \#\{\text{correctly classified points}\}$
  - ▶ biased when classes have significantly different sizes (e.g. $1/10^6$ for sick vs. healthy)
- **confusion matrix**: $C_{i,j} := \frac{1}{m} \#\{\text{points of class } j \text{ predicted as being in } i\}$
  - ▶ **precision / positive predicted value**: $\text{PPV} := \dfrac{\text{TP}}{\text{TP} + \text{FP}}$
  - ▶ **recall / sensitivity / true positive rate**: $\text{TPR} := \dfrac{\text{TP}}{\text{TP} + \text{FN}}$
  - ▶ **fall-out / false positive rate**: $\text{FPR} := \dfrac{\text{FP}}{\text{TN} + \text{FP}}$

for binary classification or one-vs.-all

Evaluating a classifier's performance

Given a test set $V = \{x'_1, \cdots, x'_m\}$ and known responses $\{y'_1, \cdots, y'_m\}$:

- **F-score**: $\mathrm{FS} := \dfrac{2}{\frac{1}{\mathrm{PPV}} + \frac{1}{\mathrm{TPR}}} = \dfrac{2 \cdot \mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}}$ (harmonic mean of prec. & rec.)
  - biased towards positives
- **receiver operating characteristic (ROC) curve**:
  - plots recall (TPR) versus fall-out (FPR)
  - perfect classifier has $\mathrm{TPR} = 1$ and $\mathrm{FPR} = 0$
  - random classifier has $\mathrm{TPR} = \mathrm{FPR}$
  - **AUC**: area under the ROC curve



What you should know

- Concepts: classification, regression, loss function, risk, empirical risk
- Regression function, k-NN regressor and its consistency
- Bayes classifier, k-NN classifier and its optimality
- Advantages and drawbacks of k-NN predictors
- Cross-validation: principle, main methods (exhaustive, non-exhaustive)
- Evaluation criteria: accuracy, confusion matrix, precision/recall/fall-out, F-score, ROC curve

Linear Models for Regression

Outline:

- **Reminder about supervised regression**
- **Linear model for regression and Ordinary Least Squares estimator**
- **Optimality and practical evaluation**
- **Degenerate settings and regularization**
- **Parametric non-linear regression using basis functions**
- **Non-parametric non-linear regression using kernels**

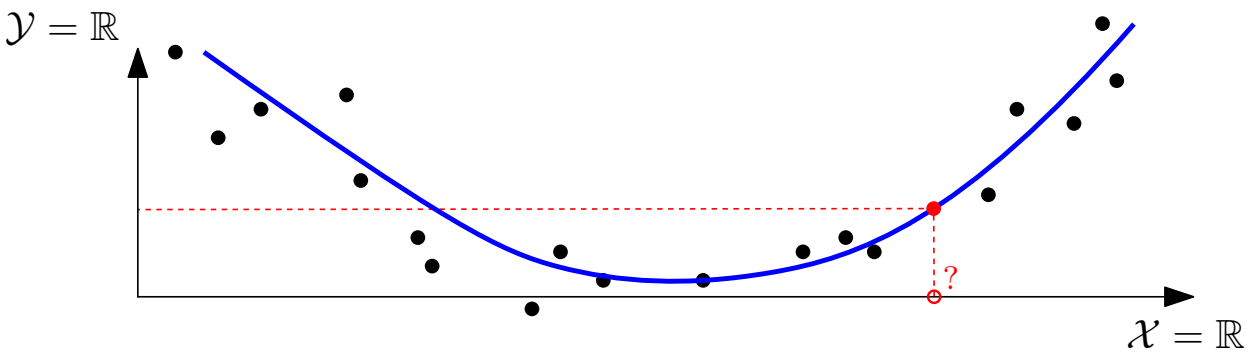
Outline

- **Reminder about supervised regression**
- Linear model for regression and Ordinary Least Squares estimator
- Optimality and practical evaluation
- Degenerate settings and regularization
- Parametric non-linear regression using basis functions
- Non-parametric non-linear regression using kernels

Supervised learning (regression)

Input: n observations + responses $(x_1, y_1), \cdots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$

Goal: build a predictor $f : \mathcal{X} \to \mathcal{Y}$ from $(x_1, y_1), \cdots, (x_n, y_n)$
whose mean prediction error on new query observations is minimal



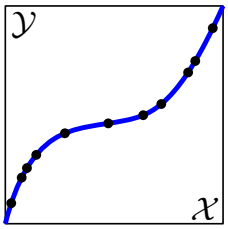
regression: $\mathcal{Y}$ continuous

Statistical framework

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

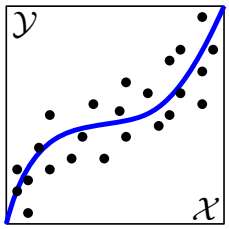
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)

$\rightarrow$ the **joint distribution** $\Pr(X, Y)$ encodes the complexity of the problem



X, Y perfectly dependent $\Rightarrow \exists$ perfect predictor

X, Y imperfectly dependent $\Rightarrow \nexists$ perfect predictor



Statistical framework

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

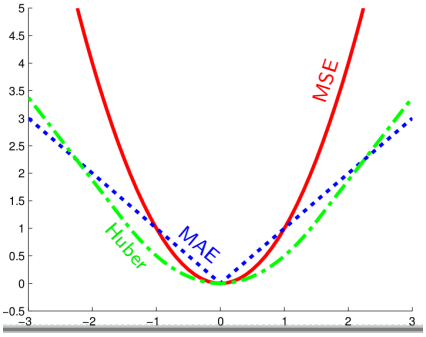
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)

$\rightarrow$ the **joint distribution** $\Pr(X, Y)$ encodes the complexity of the problem

Prediction error is measured by a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

$\rightarrow$ goal: minimize **risk** (expected prediction error): $\mathbb{E}_{(X,Y)}\, L(Y,\, f(X))$

$\rightarrow$ in practice: minimize **empirical risk**: $\dfrac{1}{n} \sum_{i=1}^{n} L(y_i,\, f(x_i))$



Regression with squared error

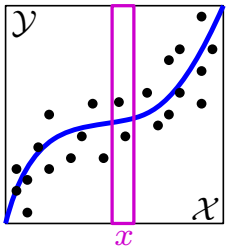
Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \mathbb{R}$ (regression)

Risk: $R(f) = \mathbb{E}_{(X,Y)} \left(Y - f(X)\right)^2$

$= \mathbb{E}_X \, \mathbb{E}_{(Y|X)} \left[(Y - f(X))^2 \mid X\right]$ (conditioning on X: $\mathbb{P}(X,Y) = \mathbb{P}(Y \mid X)\,\mathbb{P}(X)$)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} \mathbb{E}_{(Y|X)} \left[(Y - y)^2 \mid X = x\right]$$



$\rightarrow$ minimizer: $f^*(x) = \mathbb{E}_{(Y|X)} \left[Y \mid X = x\right]$ (regression function)

(best prediction of Y at point $X = x$ is the **conditional mean**)

Advantages and drawbacks of k-NN in practice

- high flexibility:
  - ▶ little prior on the fitting model ← **prior:** linear model
  - ▶ method based on distances / (dis-)similarities (no need for coordinates) ← requires linear space hence coordinates
- easiness of implementation:
  - ▶ only a few lines of code for NN-search via linear scan ← easy and efficient prediction (dot-product)
- algorithmic cost of prediction: ← algorithmic cost put on training
  - ▶ linear scan in $\Theta(nd)$
  - ▶ sublinear methods become (close to) linear in high dimensions
- slow convergence in high dimensions (**curse of dimensionality**): ← no hyper-parameter
  - ▶ asymptotic regime often not attained in practice $\rightsquigarrow$ **need to select** k

Outline

- Reminder about supervised regression
- **Linear model for regression and Ordinary Least Squares estimator**
- Optimality and practical evaluation
- Degenerate settings and regularization
- Parametric non-linear regression using basis functions
- Non-parametric non-linear regression using kernels

Linear model for regression

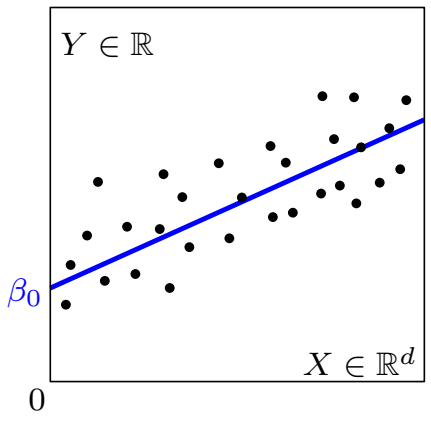
Hyp: Y depends linearly on X plus some independent noise ε:

$$Y = \beta_0 + \sum_{j=1}^{d} X_j \, \beta_j + \varepsilon$$

intercept

independent, centered, real random variable (noise)

$(\mathbb{E}[\varepsilon] = 0)$

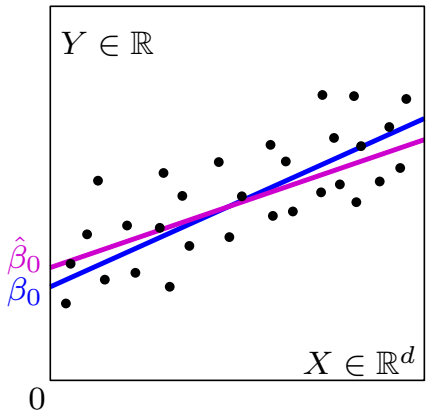


Linear model for regression

Hyp: Y depends linearly on X plus some independent noise ε:

$$Y = \beta_0 + \sum_{j=1}^{d} X_j \, \beta_j + \varepsilon = [\, 1 \;\; X^T \,] \; \beta + \varepsilon \text{ where } \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

parameter to be estimated from the data: $\hat{\beta}$



Linear predictor: $f_{\hat{\beta}}(x) := [\, 1 \;\; x^T \,] \; \hat{\beta}$

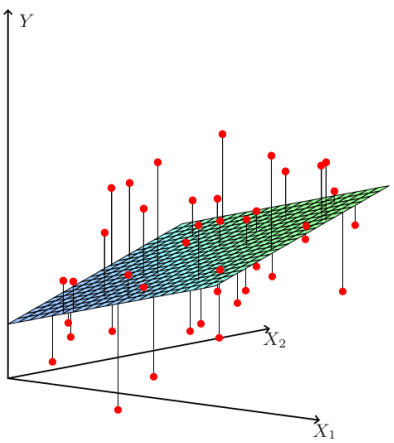
Ordinary least squares (OLS) estimator

Input: $(x_1, y_1), \cdots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$\longrightarrow$ estimate β by minimizing the empirical risk with MSE:

$$\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\beta(x_i))^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - [\, 1 \;\; x_i^T \,] \; \beta)^2$$

residual sum of squares (RSS)



Ordinary least squares (OLS) estimator

Input: $(x_1, y_1), \cdots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$\longrightarrow$ estimate β by minimizing the empirical risk with MSE:

$$\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \underbrace{\sum_{i=1}^{n} (y_i - f_\beta(x_i))^2}_{\text{residual sum of squares (RSS)}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - [\,1 \;\; x_i^T\,]\;\beta)^2$$

$$\mathrm{RSS}(\beta) = \|y - \mathbf{X}\,\beta\|_2^2 \quad \text{where} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^T \\ & \vdots \\ 1 & x_n^T \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

quadratic function of β — response vector — coordinates matrix with 1's appended

Ordinary least squares (OLS) estimator

Input: $(x_1, y_1), \cdots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$

$\longrightarrow$ estimate β by minimizing the empirical risk with MSE:

$$\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \underbrace{\sum_{i=1}^{n} (y_i - f_\beta(x_i))^2}_{\text{residual sum of squares (RSS)}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - [\,1 \;\; x_i^T\,]\;\beta)^2$$

$$\mathrm{RSS}(\beta) = \|y - \mathbf{X}\,\beta\|_2^2 \quad \text{where} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^T \\ & \vdots \\ 1 & x_n^T \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

$$\begin{cases} \nabla\,\mathrm{RSS}(\beta) = -2\,\mathbf{X}^T\,(y - \mathbf{X}\,\beta) \longleftarrow \text{minimizers satisfy } \mathbf{X}^T\,(\mathbf{y} - \mathbf{X}\,\beta) = 0 \\ \nabla^2\,\mathrm{RSS}(\beta) = 2\,\mathbf{X}^T\mathbf{X} \longleftarrow \text{positive semi-definite} \Rightarrow \text{convex functional} \end{cases}$$

Ordinary least squares (OLS) estimator

Nondegeneracy assumption: matrix $\mathbf{X}$ has full column rank

$\Rightarrow 2\,\mathbf{X}^T\mathbf{X}$ is positive definite

$\Rightarrow \hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T y$ is the unique minimizer

- predictor: $f_{\hat{\beta}}(x) = \begin{bmatrix}1 & x^T\end{bmatrix}\hat{\beta} = \begin{bmatrix}1 & x^T\end{bmatrix}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T y$

$$\mathrm{RSS}(\beta) = \|y - \mathbf{X}\,\beta\|_2^2 \quad \text{where} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^T \\ & \vdots \\ 1 & x_n^T \end{bmatrix} \in \mathbb{R}^{n\times(d+1)}$$

$$\begin{cases} \nabla\,\mathrm{RSS}(\beta) = -2\,\mathbf{X}^T(y - \mathbf{X}\,\beta) \longleftarrow \text{minimizers satisfy } \mathbf{X}^T(\mathbf{y} - \mathbf{X}\,\beta) = 0 \\ \nabla^2\,\mathrm{RSS}(\beta) = 2\,\mathbf{X}^T\mathbf{X} \longleftarrow \text{positive semi-definite} \Rightarrow \text{convex functional} \end{cases}$$

Ordinary least squares (OLS) estimator

Nondegeneracy assumption: matrix $\mathbf{X}$ has full column rank

$\Rightarrow 2\,\mathbf{X}^T\mathbf{X}$ is positive definite

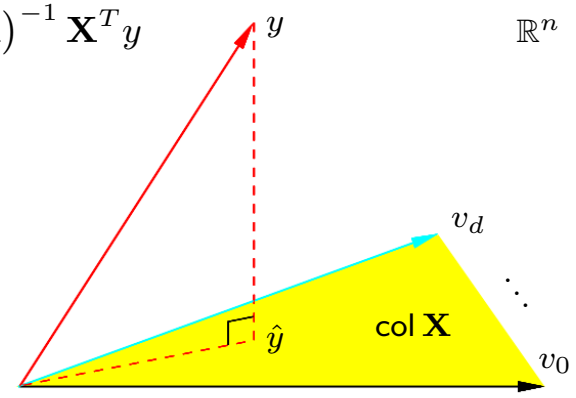
$\Rightarrow \hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T y$ is the unique minimizer

- predictor: $f_{\hat{\beta}}(x) = \begin{bmatrix}1 & x^T\end{bmatrix}\hat{\beta} = \begin{bmatrix}1 & x^T\end{bmatrix}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T y$

- fitted values:

$$\forall i, \quad \hat{y}_i := f_{\hat{\beta}}(x_i) = \begin{bmatrix}1 & x_i^T\end{bmatrix}\hat{\beta}$$

$$\hat{y} := \mathbf{X}\,\hat{\beta} = \underbrace{\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T}\, y$$



(ortho. proj. onto the linear subspace $\langle v_0, \cdots, v_d\rangle \subseteq \mathbb{R}^n$ spanned by the input variables and $v_0 \equiv 1$)

Outline

- Reminder about supervised regression
- Linear model for regression and Ordinary Least Squares estimator
- **Optimality and practical evaluation**
- Degenerate settings and regularization
- Parametric non-linear regression using basis functions
- Non-parametric non-linear regression using kernels

Optimality

Asumptions:

$(x_1, y_1), \cdots, (x_n, y_n) \overset{\text{iid}}{\sim} (X, Y)$ taking values in $\mathbb{R}^d \times \mathbb{R}$ (regression)

$Y = [\, 1 \;\; X^T \,]\, \beta + \varepsilon$ (linear model)

$\mathbb{E}[\varepsilon] = 0$ and $\mathrm{Var}(\varepsilon) < +\infty$ (centered noise with finite variance)

Then: (Gauss-Markov Theorem)

the OLS estimator $\hat{\beta}$ is unbiased: $\mathbb{E}_{(\mathbf{X},y)}[\hat{\beta}] = \beta$

the OLS estimator minimises the **MSE** among all linear estimators:

$$\hat{\beta} \in \underset{\tilde{\beta}}{\operatorname{argmin}}\ \mathbb{E}_{(\mathbf{X},y)}\, \|\tilde{\beta} - \beta\|_2^2$$

$\Rightarrow \hat{\beta}$ is a best linear unbiased estimator (**BLUE**):

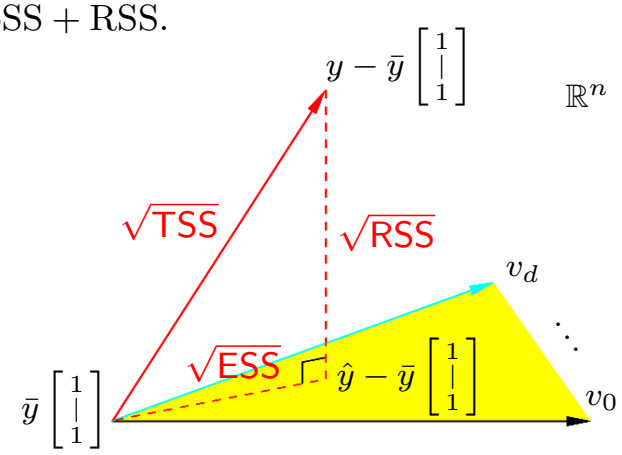
$\mathrm{Var}(\tilde{\beta}) - \mathrm{Var}(\hat{\beta})$ is positive semi-definite for any unbiased linear estimator $\tilde{\beta}$

Evaluation in practice

Let $\bar{y} := \frac{1}{n}\sum_{i=1}^n y_i$ be the empirical mean response, and $\hat{y} := \mathbf{X}\hat{\beta}$ the predictions

- **total sum of squares**: $\mathrm{TSS} := \sum_{i=1}^n (y_i - \bar{y})^2 = \left\| y - \bar{y} \begin{bmatrix} 1 \\ | \\ 1 \end{bmatrix} \right\|_2^2$
- **explained sum of squares**: $\mathrm{ESS} := \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 = \left\| \hat{y} - \bar{y} \begin{bmatrix} 1 \\ | \\ 1 \end{bmatrix} \right\|_2^2$
- **residual sum of squares**: $\mathrm{RSS} := \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \|\hat{y} - y\|_2^2$

Prop: $\mathrm{TSS} = \mathrm{ESS} + \mathrm{RSS}$.



Evaluation in practice

Let $\bar{y} := \frac{1}{n}\sum_{i=1}^n y_i$ be the empirical mean response, and $\hat{y} := \mathbf{X}\hat{\beta}$ the predictions

- **total sum of squares**: $\mathrm{TSS} := \sum_{i=1}^n (y_i - \bar{y})^2 = \left\| y - \bar{y} \begin{bmatrix} 1 \\ | \\ 1 \end{bmatrix} \right\|_2^2$
- **explained sum of squares**: $\mathrm{ESS} := \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 = \left\| \hat{y} - \bar{y} \begin{bmatrix} 1 \\ | \\ 1 \end{bmatrix} \right\|_2^2$
- **residual sum of squares**: $\mathrm{RSS} := \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \|\hat{y} - y\|_2^2$

Prop: $\mathrm{TSS} = \mathrm{ESS} + \mathrm{RSS}$.

- **fraction of variance unexplained:** $\mathrm{FVU} := \frac{\mathrm{RSS}}{\mathrm{TSS}} \in [0, 1]$
- **coefficient of determination:** $\mathrm{R}^2 := \frac{\mathrm{ESS}}{\mathrm{TSS}} = 1 - \mathrm{FVU} \in [0, 1]$

> R close to $1 \Rightarrow$ good fit
>
> R close to $0 \Rightarrow$ bad fit

Outline

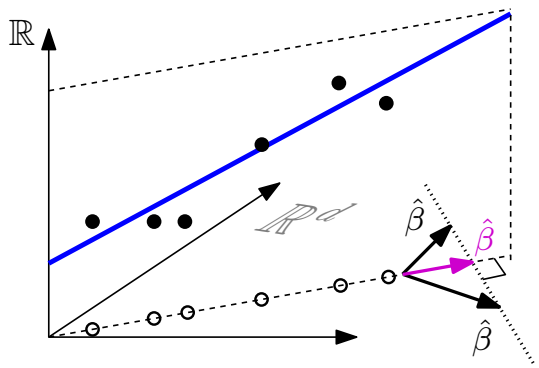
- Reminder about supervised regression
- Linear model for regression and Ordinary Least Squares estimator
- Optimality and practical evaluation
- **Degenerate settings and regularization**
- Parametric non-linear regression using basis functions
- Non-parametric non-linear regression using kernels

Degenerate settings

Q: what if the coordinates matrix $\mathbf{X}$ does not have full column rank?

(happens typically with perfectly correlated variables or when $n < d$)

- an affine subspace of minimizers $\hat{\beta}$
- one choice of $\hat{\beta}$ is more natural: the one within the subspace $\langle x_1, \cdots, x_n \rangle$ (a.k.a. the one with smallest norm)



Degenerate settings

Q: what if the coordinates matrix $\mathbf{X}$ does not have full column rank?

(happens typically with perfectly correlated variables or when $n < d$)

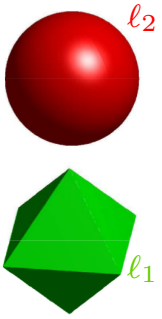
- an affine subspace of minimizers $\hat{\beta}$
- one choice of $\hat{\beta}$ is more natural: the one within the subspace $\langle x_1, \cdots, x_n \rangle$ (a.k.a. the one with smallest norm)
- solution 1: dimensionality reduction: (cf. Lecture 10)
  - preprocessing: estimate the subspace $\langle x_1, \cdots, x_n \rangle$
  - main: solve optimization problem within the estimated $\langle x_1, \cdots, x_n \rangle$

Degenerate settings

Q: what if the coordinates matrix $\mathbf{X}$ does not have full column rank?

(happens typically with perfectly correlated variables or when $n < d$)

- an affine subspace of minimizers $\hat{\beta}$
- one choice of $\hat{\beta}$ is more natural: the one within the subspace $\langle x_1, \cdots, x_n \rangle$ (a.k.a. the one with smallest norm)
- solution 2: regularized linear regression:
  - ridge: $\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - [\, 1 \;\; x_i^T \,]\, \beta)^2 \;+\; \lambda \|\beta\|_2^2$
  - lasso: $\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - [\, 1 \;\; x_i^T \,]\, \beta)^2 \;+\; \lambda \|\beta\|_1$
  - elastic net: $\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - [\, 1 \;\; x_i^T \,]\, \beta)^2 \;+\; \lambda \left(\alpha \|\beta\|_2^2 + (1-\alpha) \|\beta\|_1\right)$



Ridge regression

$$\hat{\beta} := \underset{\beta \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - [\,1 \;\; x_i^T\,]\;\beta)^2 \;+\; \lambda \|\beta\|_2^2$$

$$= \operatorname{argmin}_{\beta \in \mathbb{R}^{d+1}} \|y - \mathbf{X}\,\beta\|_2^2 \;+\; \lambda \|\beta\|_2^2$$

- $$\begin{cases} \nabla \cdot (\beta) = -2\,\mathbf{X}^T (y - \mathbf{X}\,\beta) + 2\lambda\,\beta \\ \nabla^2 \cdot (\beta) = 2\,\mathbf{X}^T\mathbf{X} + 2\lambda\,\mathbf{I}_{d+1} \end{cases}$$ ← positive definite for any $\lambda > 0$ ⇒ strictly convex functional
- $$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I}_{d+1}\right)^{-1} \mathbf{X}^T y$$
- algorithms: LU decomposition, Cholesky decomposition
  — $O(n^3)$ by Gaussian elimination, $O(n^\omega)$ by divide-and-conquer
- in practice: use library for linear algebra (e.g. LAPACK, Eigen)

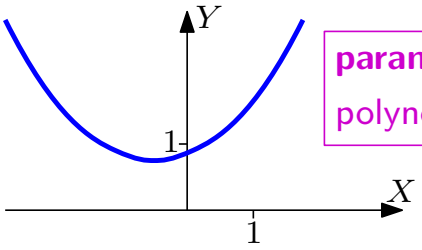
Outline

- Reminder about supervised regression
- Linear model for regression and Ordinary Least Squares estimator
- Optimality and practical evaluation
- Degenerate settings and regularization
- **Parametric non-linear regression using basis functions**
- Non-parametric non-linear regression using kernels

Non-linear regression using basis functions

Q: what if X, Y are dependent through some non-linear function?

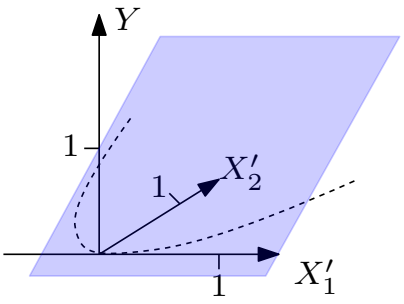
Example: $Y = X^2 + X + 1$



parametric families:
polynomials, RBFs, etc.

▶ transform initial variables:

$$\left.\begin{array}{l} X'_1 := X \\ X'_2 := X^2 \end{array}\right\} Y = X'_2 + X'_1 + 1$$



▶ solve linear regression with transformed variables:

$$f_{\hat{\beta}}(x) := [\,1 \;\; x'^T\,]\;\hat{\beta} \quad \text{where} \quad \left\{ \begin{array}{l} x' = (x, x^2) \in \mathbb{R}^2 \\ \hat{\beta} := \text{argmin}_{\beta \in \mathbb{R}^3} \; \|y - \mathbf{X}'\,\beta\|_2^2 \; + \; \lambda\,\|\beta\|_2^2 \end{array} \right.$$

Outline

- Reminder about supervised regression
- Linear model for regression and Ordinary Least Squares estimator
- Optimality and practical evaluation
- Degenerate settings and regularization
- Parametric non-linear regression using basis functions
- **Non-parametric non-linear regression using kernels**

Non-linear regression using kernels

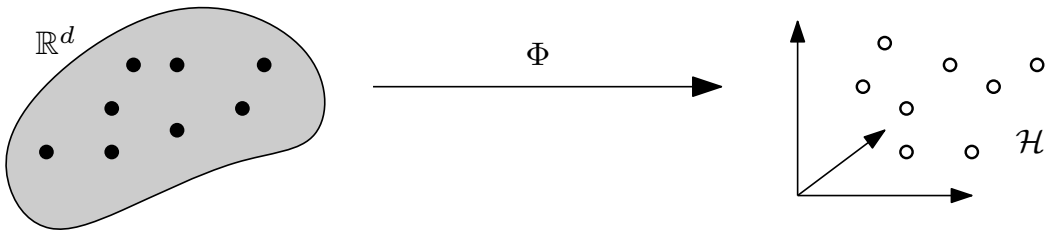
A Hilbert function space $\mathcal{H} \subset \mathbb{R}^{\mathbb{R}^d}$ is a **reproducing kernel Hilbert space** (**RKHS**) on $\mathbb{R}^d$ if $\exists \Phi : \mathbb{R}^d \to \mathcal{H}$ s.t.:

$$\forall x \in \mathbb{R}^d,\ \forall f \in \mathcal{H},\ f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}}$$

reproducing property

Terminology:

- **feature space** $\mathcal{H}$, **feature map** Φ
- **feature vectors** $\Phi(x)$
- **kernel** $k := \langle \Phi(\cdot),\ \Phi(\cdot) \rangle_{\mathcal{H}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$



Non-linear regression using kernels

A Hilbert function space $\mathcal{H} \subset \mathbb{R}^{\mathbb{R}^d}$ is a **reproducing kernel Hilbert space (RKHS)** on $\mathbb{R}^d$ if $\exists \Phi : \mathbb{R}^d \to \mathcal{H}$ s.t.:

$$\forall x \in \mathbb{R}^d,\ \forall f \in \mathcal{H},\ f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}}$$

Prop: The kernel of any RKHS on $\mathbb{R}^d$ is unique.
Conversely, k is the kernel of at most one RKHS on $\mathbb{R}^d$.

- $\Phi(x) = k(x, \cdot)$

Non-linear regression using kernels

A Hilbert function space $\mathcal{H} \subset \mathbb{R}^{\mathbb{R}^d}$ is a **reproducing kernel Hilbert space (RKHS)** on $\mathbb{R}^d$ if $\exists \Phi : \mathbb{R}^d \to \mathcal{H}$ s.t.:

$$\forall x \in \mathbb{R}^d,\ \forall f \in \mathcal{H},\ f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}}$$

Prop: The kernel of any RKHS on $\mathbb{R}^d$ is unique.
Conversely, k is the kernel of at most one RKHS on $\mathbb{R}^d$.

Thm: [Moore 1950] $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel iff it is *positive (semi-)definite*, i.e. $\forall n \in \mathbb{N},\ \forall x_1, \cdots, x_n \in \mathbb{R}^d$, the Gram matrix $(k(x_i, x_j))_{i,j}$ is positive semi-definite.

Examples:

- linear: $k(x, y) = \langle x, y \rangle$ $\quad \mathcal{H} = (\mathbb{R}^d)^*,\ \Phi(x) = \langle x, \cdot \rangle$
- polynomial: $k(x, y) = (1 + \langle x, y \rangle)^N = \sum_{n_1 + \cdots + n_d = N} \binom{N}{n_1, \cdots, n_d} \underbrace{x_1^{n_1} \cdots x_d^{n_d}}_{\propto \text{ coord. of } \Phi(x)} y_1^{n_1} \cdots y_d^{n_d}$
- Gaussian: $k(x, y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right),\ \sigma > 0.$ $\quad \mathcal{H} \subset L_2(\mathbb{R}^d)$

Non-linear regression using kernels

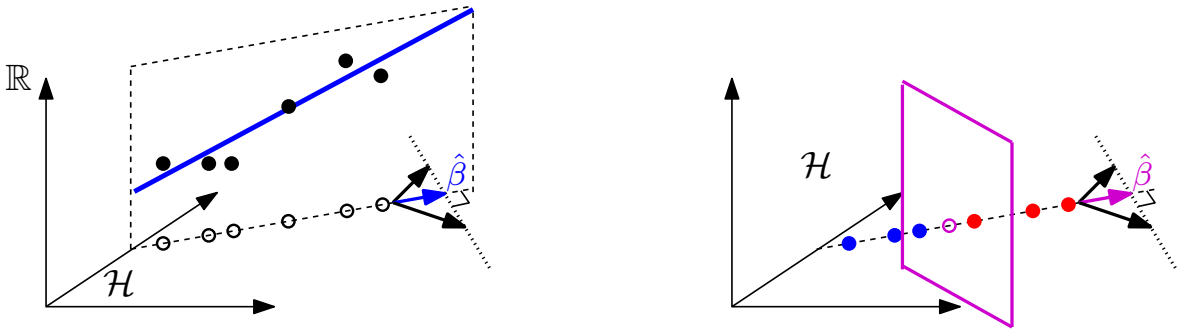
A Hilbert function space $\mathcal{H} \subset \mathbb{R}^{\mathbb{R}^d}$ is a **reproducing kernel Hilbert space** (**RKHS**) on $\mathbb{R}^d$ if $\exists \Phi : \mathbb{R}^d \to \mathcal{H}$ s.t.:

$$\forall x \in \mathbb{R}^d,\ \forall f \in \mathcal{H},\ f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}}$$

Thm: (Representer) [Kimeldorf, Wahba 1971] [Schölkopf et al 2001]
Given RKHS $\mathcal{H}$ with kernel k, any function $f^* \in \mathcal{H}$ minimizing

$$\tfrac{1}{n} \textstyle\sum_{i=1}^n L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

is of the form $f^*(\cdot) = \sum_{j=1}^n \alpha_j k(x_j, \cdot)$, where $\alpha_1, \cdots, \alpha_n \in \mathbb{R}$.



Non-linear regression using kernels

A Hilbert function space $\mathcal{H} \subset \mathbb{R}^{\mathbb{R}^d}$ is a **reproducing kernel Hilbert space** (**RKHS**) on $\mathbb{R}^d$ if $\exists \Phi : \mathbb{R}^d \to \mathcal{H}$ s.t.:

$$\forall x \in \mathbb{R}^d,\ \forall f \in \mathcal{H},\ f(x) = \langle f, \Phi(x) \rangle_{\mathcal{H}}$$

Thm: (Representer) [Kimeldorf, Wahba 1971] [Schölkopf et al 2001]
Given RKHS $\mathcal{H}$ with kernel k, any function $f^* \in \mathcal{H}$ minimizing

$$\tfrac{1}{n} \textstyle\sum_{i=1}^n L(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2$$

is of the form $f^*(\cdot) = \sum_{j=1}^n \alpha_j k(x_j, \cdot)$, where $\alpha_1, \cdots, \alpha_n \in \mathbb{R}$.

- $\operatorname{argmin}_{\alpha} \dfrac{1}{n} \displaystyle\sum_{i=1}^n L\left(y_i, \sum_{j=1}^n \alpha_j k(x_j, x_i)\right) + \lambda \sum_{i,j=1}^n \alpha_i \alpha_j k(x_i, x_j)$

where $\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$

only the $k(x_i, x_j)$ are required to minimize (**kernel trick**)

Non-linear regression using kernels

Case of regression with squared error:

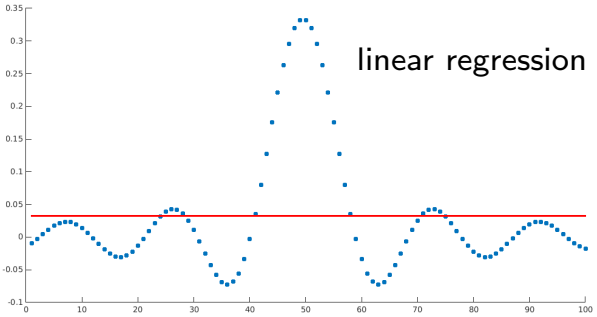
$$\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \alpha_j k(x_j, x_i) \right)^2 + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)$$

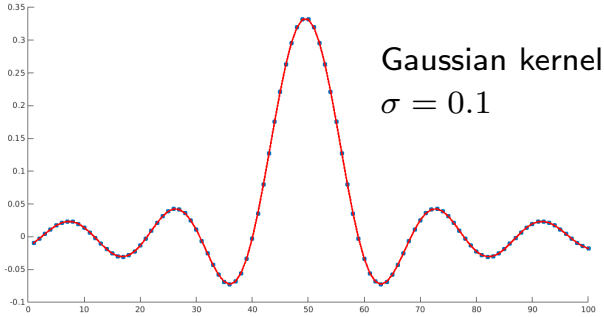
$$= \underset{\alpha}{\operatorname{argmin}} \; \|y - \mathbf{K}\alpha\|^2 + \lambda \, \alpha^T \mathbf{K} \alpha \qquad \text{where } \mathbf{K}_{ij} := k(x_i, x_j)$$

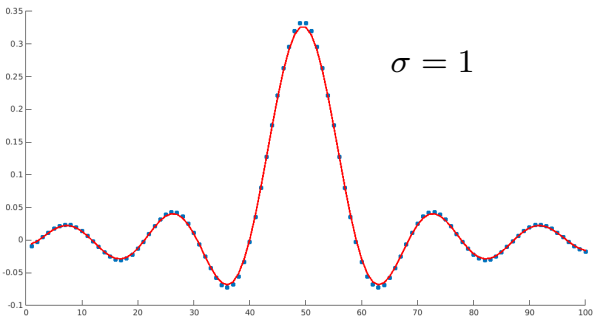
- $\hat{\alpha} = (\mathbf{K} + \lambda \, \mathbf{I}_n)^{-1} \, y$
- $\hat{f}(x) = \sum_{j=1}^{n} \hat{\alpha}_j \, k(x_j, x)$

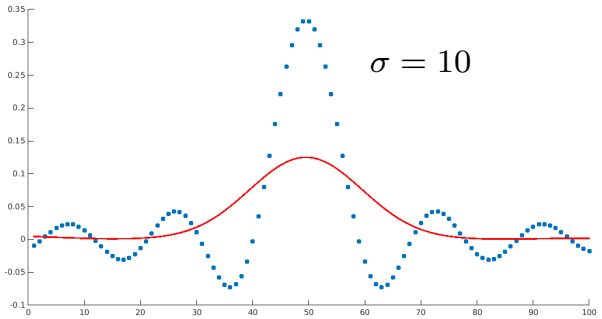
Non-linear regression using kernels

Experimentation: $y = \dfrac{sin((x - 49.5)/3)}{x - 49.5}$



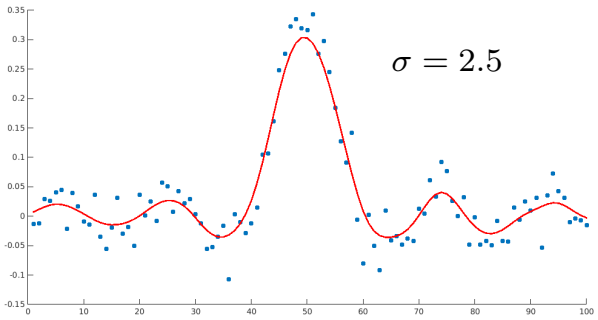


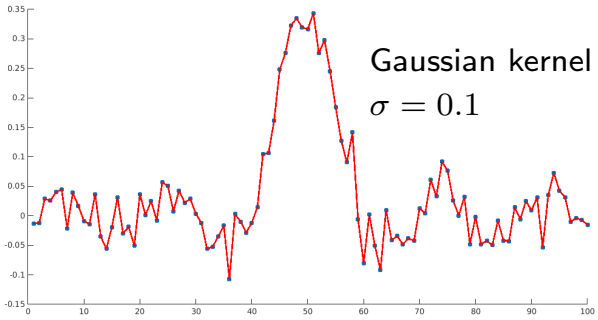


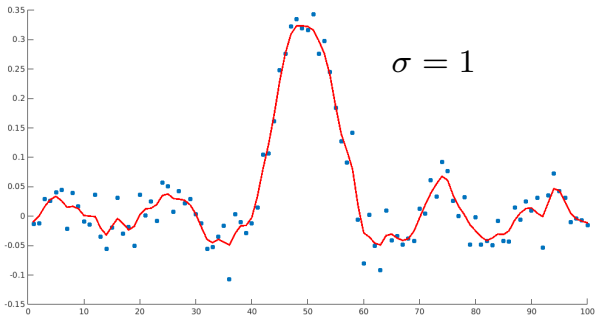


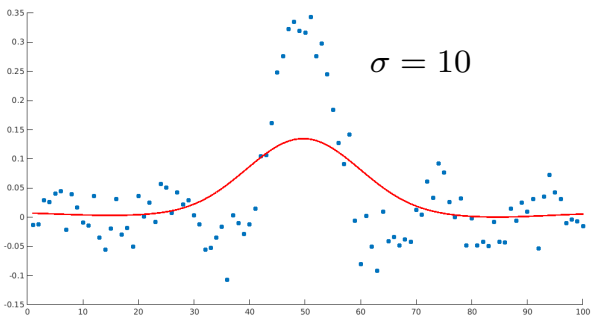
Non-linear regression using kernels

Experimentation: $y = \dfrac{sin((x - 49.5)/3)}{x - 49.5} + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 0.1)$









What you should know

- Linear regression model and OLS estimator
- Gauss-Markov theorem, $\mathrm{TSS} = \mathrm{ESS} + \mathrm{RSS}$, FVU, R^2
- Ridge regression, existence of Lasso regression
- Principles of non-linear regression:
  - basis functions
  - kernels: definition, Moore and Representer theorems (kernel trick)

Linear Models for Classification

Outline:

- **Reminder about supervised classification**
- **Principles of linear methods for classification**
- **Textbook case: linear regression for classification**
- **Logistic regression:**
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- **Non-linear classification using kernels**

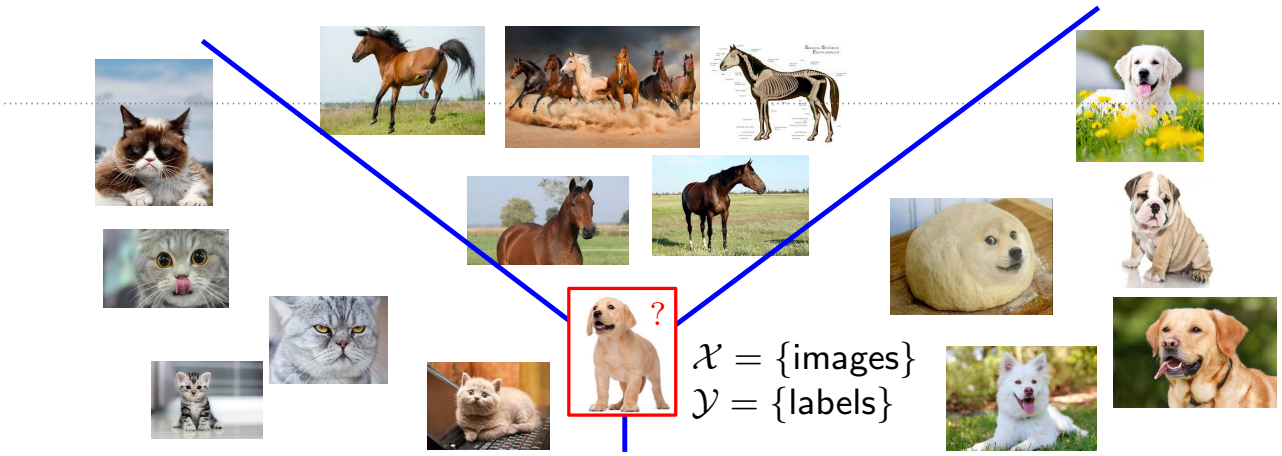
Outline

- **Reminder about supervised classification**
- **Principles of linear methods for classification**
- **Textbook case: linear regression for classification**
- **Logistic regression:**
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- **Non-linear classification using kernels**

Supervised learning (classification)

Input: n observations + responses $(x_1, y_1), \cdots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$

Goal: build a predictor $f : \mathcal{X} \to \mathcal{Y}$ from $(x_1, y_1), \cdots, (x_n, y_n)$
whose mean prediction error on new query observations is minimal



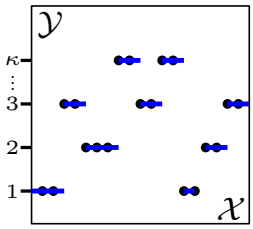
classification: $\mathcal{Y}$ discrete

Statistical framework

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

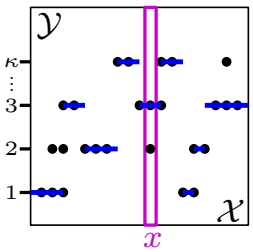
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

$\rightarrow$ the **joint distribution** $\Pr(X, Y)$ encodes the complexity of the problem



X, Y perfectly dependent $\Rightarrow \exists$ perfect predictor

X, Y imperfectly dependent $\Rightarrow \nexists$ perfect predictor



Statistical framework

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

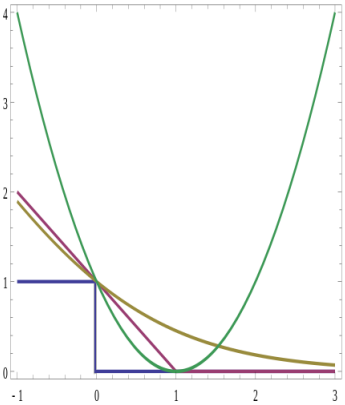
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

$\rightarrow$ the **joint distribution** $\Pr(X, Y)$ encodes the complexity of the problem

Prediction error is measured by a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

$\rightarrow$ goal: minimize **risk** (expected prediction error): $\mathbb{E}_{(X,Y)}\, L(Y,\, f(X))$

$\rightarrow$ in practice: minimize **empirical risk**: $\dfrac{1}{n} \sum_{i=1}^{n} L(y_i,\, f(x_i))$



Classification with 0-1 loss

Hyp: $x_i \overset{\text{iid}}{\sim} X$ with values in $\mathcal{X} = \mathbb{R}^d$

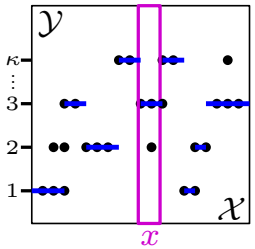
$y_i \overset{\text{iid}}{\sim} Y$ with values in $\mathcal{Y} = \{1, \cdots, \kappa\}$ (classification)

Risk: $R(f) = \mathbb{E}_{(X,Y)}\, \mathbb{1}_{Y \neq f(X)}$

$= \mathbb{E}_X\, \mathbb{E}_{(Y|X)}\, \left[\mathbb{1}_{Y \neq f(X)} \mid X\right]$ (conditioning on X)

$\rightarrow$ minimize risk pointwise (i.e. independently for each value x of X):

$$f^*(x) := \operatorname*{argmin}_{y \in \{1,\cdots,\kappa\}} \mathbb{E}_{(Y|X)}\, [\mathbb{1}_{Y \neq y} \mid X = x]$$



$$= \operatorname*{argmin}_{y \in \{1,\cdots,\kappa\}} \sum_{r=1}^{\kappa} \mathbb{1}_{r \neq y}\, \mathbb{P}(Y = r \mid X = x) \quad (Y \text{ categorical variable})$$

$$= \operatorname*{argmin}_{y \in \{1,\cdots,\kappa\}} 1 - \mathbb{P}(Y = y \mid X = x) = \operatorname*{argmax}_{y \in \{1,\cdots,\kappa\}} \mathbb{P}(Y = y \mid X = x)$$

(best prediction at x **maximizes the posterior probability** $\mathbb{P}(Y \mid X)$ (Bayes classifier))

Advantages and drawbacks of k-NN in practice

- high flexibility:
  - ▶ little prior on the fitting model ← **prior:** linear model
  - ▶ method based on distances / (dis-)similarities (no need for coordinates) ← requires linear space hence coordinates
- easiness of implementation:
  - ▶ only a few lines of code for NN-search via linear scan ← easy and efficient prediction (dot-product)
- algorithmic cost of prediction: ← algorithmic cost put on pre-training
  - ▶ linear scan in $\Theta(nd)$
  - ▶ sublinear methods become (close to) linear in high dimensions
- slow convergence in high dimensions (**curse of dimensionality**): ← no hyper-parameter
  - ▶ asymptotic regime often not attained in practice $\rightsquigarrow$ **need to select** k

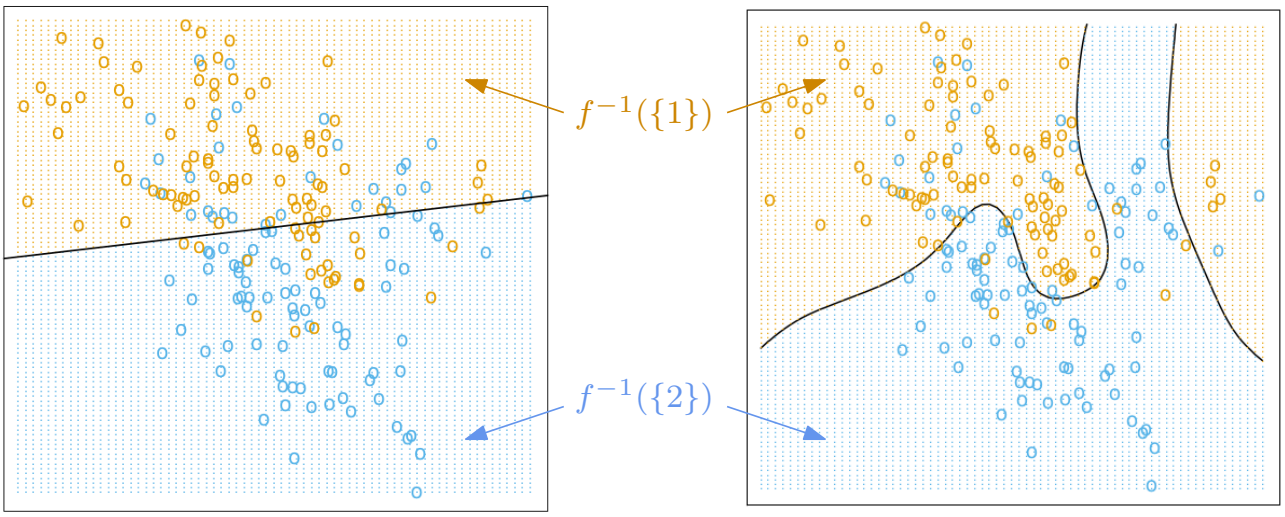
Outline

- **Reminder about supervised classification**
- **Principles of linear methods for classification**
- **Textbook case: linear regression for classification**
- **Logistic regression:**
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- **Non-linear classification using kernels**

Linear methods for classification

Response variable Y is discrete

- consider the **fibers** of the predictor f: $f^{-1}(\{1\}), \cdots, f^{-1}(\{\kappa\})$
- a linear classifier produces **linear decision boundaries**



Linear methods for classification

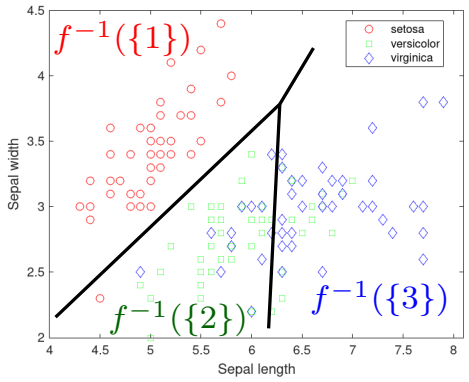
Response variable Y is discrete

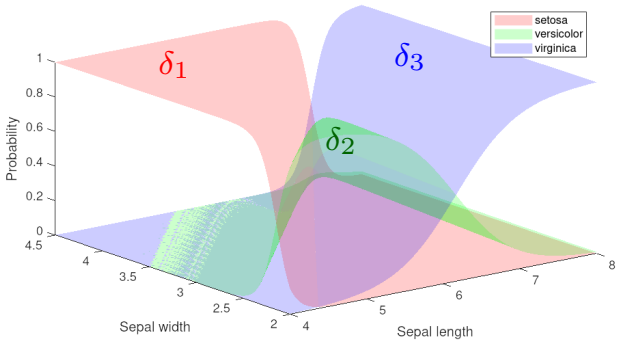
- consider the **fibers** of the predictor f: $f^{-1}(\{1\}), \cdots, f^{-1}(\{\kappa\})$
- a linear classifier produces **linear decision boundaries**

2 types of approaches:

- model posterior probability (**discriminant function** δ_y) for each class y, then classify by taking $\operatorname{argmax}_{y \in \{1,\cdots,\kappa\}} \delta_y(x)$

  e.g. linear / logistic regression, LDA





Linear methods for classification

Response variable Y is discrete

- consider the **fibers** of the predictor f: $f^{-1}(\{1\}), \cdots, f^{-1}(\{\kappa\})$
- a linear classifier produces **linear decision boundaries**

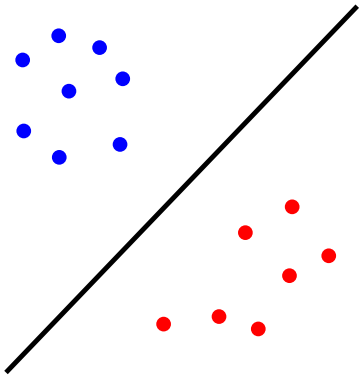
2 types of approaches:

- model posterior probability (**discriminant function** δ_y) for each class y, then classify by taking $\operatorname{argmax}_{y \in \{1,\cdots,\kappa\}} \delta_y(x)$

  e.g. linear / logistic regression, LDA

- model the separating hyperplane(s) directly

  e.g. SVM, perceptron, decision tree



Outline

- **Reminder about supervised classification**
- **Principles of linear methods for classification**
- **Textbook case: linear regression for classification**
- **Logistic regression:**
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- **Non-linear classification using kernels**

Linear regression for classification

Use a **linear model** for the discriminant functions (yields linear decision boundaries):

- $\forall y \in \{1, \cdots, \kappa\}, \quad \delta_y(x) := \left[\, 1 \; x^T \,\right] \beta_y$ for some parameter vector $\beta_y \in \mathbb{R}^{d+1}$
- matrix of parameters: $\mathbf{B} := \left[\, \beta_1 \; \cdots \; \beta_\kappa \,\right] \in \mathbb{R}^{d+1 \times \kappa}$

Fit the model by least squares: $\hat{\mathbf{B}} := \underset{\mathbf{B}}{\operatorname{argmin}} \sum_{i=1}^{n} \| Z(y_i) - \left[\, 1 \; x_i^T \,\right] \mathbf{B} \|_2^2$

where $Z(y_i) = \left[\, \mathbb{1}_{y_i=1} \; \cdots \; \mathbb{1}_{y_i=\kappa} \,\right]$ is the **row indicator vector** of response y_i

Linear regression for classification

Use a **linear model** for the discriminant functions (yields linear decision boundaries):

- $\forall y \in \{1, \cdots, \kappa\}, \quad \delta_y(x) := \begin{bmatrix} 1 & x^T \end{bmatrix} \beta_y$ for some parameter vector $\beta_y \in \mathbb{R}^{d+1}$
- matrix of parameters: $\mathbf{B} := \begin{bmatrix} \beta_1 & \cdots & \beta_\kappa \end{bmatrix} \in \mathbb{R}^{d+1 \times \kappa}$

Fit the model by least squares: $\hat{\mathbf{B}} := \underset{\mathbf{B}}{\operatorname{argmin}} \|\mathbf{Z} - \mathbf{X}\,\mathbf{B}\|_F^2$

(Frobenius norm: $\|M\|_F^2 := \sum_{i,j} M_{ij}^2$)

where $Z(y_i) = \begin{bmatrix} \mathbb{1}_{y_i=1} & \cdots & \mathbb{1}_{y_i=\kappa} \end{bmatrix}$ is the **row indicator vector** of response y_i

- input indicator response matrix: $\mathbf{Z} := \begin{bmatrix} Z(y_1) \\ \vdots \\ Z(y_n) \end{bmatrix} \in \mathbb{R}^{n \times \kappa}$
- random row indicator vector: $Z(Y) := \begin{bmatrix} \mathbb{1}_{Y=1} & \cdots & \mathbb{1}_{Y=\kappa} \end{bmatrix} \in \mathbb{R}^{1 \times \kappa}$

Linear regression for classification

Use a **linear model** for the discriminant functions (yields linear decision boundaries):

- $\forall y \in \{1, \cdots, \kappa\}, \quad \delta_y(x) := \begin{bmatrix} 1 & x^T \end{bmatrix} \beta_y$ for some parameter vector $\beta_y \in \mathbb{R}^{d+1}$
- matrix of parameters: $\mathbf{B} := \begin{bmatrix} \beta_1 & \cdots & \beta_\kappa \end{bmatrix} \in \mathbb{R}^{d+1 \times \kappa}$

Fit the model by least squares: $\hat{\mathbf{B}} := \underset{\mathbf{B}}{\operatorname{argmin}} \|\mathbf{Z} - \mathbf{X}\,\mathbf{B}\|_F^2$

Unique minimum (assuming full column rank in $\mathbf{X}^T\mathbf{X}$): $\hat{\mathbf{B}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{Z}$

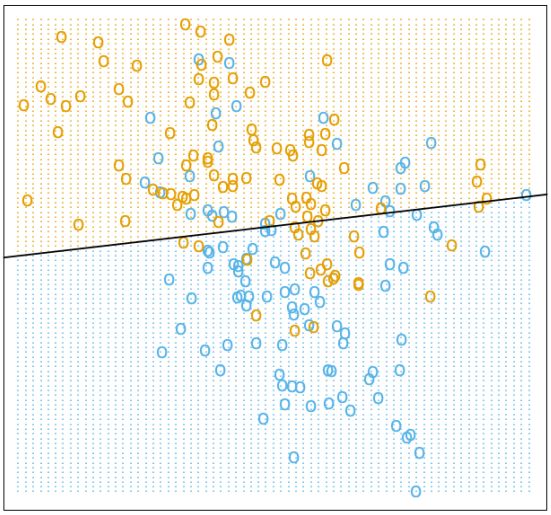
- row vector of estimated discriminant functions:

  $\hat{\delta}(x) := \begin{bmatrix} \hat{\delta}_1(x) & \cdots & \hat{\delta}_\kappa(x) \end{bmatrix} = \begin{bmatrix} 1 & x^T \end{bmatrix} \hat{\mathbf{B}}$

- classifier: $\hat{f}(x) := \underset{y \in \{1, \cdots, \kappa\}}{\operatorname{argmax}} \hat{\delta}_y(x)$

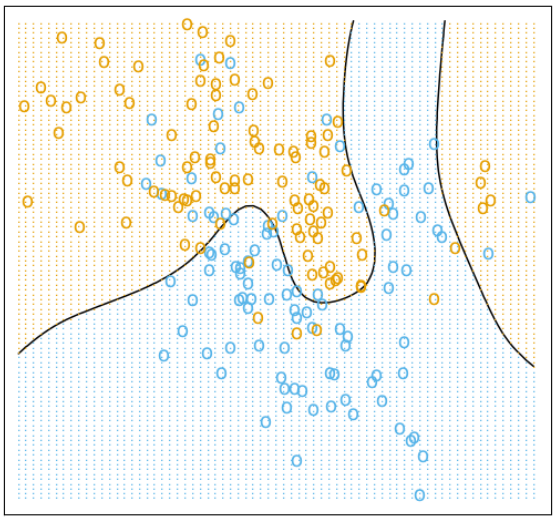
Linear regression for classification

Experimental results: $n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



linear regression

error rate $\approx 34\%$



Bayes classifier

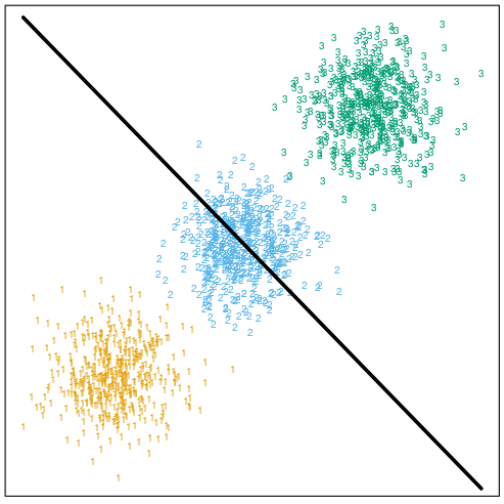
error rate $\approx 21\%$

Linear regression for classification

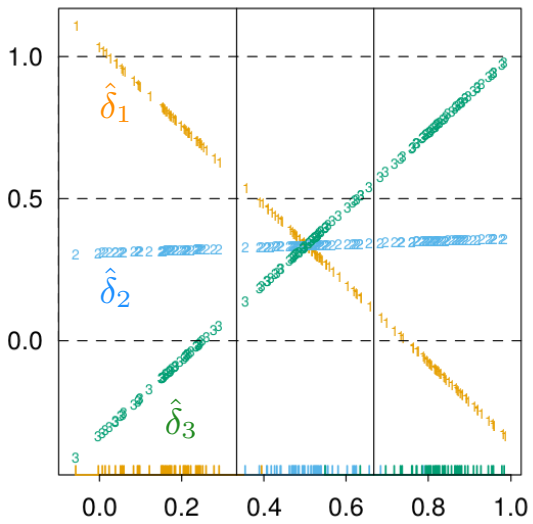
Experimental results:

$n = 100 + 100 + 100$, $d = 2$, Bayes error rate $\approx 2.5\%$

error rate $\approx 33\%$



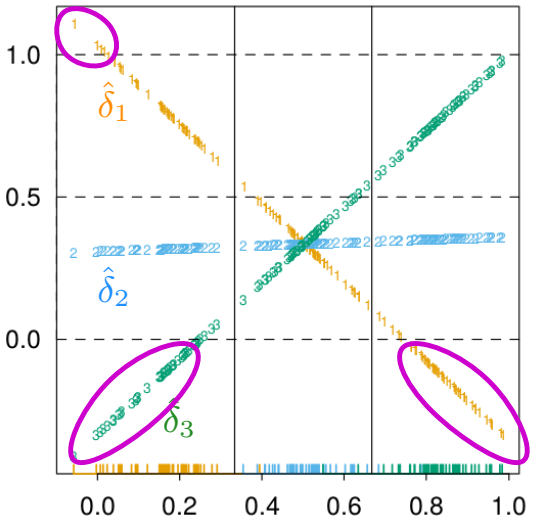
diagonal cross-section



Linear regression for classification

What is happening:

- the $\hat{\delta}_y(x)$ supposedly model the posterior probabilities $\mathbb{P}(Y = y \mid X = x)$ (cf. Bayes classifier)
- they sum up to 1 (in centered model)
- they may fall outside $[0, 1]$



Outline

- Reminder about supervised classification
- Principles of linear methods for classification
- Textbook case: linear regression for classification
- **Logistic regression:**
  - binary
  - multi-class
- Support Vector Machines (SVM):
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- Non-linear classification using kernels

Logistic regression for binary classification

Generalized linear model for discriminant functions:

- $\delta_1(x) := \sigma\left([\,1\ x^T\,]\,\beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

  forces $\delta_1(x) \in [0,1]$

- $\delta_2(x) := 1 - \delta_1(x)$

  forces $\delta_1(x) + \delta_2(x) = 1$, hence $\delta_2(x) \in [0,1]$

Logistic regression for binary classification

Generalized linear model for discriminant functions:

- $\delta_1(x) := \sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

- $\delta_2(x) := 1 - \delta_1(x) = \sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$ ← symmetric problem in $1, 2$

Logistic regression for binary classification

Generalized linear model for discriminant functions:

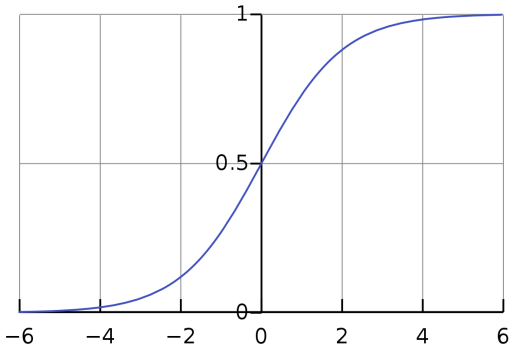
- $\delta_1(x) := \sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

- $\delta_2(x) := 1 - \delta_1(x) = \sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$

Properties of the logistic sigmoid:

- monotonic homeomorphism $\mathbb{R} \to (0, 1)$
- $\sigma^{-1}(u) = \ln \frac{u}{1-u}$ (logit function)
- $\sigma(t) + \sigma(-t) = 1$
- $\sigma'(t) = \sigma(t)\,(1 - \sigma(t))$



Logistic regression for binary classification

Generalized linear model for discriminant functions:

- $\delta_1(x) := \sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

- $\delta_2(x) := 1 - \delta_1(x) = \sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$

Properties of the regression and associated classifier:

- model for δ_y $(y = 1, 2)$ assumes $\mathbb{P}(Y = y \mid X = x)$ follows **logistic distribution**

$$\mathbb{P}(Y = 1 \mid X = x) = \delta_1(x) = \sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$$

$$\mathbb{P}(Y = 2 \mid X = x) = 1 - \delta_1(x) = \delta_2(x) = \sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$$

Logistic regression for binary classification

Generalized linear model for discriminant functions:

- $\delta_1(x) := \sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

- $\delta_2(x) := 1 - \delta_1(x) = \sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)$

Properties of the regression and associated classifier:

- model for δ_y $(y = 1, 2)$ assumes $\mathbb{P}(Y = y \mid X = x)$ follows **logistic distribution**

- model makes **probability ratio log-linear**

$$\ln \frac{\mathbb{P}(Y = 1 \mid X = x)}{\mathbb{P}(Y = 2 \mid X = x)} = \ln \frac{\sigma\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)}{\sigma\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)} = \ln \frac{1+\exp\left(\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)}{1+\exp\left(-\left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1\right)} = \left[\begin{smallmatrix}1 & x^T\end{smallmatrix}\right]\beta_1$$

Logistic regression for binary classification

Generalized linear model for discriminant functions:

- $\delta_1(x) := \sigma\left([\,1\ x^T\,]\ \beta_1\right)$ for some parameter $\beta_1 \in \mathbb{R}^{d+1}$

  where $\sigma(t) := \dfrac{\exp(t)}{1+\exp(t)} = \dfrac{1}{1+\exp(-t)}$ (logistic sigmoid function)

- $\delta_2(x) := 1 - \delta_1(x) = \sigma\left(-[\,1\ x^T\,]\ \beta_1\right)$

Properties of the regression and associated classifier:

- model for δ_y $(y = 1, 2)$ assumes $\mathbb{P}(Y = y \mid X = x)$ follows **logistic distribution**
- model makes **probability ratio log-linear**
- yields **linear decision boundary**: $\delta_1(x) = \delta_2(x) \iff \delta_1(x) = 1/2$
  $\iff [\,1\ x^T\,]\ \beta_1 = \sigma^{-1}(1/2)$

Logistic regression for binary classification

Model fitting by **maximum likelihood**:

$$\forall i, \quad \mathcal{L}(y_i; x_i, \beta_1) := \mathbb{P}(Y = y_i \mid X = x_i; \beta_1)$$

$$\Rightarrow \mathcal{L}((y_i)_{i=1}^n; (x_i)_{i=1}^n, \beta_1) = \prod_{i=1}^n \mathbb{P}(Y = y_i \mid X = x_i; \beta_1) \qquad \text{(independence)}$$

$$\Rightarrow \log \mathcal{L}((y_i)_{i=1}^n; (x_i)_{i=1}^n, \beta_1) = \sum_{i=1}^n \log \mathbb{P}(Y = y_i \mid X = x_i; \beta_1)$$

Change of variable: $Z := \mathbb{1}_{Y=1} \in \{0, 1\}$ $\qquad \forall i,\ z_i := \mathbb{1}_{y_i=1} \in \{0, 1\}$

$$\begin{aligned}
\Rightarrow \forall i, \quad & \log \mathbb{P}(Z = z_i \mid X = x_i; \beta_1) \\
& = z_i \log \mathbb{P}(Z = 1 \mid X = x_i; \beta_1) + (1 - z_i) \log \mathbb{P}(Z = 0 \mid X = x_i; \beta_1) \\
& = z_i \log \sigma\left([\,1\ x_i^T\,]\ \beta_1\right) + (1 - z_i) \log \sigma\left(-[\,1\ x_i^T\,]\ \beta_1\right) \\
& = z_i\ [\,1\ x_i^T\,]\ \beta_1 - \log\left(1 + \exp\left([\,1\ x_i^T\,]\ \beta_1\right)\right)
\end{aligned}$$

Logistic regression for binary classification

Model fitting by **maximum likelihood**:

$$\hat{\beta}_1 := \underset{\beta_1}{\operatorname{argmax}}\ \ell(\beta_1) \text{ where } \ell(\beta_1) := \log \mathcal{L}\left((y_i)_{i=1}^n; (x_i)_{i=1}^n, \beta_1\right)$$
$$= \sum_{i=1}^n z_i \left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1 - \log\left(1 + \exp\left(\left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1\right)\right)$$

$$\begin{cases} \nabla \ell(\beta_1) = \sum_{i=1}^n \left(z_i - \sigma\left(\left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1\right)\right) \left[\begin{smallmatrix}1 \\ x_i\end{smallmatrix}\right] \\ \nabla^2 \ell(\beta_1) = -\sum_{i=1}^n \sigma\left(\left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1\right)\left(1 - \sigma\left(\left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1\right)\right) \left[\begin{smallmatrix}1 \\ x_i\end{smallmatrix}\right] \left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \end{cases}$$

negative semi-definite $\Rightarrow$ $\ell(\beta_1)$ is a **concave** function

- choose $\hat{\beta}_1$ arbitrarily in the solution set of $\nabla \ell(\beta_1) = 0$

$d+1$ non-linear equations in β_1

Logistic regression for binary classification

Model fitting by **maximum likelihood**:

$$\hat{\beta}_1 := \underset{\beta_1}{\operatorname{argmax}}\ \ell(\beta_1) \text{ where } \ell(\beta_1) := \log \mathcal{L}\left((y_i)_{i=1}^n; (x_i)_{i=1}^n, \beta_1\right)$$
$$= \sum_{i=1}^n z_i \left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1 - \log\left(1 + \exp\left(\left[\begin{smallmatrix}1 & x_i^T\end{smallmatrix}\right] \beta_1\right)\right)$$

Newton-Raphson's method:

INIT: $\hat{\beta}_1 \longleftarrow 0$ `//or some arbitrary vector`

REPEAT:

$\hat{\beta}_1 \longleftarrow \hat{\beta}_1 - \left(\nabla^2 \ell(\hat{\beta}_1)\right)^{-1} \nabla \ell(\hat{\beta}_1)$ `// assuming non-singular Hessian`

UNTIL CONVERGENCE `// requires convergence threshold`

Thm: if a maximum $\bar{\beta}_1$ is s.t. $\nabla^2 \ell(\bar{\beta}_1)$ is non-singular, then $\bar{\beta}_1$ is the **unique max.** and, for an initial $\hat{\beta}_1$ close enough to $\bar{\beta}_1$, convergence to $\bar{\beta}_1$ is **quadratic**.

Logistic regression for binary classification

Degenerate cases (singular Hessian):

- regularized logistic regression:

$$\hat{\beta}_1 := \underset{\beta_1}{\operatorname{argmax}} \sum_{i=1}^{n} \left[z_i \left[1 \; x_i^T \right] \beta_1 - \log\left(1 + \exp\left(\left[1 \; x_i^T \right] \beta_1\right)\right)\right] - \lambda \left\|\beta_1\right\|_p^p$$

- case $p = 2$ (Tikhonov):

$$\begin{cases} \nabla \ell(\beta_1) = \sum_{i=1}^{n} \left(z_i - \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\right) \left[\begin{smallmatrix} 1 \\ x_i \end{smallmatrix} \right] - 2\lambda \beta_1 \\ \nabla^2 \ell(\beta_1) = -\sum_{i=1}^{n} \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\left(1 - \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\right) \left[\begin{smallmatrix} 1 \\ x_i \end{smallmatrix} \right] \left[1 \; x_i^T \right] - 2\lambda \mathbf{I}_{d+1} \end{cases}$$

negative definite $\Rightarrow$ **strictly concave** functional

Logistic regression for binary classification

Degenerate cases (singular Hessian):

- regularized logistic regression:

$$\hat{\beta}_1 := \underset{\beta_1}{\operatorname{argmax}} \sum_{i=1}^{n} \left[z_i \left[1 \; x_i^T \right] \beta_1 - \log\left(1 + \exp\left(\left[1 \; x_i^T \right] \beta_1\right)\right)\right] - \lambda \left\|\beta_1\right\|_p^p$$

- case $p = 2$ (Tikhonov):

$$\begin{cases} \nabla \ell(\beta_1) = \sum_{i=1}^{n} \left(z_i - \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\right) \left[\begin{smallmatrix} 1 \\ x_i \end{smallmatrix} \right] - 2\lambda \beta_1 \\ \nabla^2 \ell(\beta_1) = -\sum_{i=1}^{n} \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\left(1 - \sigma\left(\left[1 \; x_i^T \right] \beta_1\right)\right) \left[\begin{smallmatrix} 1 \\ x_i \end{smallmatrix} \right] \left[1 \; x_i^T \right] - 2\lambda \mathbf{I}_{d+1} \end{cases}$$

- apply Newton-Raphson, with guaranteed quadratic convergence

  (small values λ may lead to numerical instabilities in practice though)

Outline

- Reminder about supervised classification
- Principles of linear methods for classification
- Textbook case: linear regression for classification
- **Logistic regression:**
  - binary
  - multi-class
- Support Vector Machines (SVM):
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- Non-linear classification using kernels

Multi-class logistic regression

Log-linear model for posterior probability ratios:

$$\begin{cases} \ln \dfrac{\mathbb{P}(Y=1 \mid X=x)}{\mathbb{P}(Y=\kappa \mid X=x)} = [\,1 \;\; x^T\,]\;\beta_1 \\ \qquad\qquad \vdots \\ \ln \dfrac{\mathbb{P}(Y=\kappa-1 \mid X=x)}{\mathbb{P}(Y=\kappa \mid X=x)} = [\,1 \;\; x^T\,]\;\beta_{\kappa-1} \end{cases}$$

parameter matrix

$\mathbf{B} := [\,\beta_1 \;\cdots\; \beta_{\kappa-1}\,] \in \mathbb{R}^{d+1\times\kappa-1}$

▶ generalized linear model for discriminant functions:

$$\begin{cases} \delta_y(x) := \mathbb{P}(Y=y \mid X=x) = \dfrac{\exp\left([\,1 \;\; x^T\,]\;\beta_y\right)}{1+\sum_{z<\kappa}\exp\left([\,1 \;\; x^T\,]\;\beta_z\right)} & \text{for } y=1,\cdots,\kappa-1 \\ \delta_\kappa(x) := \mathbb{P}(Y=\kappa \mid X=x) = \dfrac{1}{1+\sum_{z<\kappa}\exp\left([\,1 \;\; x^T\,]\;\beta_z\right)} & \end{cases}$$

generalized sigmoid (**softmax**) $\in [0,1]$

$= 1 - \sum_{y<\kappa}\delta_y(x)$

Multi-class logistic regression

Log-linear model for posterior probability ratios:

$$\begin{cases} \ln \dfrac{\mathbb{P}(Y=1 \mid X=x)}{\mathbb{P}(Y=\kappa \mid X=x)} = [\,1 \;\; x^T\,]\;\beta_1 \\ \qquad\qquad \vdots \\ \ln \dfrac{\mathbb{P}(Y=\kappa-1 \mid X=x)}{\mathbb{P}(Y=\kappa \mid X=x)} = [\,1 \;\; x^T\,]\;\beta_{\kappa-1} \end{cases}$$

parameter matrix

$$\mathbf{B} := [\,\beta_1 \;\cdots\; \beta_{\kappa-1}\,] \in \mathbb{R}^{d+1\times\kappa-1}$$

- generalized linear model for discriminant functions:

$$\begin{cases} \delta_y(x) := \mathbb{P}(Y=y \mid X=x) = \dfrac{\exp\left([\,1 \;\; x^T\,]\;\beta_y\right)}{1+\sum_{z<\kappa}\exp\left([\,1 \;\; x^T\,]\;\beta_z\right)} & \text{for } y=1,\cdots,\kappa-1 \\ \\ \delta_\kappa(x) := \mathbb{P}(Y=\kappa \mid X=x) = \dfrac{1}{1+\sum_{z<\kappa}\exp\left([\,1 \;\; x^T\,]\;\beta_z\right)} \end{cases}$$

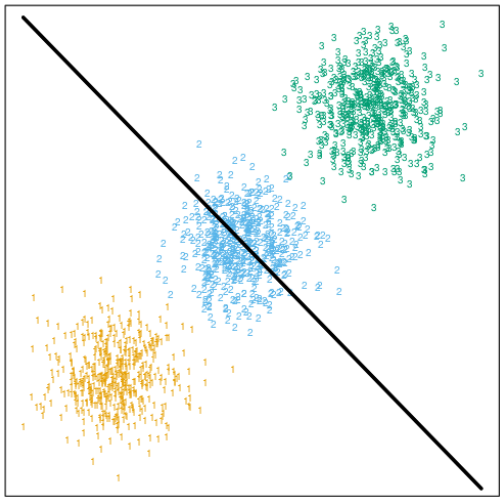
- estimate $\mathbf{B}$ by maximum likelihood and Newton-Raphson's algorithm.

Multi-class logistic regression

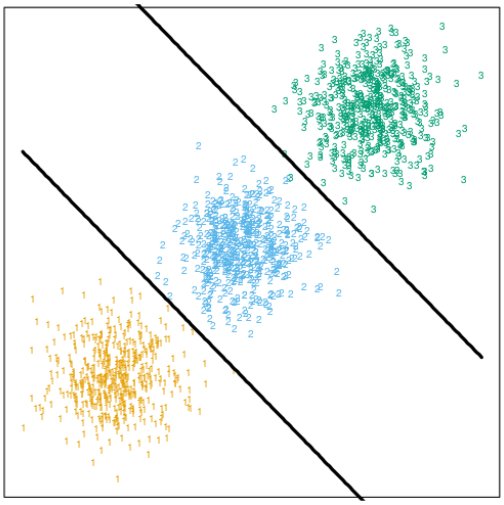
Back to our running experiment:

$n = 100 + 100 + 100$, $d = 2$, Bayes error rate $\approx 2.5\%$

linear (error rate $\approx 33\%$)



logistic (error rate $\approx 3\%$)



Outline

- Reminder about supervised classification
- Principles of linear methods for classification
- Textbook case: linear regression for classification
- Logistic regression:
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- Non-linear classification using kernels

Support Vector Machines (SVM)

Principle: **explicitly construct** the 'best' hyperplanes separating the various classes.

▶ the hyperplanes that maximize the **margins** (closest distances to data points)

Hyperplane equation: $x^T\beta - \beta_0 = 0$

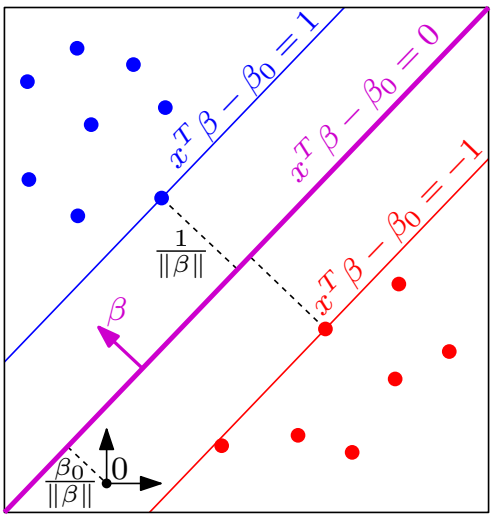
▶ parameters: $\beta \in \mathbb{R}^d \setminus \{0\}$, $\beta_0 \in \mathbb{R}$

▶ β is normal to the hyperplane

▶ $\frac{\beta_0}{\|\beta\|}$ is the shift from the origin along β

▶ fix $\frac{1}{\|\beta\|}$ to be the margin

⇒ **maximizing the margin is equivalent to minimizing $\|\beta\|$ or $\|\beta\|^2$**



⇒ slab boundaries have equations $x^T\beta - \beta_0 = \pm 1$

Support Vector Machines (SVM)

Principle: **explicitly construct** the 'best' hyperplanes separating the various classes.

- the hyperplanes that maximize the **margins** (closest distances to data points)

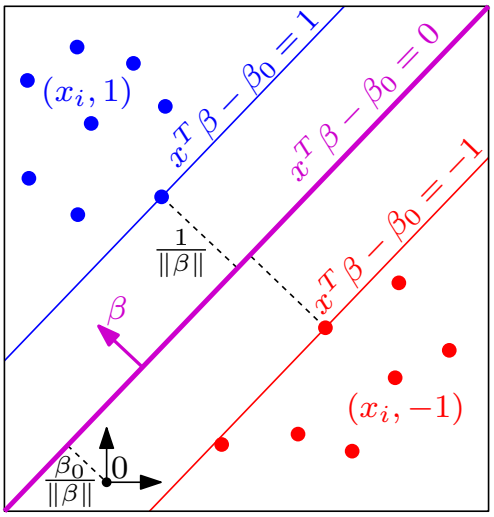
Binary classification case ($\mathcal{Y} = \{-1, 1\}$):

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \|\beta\|^2 \text{ subject to:} \quad \text{(maximize margin)}$$

$$\begin{cases} x_i^T \beta - \beta_0 \geq 1 & \forall i \text{ s.t. } y_i = 1 \\ x_i^T \beta - \beta_0 \leq -1 & \forall i \text{ s.t. } y_i = -1 \end{cases}$$

$$\Leftrightarrow y_i \left(x_i^T \beta - \beta_0 \right) \geq 1 \quad \forall i = 1, \cdots, n$$

(leave data points outside slab, on correct side)



- quadratic programming problem $\rightsquigarrow$ solvers: ellipsoid, interior point, etc. (w/ pos. definite quadratic form)

Support Vector Machines (SVM)

Principle: **explicitly construct** the 'best' hyperplanes separating the various classes.

- the hyperplanes that maximize the **margins** (closest distances to data points)

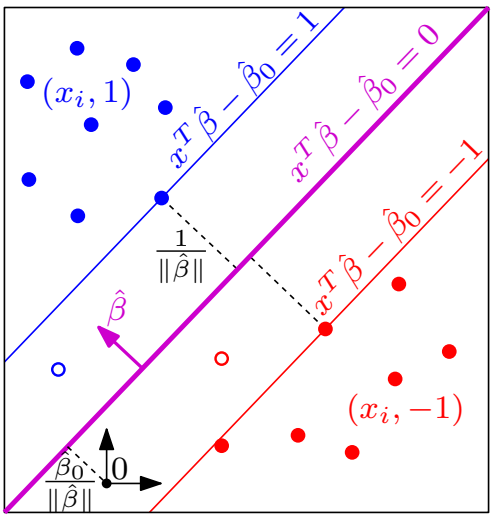
Binary classification case ($\mathcal{Y} = \{-1, 1\}$):

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \|\beta\|^2 \text{ subject to:} \quad \text{(maximize margin)}$$

$$\begin{cases} x_i^T \beta - \beta_0 \geq 1 & \forall i \text{ s.t. } y_i = 1 \\ x_i^T \beta - \beta_0 \leq -1 & \forall i \text{ s.t. } y_i = -1 \end{cases}$$

$$\Leftrightarrow y_i \left(x_i^T \beta - \beta_0 \right) \geq 1 \quad \forall i = 1, \cdots, n$$

(leave data points outside slab, on correct side)



- quadratic programming problem (w/ pos. definite quadratic form)
- classifier: $\hat{f}(x) = \operatorname{sign}\left(x^T \hat{\beta} - \hat{\beta}_0 \right)$

Outline

- Reminder about supervised classification
- Principles of linear methods for classification
- Textbook case: linear regression for classification
- Logistic regression:
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- Non-linear classification using kernels

When classes are not linearly separable

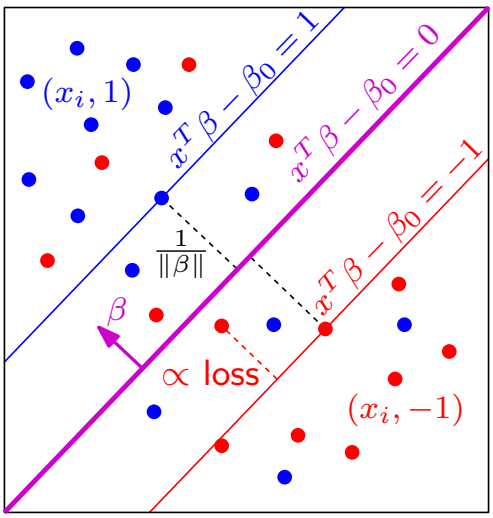
Hinge loss: $\max\left\{0, 1 - y_i\left(x_i^T\beta - \beta_0\right)\right\}$

(loss compared to a slab excluding observation x_i)

Relaxed optimization (soft margin):

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta,\beta_0}{\operatorname{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n} \max\left\{0, 1 - y_i\left(x_i^T\beta - \beta_0\right)\right\}}_{\text{(minimize mean loss)}} + \underbrace{\lambda\|\beta\|^2}_{\text{(maximize margin)}}$$

mixing (trade-off) parameter



- when classes are linearly separable, recover problem with hard constraints by taking $\lambda > 0$ small enough.

When classes are not linearly separable

Hinge loss: $\max\left\{0, 1 - y_i\left(x_i^T\beta - \beta_0\right)\right\}$

(loss compared to a slab excluding observation x_i)

Conversion to quadratic program:

- slack variables:

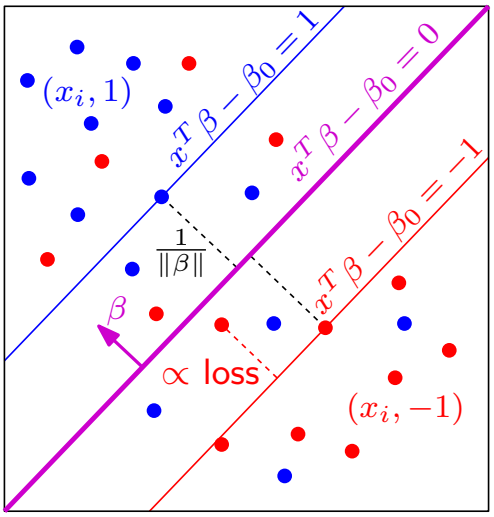
$$\xi_i := \max\left\{0, 1 - y_i(x_i^T\beta - \beta_0)\right\}$$

- substitution:

$$\hat\beta, \hat\beta_0, (\hat\xi_i)_{i=1}^n := \underset{\beta,\beta_0,(\xi_i)_{i=1}^n}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^n \xi_i + \lambda\|\beta\|^2$$

subject to:

$$\forall i \quad \xi_i \geq 0 \text{ and } y_i(x_i^T\beta - \beta_0) \geq 1 - \xi_i$$



($\forall i$, optimum turns one inequality into an equality $\rightsquigarrow \hat\xi_i = \max\left\{0, 1 - y_i(x_i^T\hat\beta - \hat\beta_0)\right\}$)

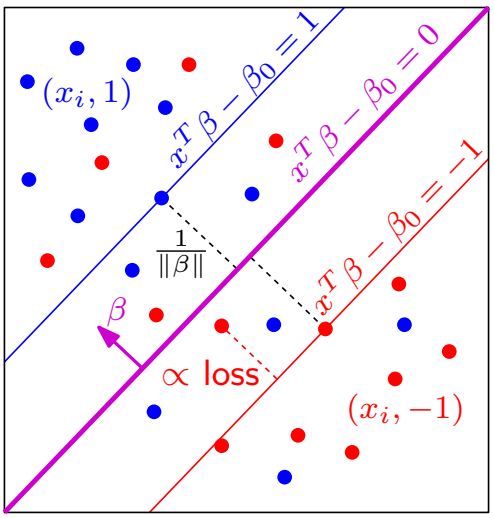
When classes are not linearly separable

Hinge loss: $\max\left\{0, 1 - y_i\left(x_i^T\beta - \beta_0\right)\right\}$

(loss compared to a slab excluding observation x_i)

Interpretation:

- $\hat\xi_i > 0$: x_i is on wrong side of slab boundary
- $\hat\xi_i = 0$: **support vectors**
  - $y_i(x_i^T\hat\beta - \hat\beta_0) = 1$: x_i is on slab boundary
  - $y_i(x_i^T\hat\beta - \hat\beta_0) > 1$: x_i is on correct side



$$\hat\beta, \hat\beta_0, (\hat\xi_i)_{i=1}^n := \underset{\beta,\beta_0,(\xi_i)_{i=1}^n}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^n \xi_i + \lambda\|\beta\|^2$$

subject to:

$$\forall i \quad \xi_i \geq 0 \text{ and } y_i(x_i^T\beta - \beta_0) \geq 1 - \xi_i$$

($\forall i$, optimum turns one inequality into an equality $\rightsquigarrow \hat\xi_i = \max\left\{0, 1 - y_i(x_i^T\hat\beta - \hat\beta_0)\right\}$)

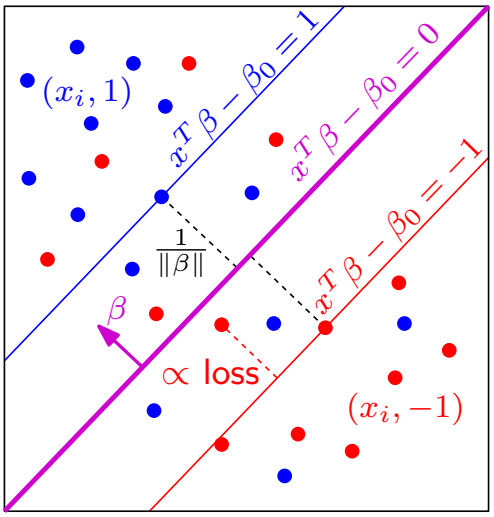
When classes are not linearly separable

Hinge loss: $\max\left\{0, 1 - y_i\left(x_i^T\beta - \beta_0\right)\right\}$

(loss compared to a slab excluding observation x_i)

Interpretation:

- $\hat{\xi}_i > 0$: x_i is on wrong side of slab boundary
- $\hat{\xi}_i = 0$:
  - $y_i(x_i^T\hat{\beta} - \hat{\beta}_0) = 1$: x_i is on slab boundary
  - $y_i(x_i^T\hat{\beta} - \hat{\beta}_0) > 1$: x_i is on correct side



$$\hat{\beta}, \hat{\beta}_0, (\hat{\xi}_i)_{i=1}^n := \underset{\beta, \beta_0, (\xi_i)_{i=1}^n}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^n \xi_i + \lambda\|\beta\|^2$$

(select e.g. by cross-validation)

subject to:

$$\forall i \quad \xi_i \geq 0 \text{ and } y_i(x_i^T\beta - \beta_0) \geq 1 - \xi_i$$

Outline

- Reminder about supervised classification
- Principles of linear methods for classification
- Textbook case: linear regression for classification
- Logistic regression:
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- Non-linear classification using kernels

Multi-class SVM

Principle: convert multi-class problem into multiple binary problems.

▶ **One-vs-all:**

- train 1 classifier $(\hat{\beta}^y, \hat{\beta}_0^y)$ for each class $y = 1, \cdots, \kappa$, to discriminate this class (assigned label 1) from the rest of the data (assigned label -1).
- assign each new observation $x \in \mathbb{R}^d$ to the class $\underset{y=1,\cdots,\kappa}{\operatorname{argmax}} x^T \hat{\beta}^y - \hat{\beta}_0^y$

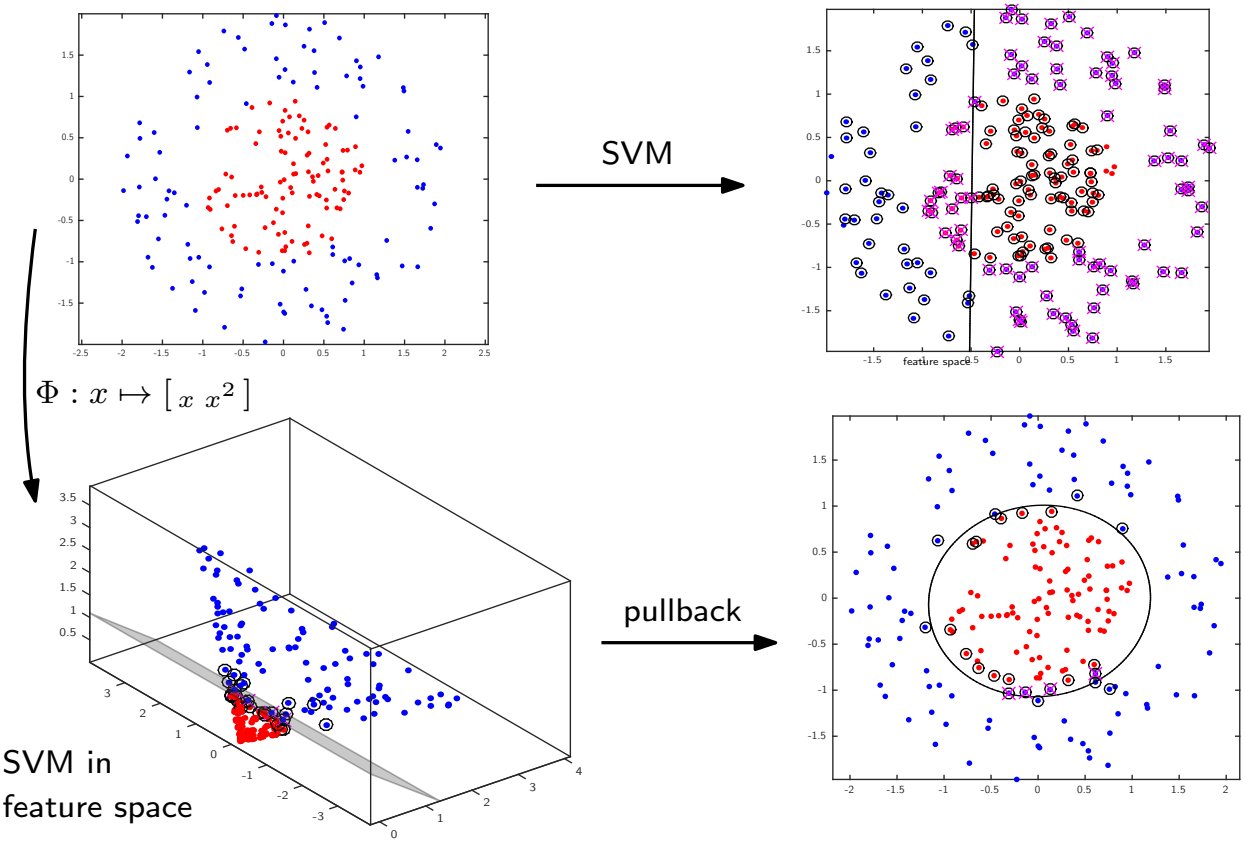
▶ **One-vs-one:**

- train 1 classifier $(\hat{\beta}^{y,y'}, \hat{\beta}_0^{y,y'})$ for each pair of classes $y \neq y' \in \{1, \cdots, \kappa\}$, to discriminate y from y' in their joint subpopulations.
- given a new observation $x \in \mathbb{R}^d$, decide between y and y' using $\operatorname{sign}\left(x^T \hat{\beta}^{y,y'} - \hat{\beta}_0^{y,y'}\right)$ for each pair of classes $y \neq y'$, then assign x to the class with the highest number of positive answers.

Outline

- **Reminder about supervised classification**
- **Principles of linear methods for classification**
- **Textbook case: linear regression for classification**
- **Logistic regression:**
  - binary
  - multi-class
- **Support Vector Machines (SVM):**
  - binary, linearly separable classes
  - binary, non-linearly separable classes
  - multi-class
- **Non-linear classification using kernels**

Kernel SVM



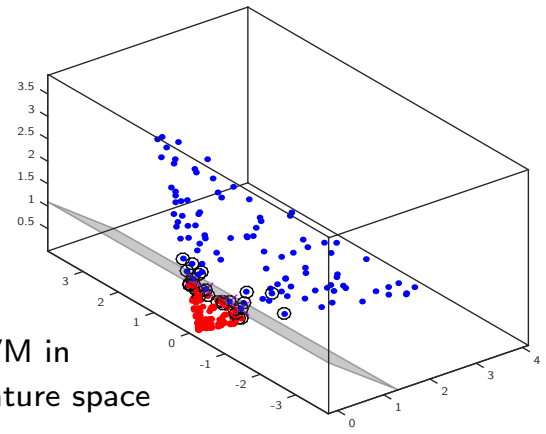
Kernel SVM

Quadratic program (hard margin / no slack):

$$\underset{\beta,\beta_0}{\operatorname{argmin}} \|\beta\|^2 \quad \text{subject to} \quad y_i(\Phi(x_i)^T\beta - \beta_0) \geq 1 \quad \forall i = 1, \cdots, n$$

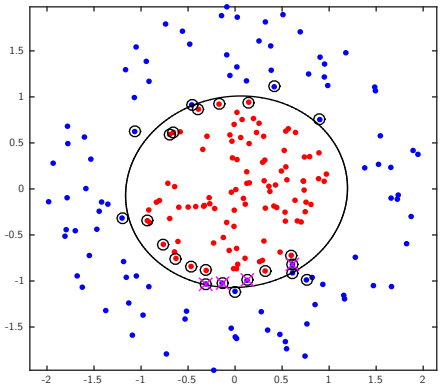
(vector in feature space —**may be infinite-dimensional**)

(inner product in feature space)



SVM in feature space

pullback

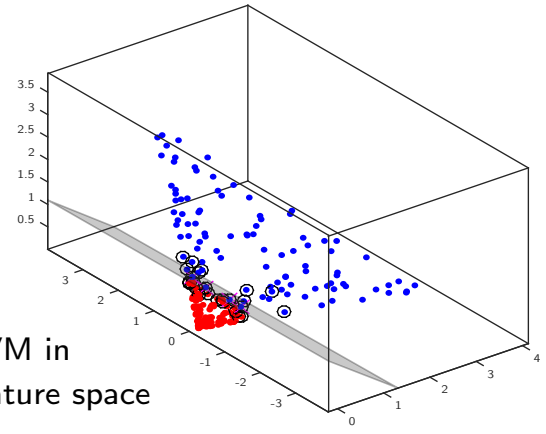


Kernel SVM

Quadratic program (hard margin / no slack):

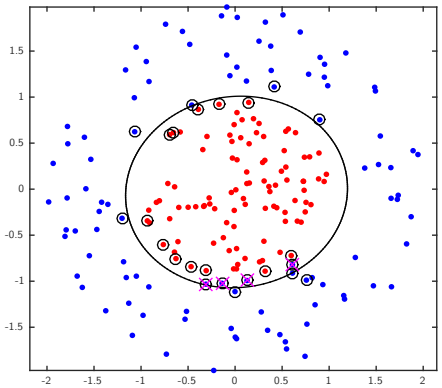
$$\underset{\alpha,\beta_0}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\, y_i\, k(x_i, x_j)\, y_j\, \alpha_j \quad \text{subj. to} \quad y_i\left(\sum_{j=1}^{n} \alpha_j\, y_j\, k(x_i, x_j) - \beta_0\right) \geq 1 \quad \forall i$$

Representer Thm $\implies \hat{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i\, \Phi(x_i) = \sum_{i=1}^{n} \alpha_i\, y_i\, k(x_i, \cdot)$



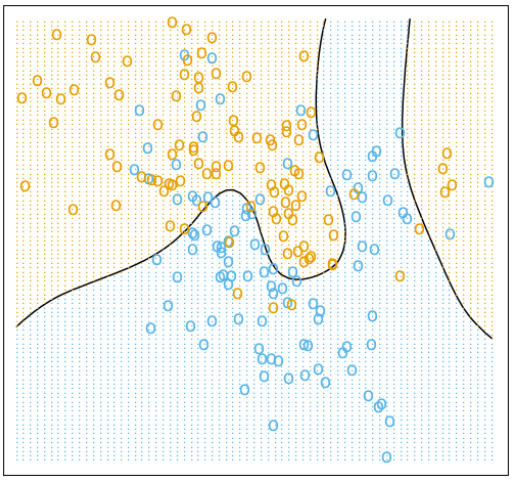
SVM in feature space

pullback



Experimental results

$n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$

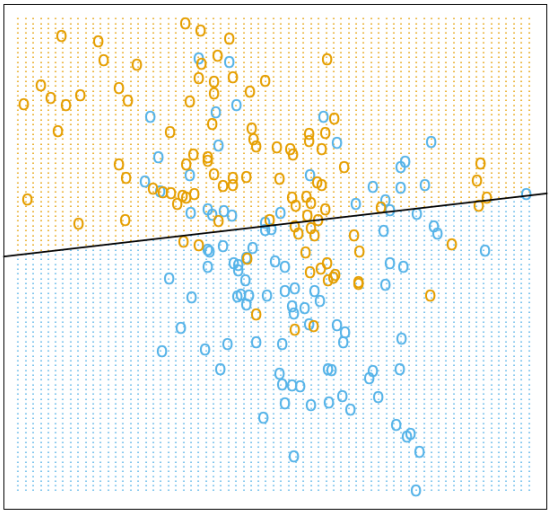


Bayes classifier

error rate $\approx 21\%$

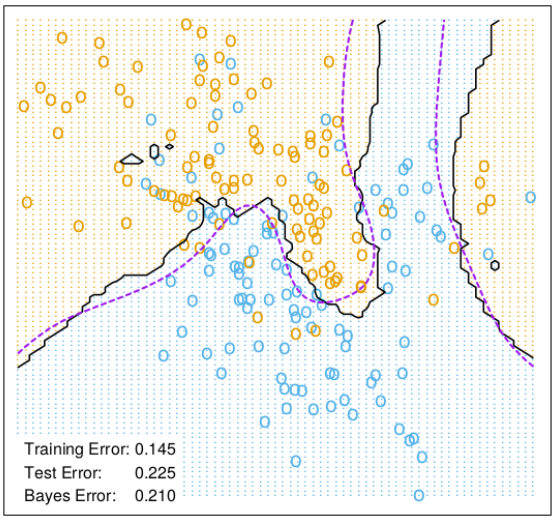
Experimental results

$n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



linear regression

error rate $\approx 34\%$

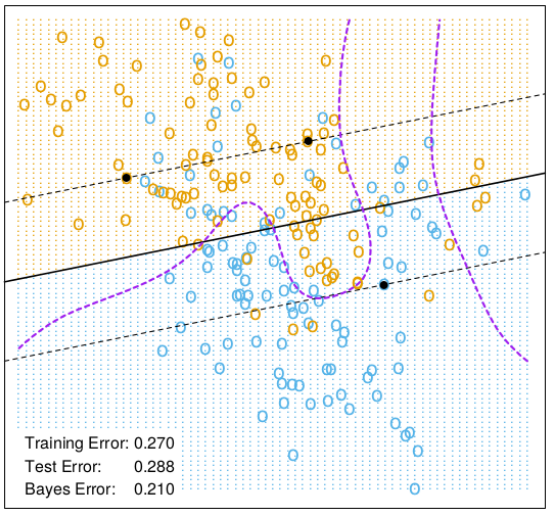


7-NN classifier

error rate $\approx 22.5\%$

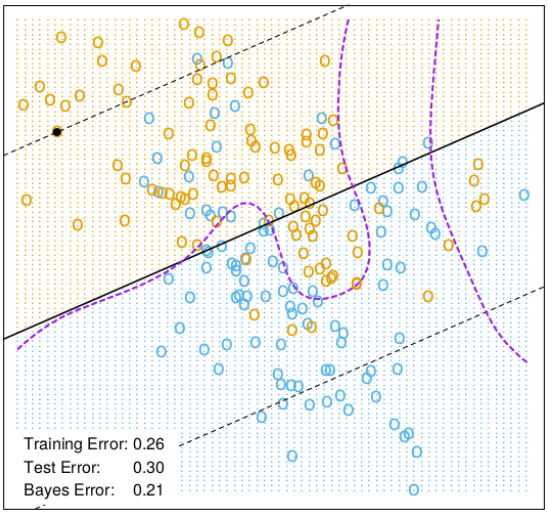
Experimental results

$n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



SVM with $\lambda = 10^{-2}$

error rate $\approx 28.8\%$

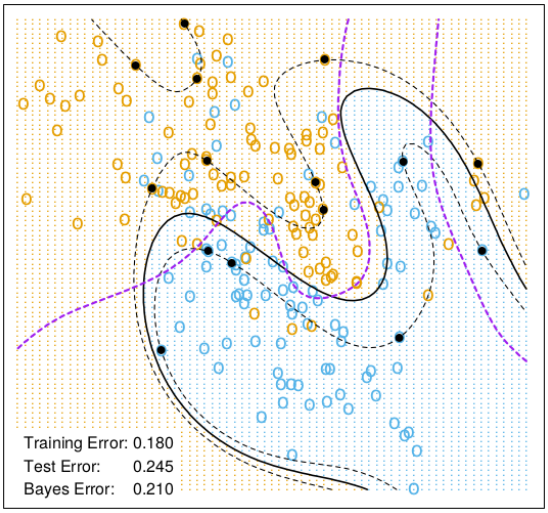


SVM with $\lambda = 10^{4}$

error rate $\approx 30\%$

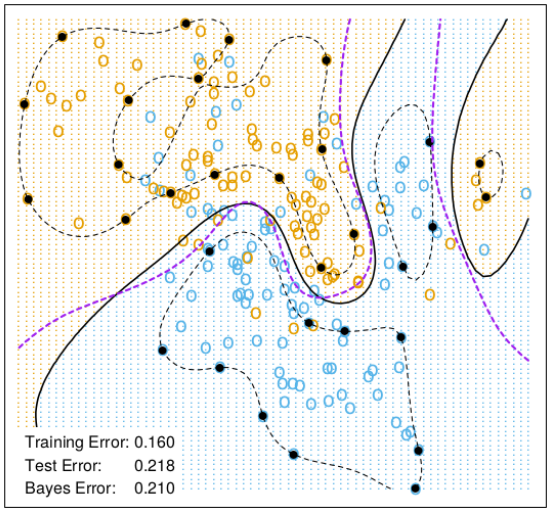
Experimental results

$n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



SVM with deg.-4 polynomial kernel

error rate $\approx 24.5\%$

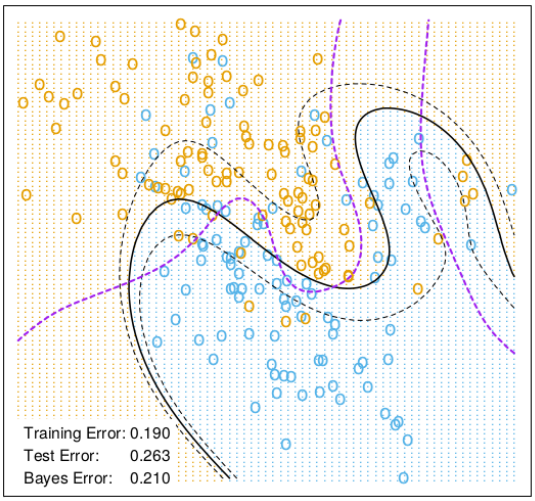


SVM with Gaussian kernel

error rate $\approx 21.8\%$

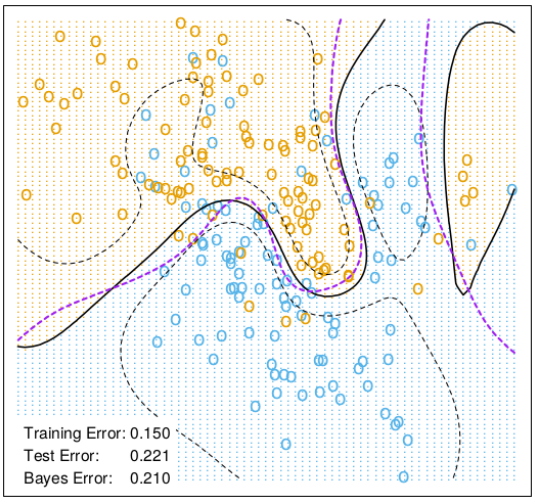
Experimental results

$n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



logistic reg. with deg.-4 polynomial kernel

error rate $\approx 26.3\%$



logistic reg. with Gaussian kernel

error rate $\approx 22.1\%$

What you should know

- Two types of linear approaches for classification
- Logistic regression:
  - generalized linear model
  - fitting by likelihood maximization & Newton-Raphson's method, convergence guarantees
  - degenerate cases & regularization
  - extension to multi-class
- Support Vector Machines (SVM):
  - margin maximization & quadratic programming problem
  - hinge loss, soft margin, relaxed quadratic programming problem, support vectors
  - extension to multi-class
  - kernel SVM

Artificial Neural Networks

Outline:

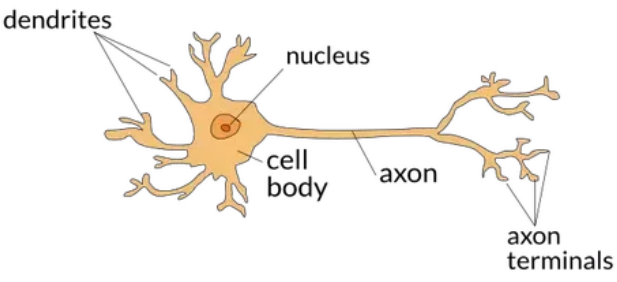
- **Connectionist machine learning: principles & historical landmarks**
- **Rosenblatt's perceptron algorithm**
- **Connectionist view on perceptron, multi-layer perceptron (MLP)**
- **Approximation power of MLP**
- **Training of MLP, gradient back-propagation in neural networks**
- **Regularization of neural networks**
- **Convolutional neural networks (CNN)**

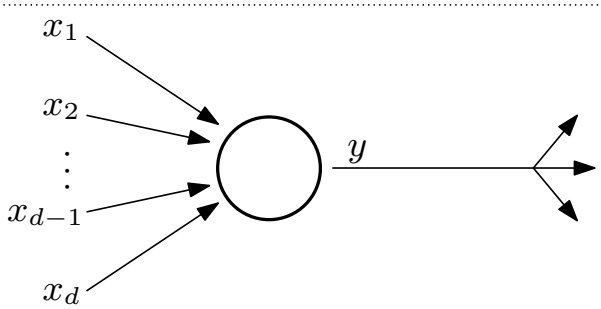
Outline

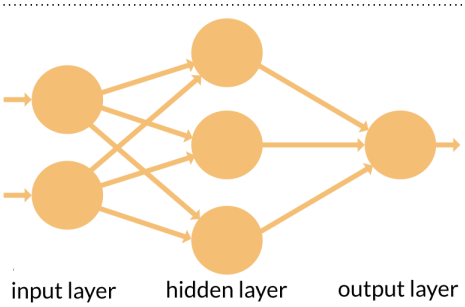
- **Connectionist machine learning: principles & historical landmarks**
- Rosenblatt's perceptron algorithm
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- Approximation power of MLP
- Training of MLP, gradient back-propagation in neural networks
- Regularization of neural networks
- Convolutional neural networks (CNN)

Connectionist approach to machine learning

- Modeled after the human brain
- small units (neurons) learn simple functions
- networks of these units can learn complicated functions

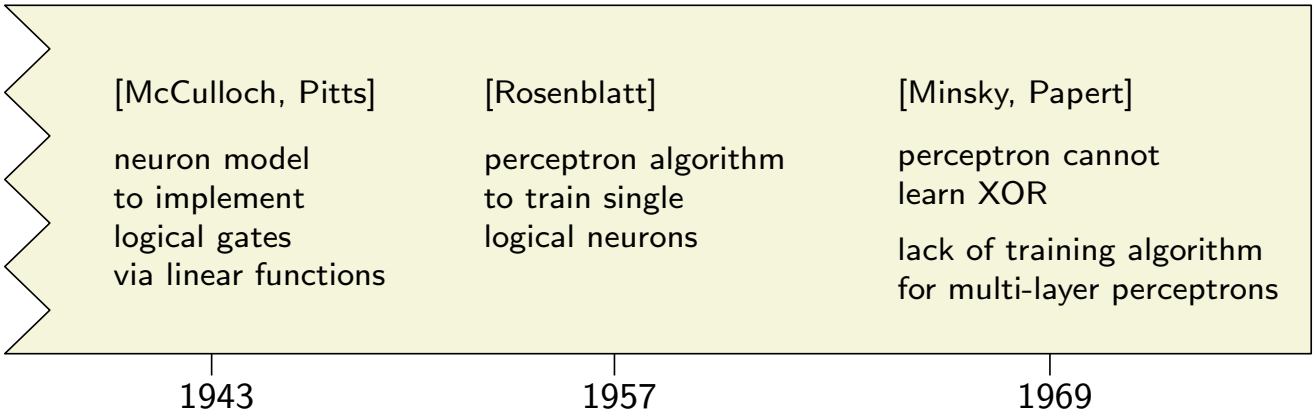






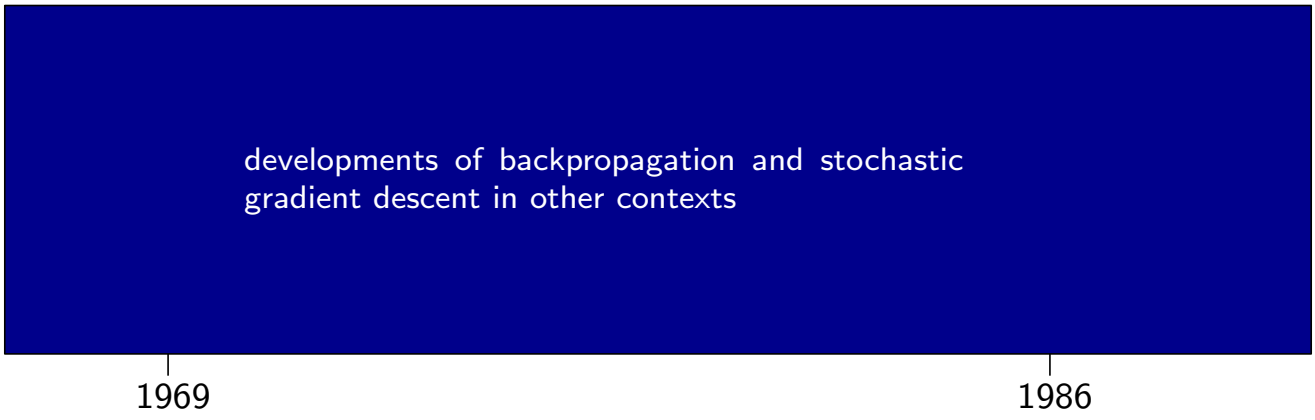
Historical landmarks

Birth of connectionist approach:



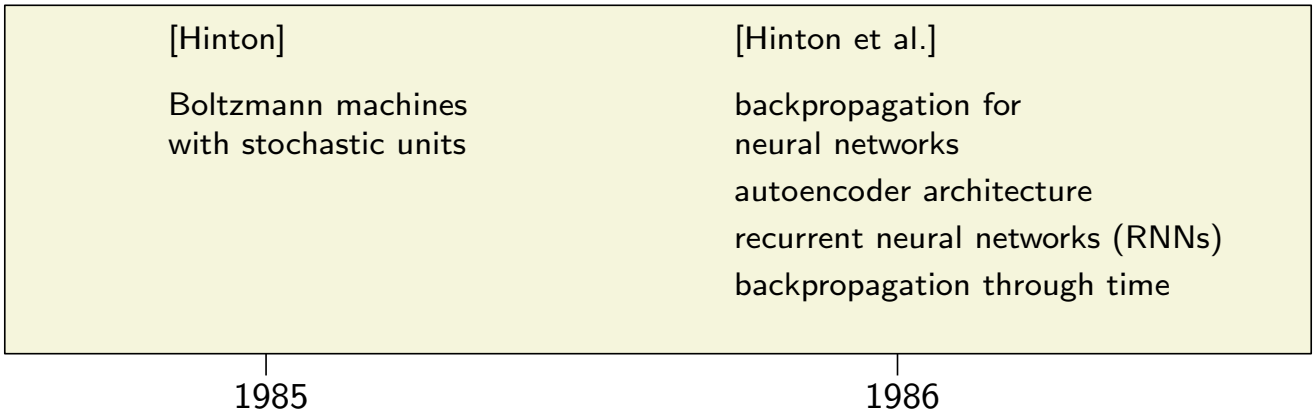
Historical landmarks

First AI winter:



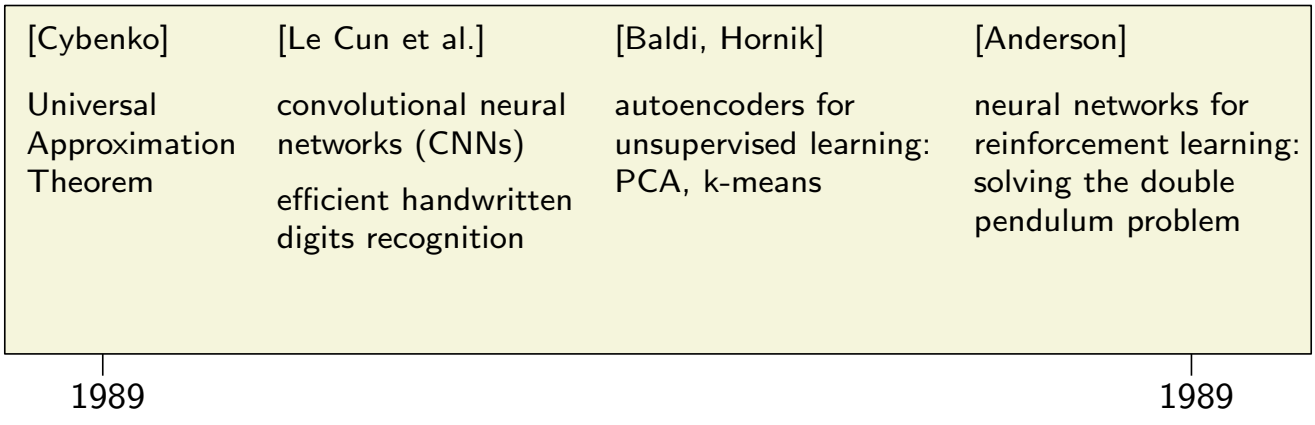
Historical landmarks

Rebirth and expansion:



Historical landmarks

Rebirth and expansion:



Historical landmarks

Rebirth and expansion:

[Siegelmann, Sontag]	[Lin]	[Bengio et al.]	[Schmidhuber, Hochreiter]
Universal Turing machine using RNNs	neural networks for reinforcement learning tasks in robotics	backpropagation ineffective on networks with 'many' hidden layers (e.g. RNNs) due to vanishing gradient issue	Long Short-Term Memory networks (LSTMs) suffering less than other RNNs from vanishing gradient
1991	1993	1994	1997

Historical landmarks

Second AI winter:

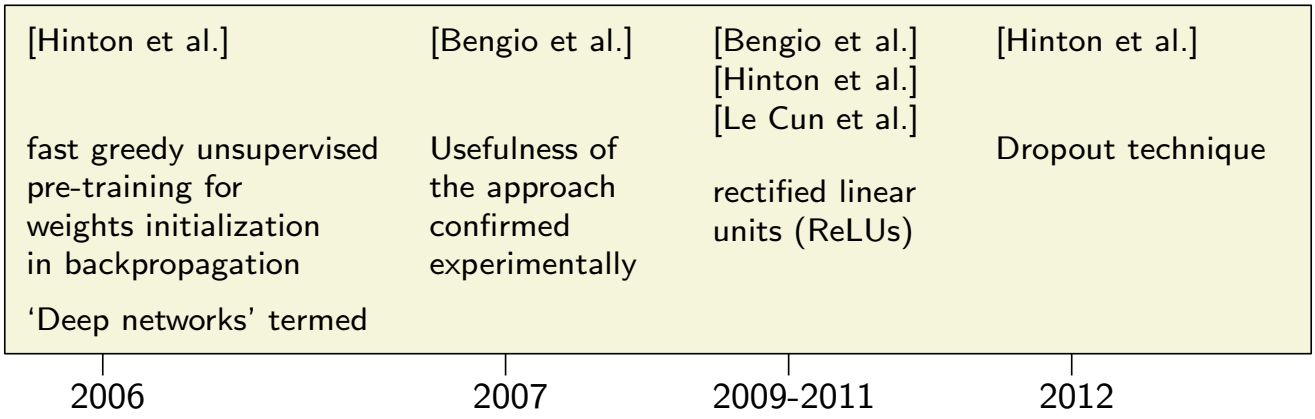
rapid development of promising statistical learning techniques: random forests, SVMs, kernels

the (now small) neural networks community takes refuge at the Canadian Institute for Advanced Research (CIFAR)

1998 — 2006

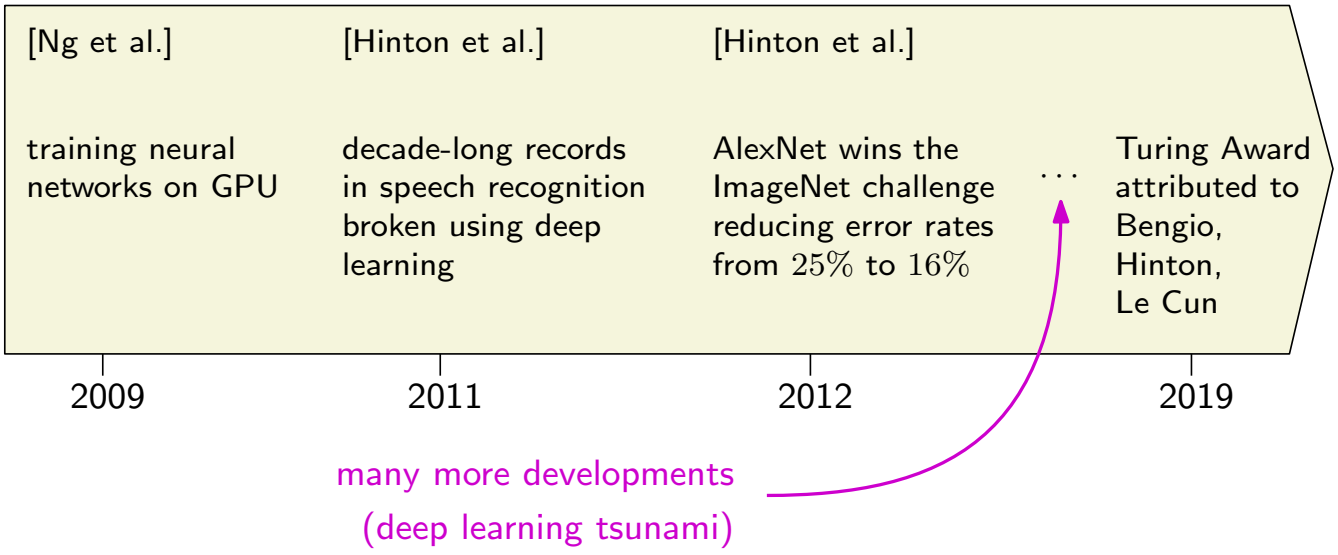
Historical landmarks

'Deep learning conspiracy':



Historical landmarks

Deep learning and big data:



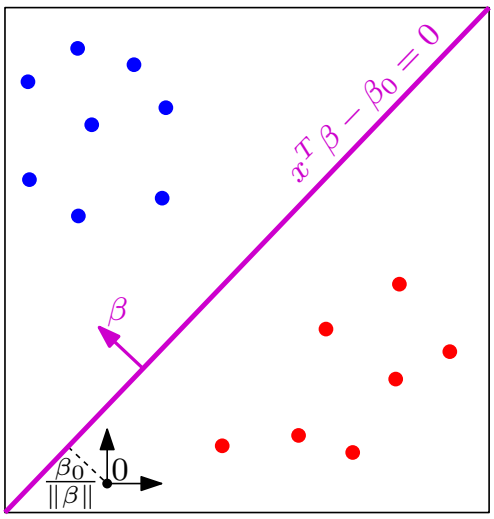
Outline

- Connectionist machine learning: principles & historical landmarks
- **Rosenblatt's perceptron algorithm**
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- Approximation power of MLP
- Training of MLP, gradient back-propagation in neural networks
- Regularization of neural networks
- Convolutional neural networks (CNN)

Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$

classifier: $y(x) := \text{sign}\left(\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta 0 \\ \beta \end{bmatrix}\right)$

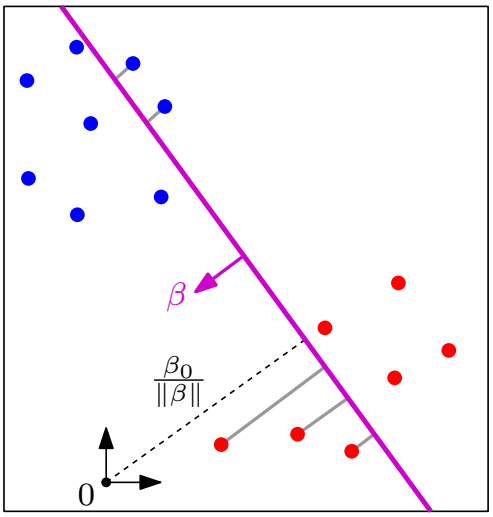


Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$
- fits the model by minimizing the distances of the misclassified points to the decision boundary:

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \sum_{\substack{x_i \text{ misclassified} \\ \text{at } \beta, \beta_0}} - y_i \underbrace{\begin{bmatrix} 1 & x_i \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix}}_{\|\beta\| \times \text{signed distance to hyperplane}}$$

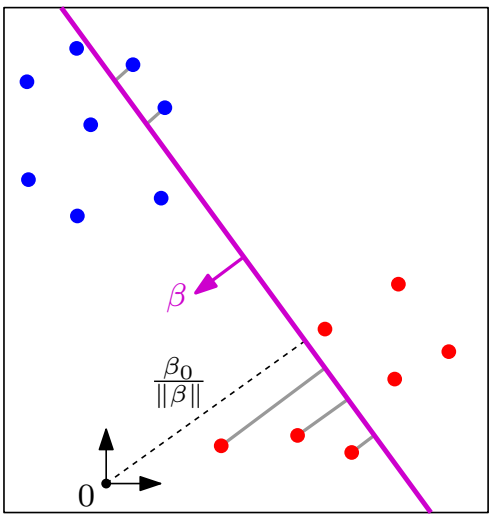
classifier: $\hat{y}(x) := \operatorname{sign}\left(\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\hat{\beta}_0 \\ \hat{\beta} \end{bmatrix} \right)$



Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$
- fits the model by minimizing the distances of the misclassified points to the decision boundary:

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \sum_{\substack{x_i \text{ misclassified} \\ \text{at } \beta, \beta_0}} - y_i \begin{bmatrix} 1 & x_i \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix}$$



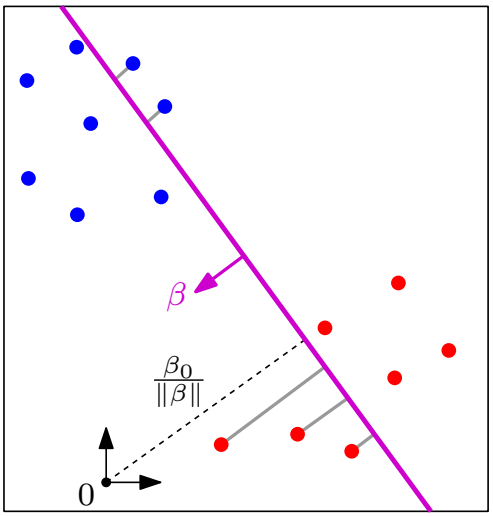
- piecewise linear functional optimization:

$$\begin{cases} \dfrac{\partial}{\partial \beta} = - \displaystyle\sum_{\substack{x_i \text{ misclassified} \\ \text{at } \beta, \beta_0}} y_i \, x_i \\ \dfrac{\partial}{\partial \beta_0} = \displaystyle\sum_{\substack{x_i \text{ misclassified} \\ \text{at } \beta, \beta_0}} y_i \end{cases}$$

Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$
- fits the model by minimizing the distances of the misclassified points to the decision boundary:

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \sum_{\substack{x_i \\ \text{misclassified} \\ \text{at } \beta, \beta_0}} - y_i \begin{bmatrix} 1 & x_i \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix}$$



- piecewise linear functional optimization:

INIT: set $\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix}$ at random

(gradient descent)

REPEAT:

compute misclassified set M

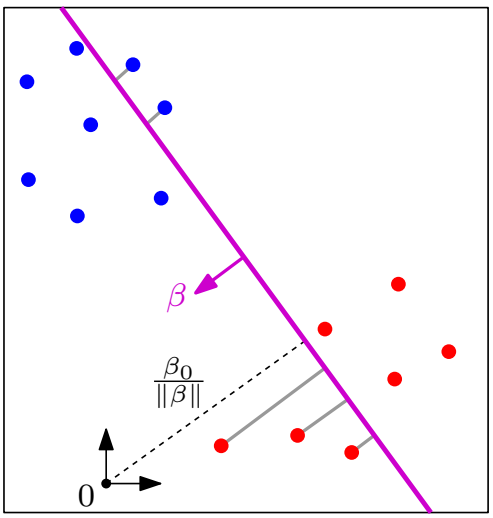
$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix} \longleftarrow \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix} + \varrho \sum_{x_i \in M} \begin{bmatrix} -y_i \\ y_i x_i \end{bmatrix}$

UNTIL CONVERGENCE `// requires convergence threshold`

Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$
- fits the model by minimizing the distances of the misclassified points to the decision boundary:

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \sum_{\substack{x_i \\ \text{misclassified} \\ \text{at } \beta, \beta_0}} - y_i \begin{bmatrix} 1 & x_i \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix}$$



- piecewise linear functional optimization:

INIT: set $\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix}$ at random

(stochastic)
(gradient descent)

REPEAT:

FOREACH $i = 1, \cdots, n$ DO

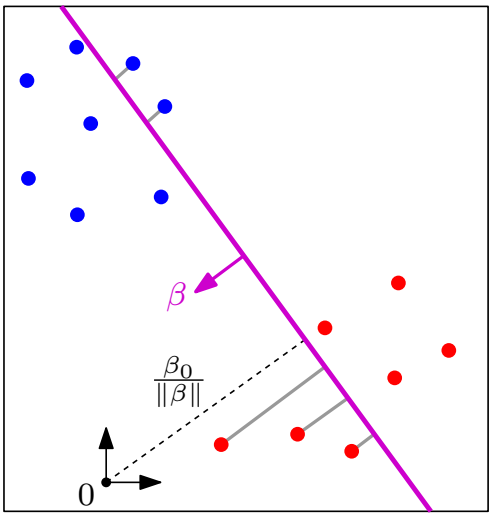
IF $y_i \left(x_i^T \hat{\beta} - \hat{\beta}_0 \right) < 0$ THEN $\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix} \longleftarrow \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix} + \varrho \begin{bmatrix} -y_i \\ y_i x_i \end{bmatrix}$ `// update` $\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{bmatrix}$ `and` M

UNTIL CONVERGENCE `// requires convergence threshold (in principle)`

Rosenblatt's perceptron algorithm

- designed for binary classification ($\mathcal{Y} = \{-1, 1\}$)
- models the separating hyperplane directly: $\begin{bmatrix} 1 & x \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix} = 0$
- fits the model by minimizing the distances of the misclassified points to the decision boundary:

$$\hat{\beta}, \hat{\beta}_0 := \underset{\beta, \beta_0}{\operatorname{argmin}} \sum_{\substack{x_i \text{ misclassified} \\ \text{at } \beta, \beta_0}} - y_i \begin{bmatrix} 1 & x_i \end{bmatrix}^T \begin{bmatrix} -\beta_0 \\ \beta \end{bmatrix}$$



Thm: [Rosenblatt 1960] [Novikoff 1962]

- If the two classes are linearly separable, then stochastic gradient descent with $\varrho = 1$ makes the energy converge to 0 in finitely many steps.
- More precisely, if $\exists$ separating hyperplane with margin γ and if $\|x_i\| \leq R \ \forall i = 1, \cdots, n$, then convergence occurs after $O(R^2/\gamma^2)$ steps.

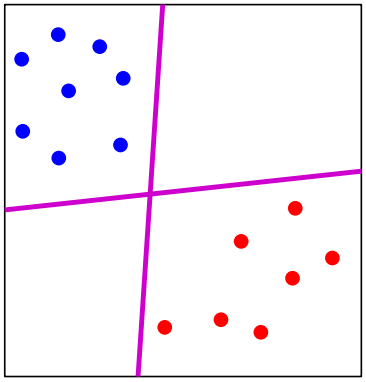
Rosenblatt's perceptron algorithm

Advantages:

- linear structure of energy permits the use of **stochastic gradient descent**
- stochastic gradient descent **scales up well** and **allows for re-training**

Drawbacks:

- no unique solution $\rightsquigarrow$ solution depends on initialization
- small margins lead to long convergence times
- algorithm behaves badly on non-separable classes:
  - convergence to irrelevant configurations
  - cyclic behavior (with potentially long cycles)

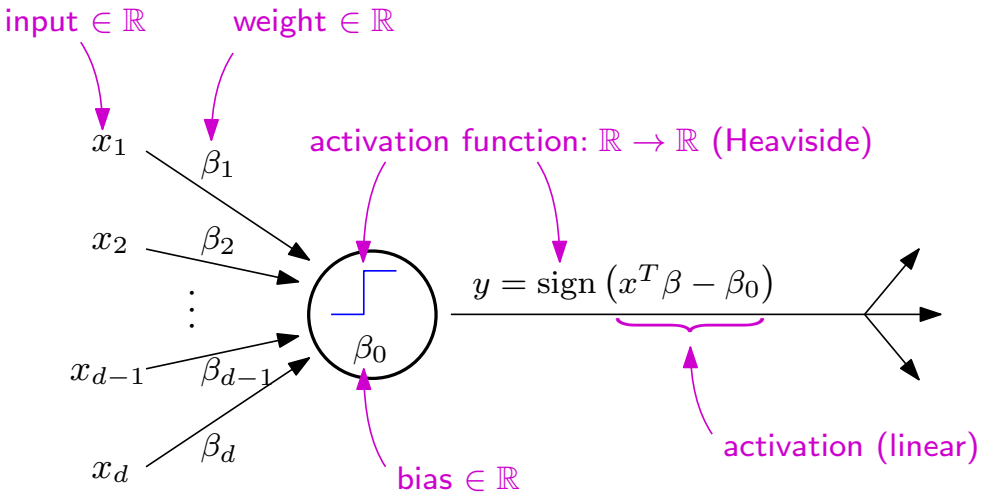


Outline

- Connectionist machine learning: principles & historical landmarks
- Rosenblatt's perceptron algorithm
- **Connectionist view on perceptron, multi-layer perceptron (MLP)**
- Approximation power of MLP
- Training of MLP, gradient back-propagation in neural networks
- Regularization of neural networks
- Convolutional neural networks (CNN)

Connectionist viewpoint on the perceptron

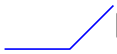
Perceptron neuron:



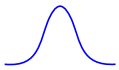
Connectionist viewpoint on the perceptron

- In practice, **smooth or piecewise smooth activation functions** are used:

identity: $t \mapsto t$ $\rightsquigarrow$ neuron implements linear regression



ReLU: $t \mapsto \max\{0, t\}$ $\rightsquigarrow$ simpler gradient expressions, faster training
helps address the vanishing gradient effect



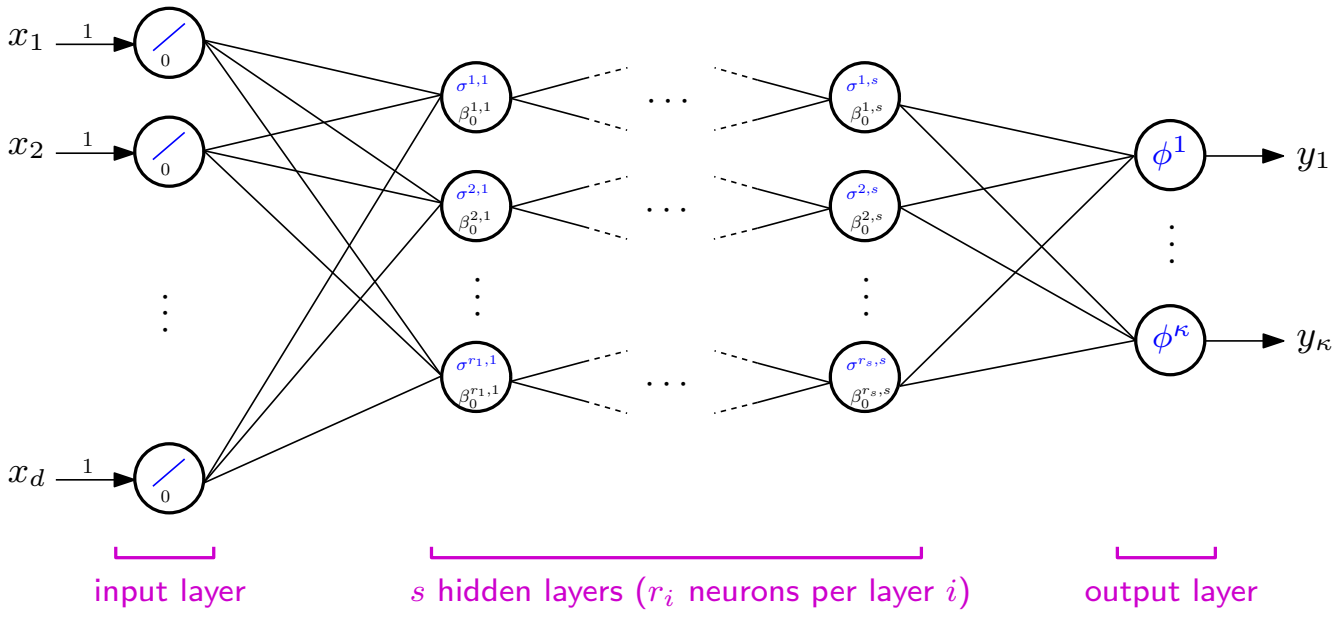
Gaussian: $t \mapsto \exp(-t^2)$ $\rightsquigarrow$ neuron implements linear regression with rbf

sigmoid: $t \mapsto \dfrac{1}{1+\exp(-t)}$ $\rightsquigarrow$ neuron implements logistic regression

softmax$_i$: $(x_1, \cdots x_d) \mapsto \dfrac{\exp(x_i)}{\sum_j \exp(x_j)}$ $\rightsquigarrow$ produces outputs in $[0, 1]$
outputs sum up to 1

Multi-layer perceptron (MLP)

Feedforward, **full connectivity** between consecutive layers:

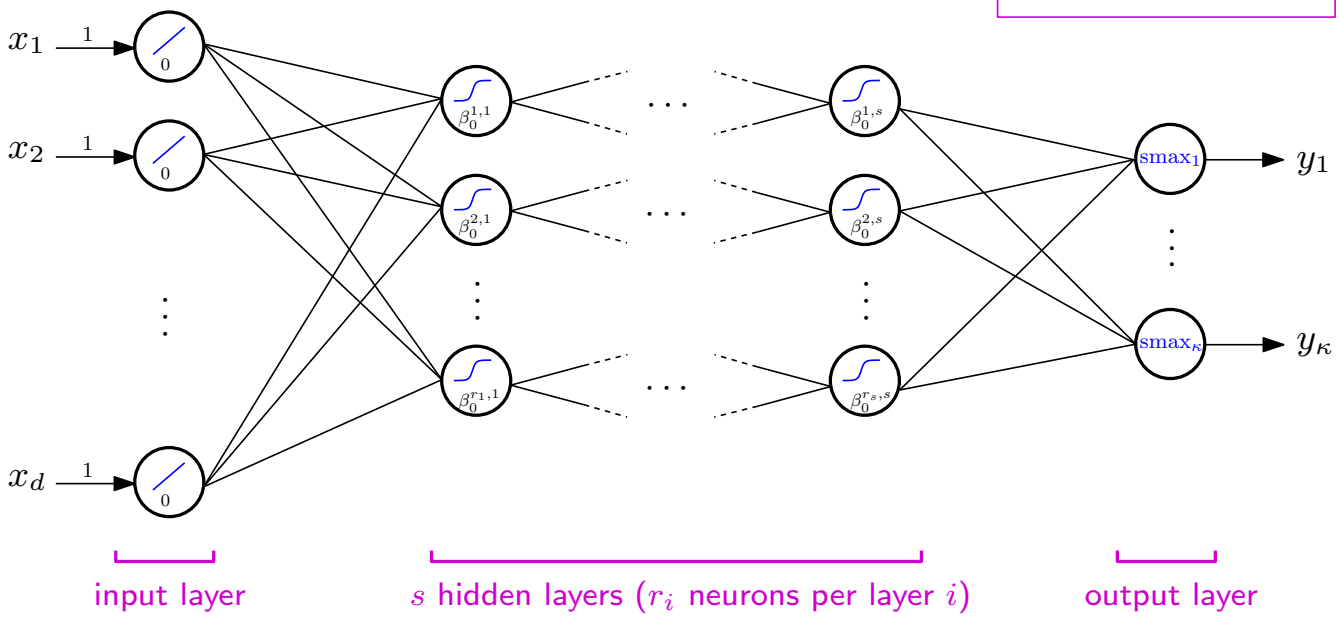


feed forward: evaluate layer per layer in a forward pass

Multi-layer perceptron (MLP)

Feedforward, **full connectivity** between consecutive layers:

vanilla MLP for κ-class classification



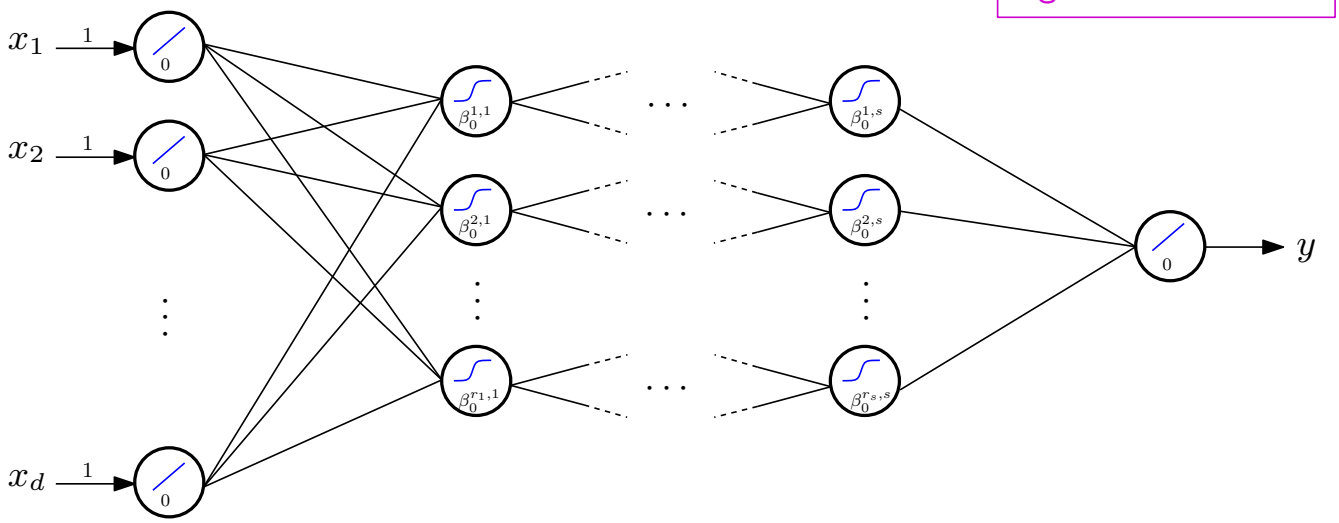
input layer — s hidden layers (r_i neurons per layer i) — output layer

feed forward: evaluate layer per layer in a forward pass

Multi-layer perceptron (MLP)

Feedforward, **full connectivity** between consecutive layers:

vanilla MLP for regression



input layer — s hidden layers (r_i neurons per layer i) — output layer

feed forward: evaluate layer per layer in a forward pass

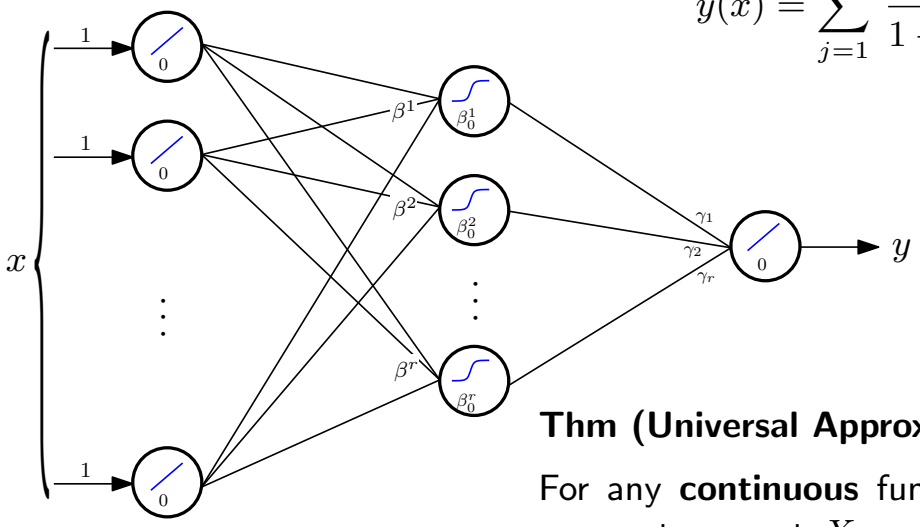
Outline

- Connectionist machine learning: principles & historical landmarks
- Rosenblatt's perceptron algorithm
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- **Approximation power of MLP**
- Training of MLP, gradient back-propagation in neural networks
- Regularization of neural networks
- Convolutional neural networks (CNN)

Approximation power

The 1-hidden layer case:

$$y(x) = \sum_{j=1}^{r} \frac{\gamma_j}{1 + \exp\left(\beta_0^j - x^T \beta^j\right)}$$



Thm (Universal Approximation): [Cybenko 1989]

For any **continuous** function $f : \mathbb{R}^d \to \mathbb{R}$ with compact support X, and any $\varepsilon > 0$, there exist a hidden layer size $r \in \mathbb{N}$ and parameter values $\beta_0^j, \beta^j, \gamma_j$ for $1 \leq j \leq r$ such that

$$|f(x) - y(x)| \leq \varepsilon \quad \forall x \in X.$$

Outline

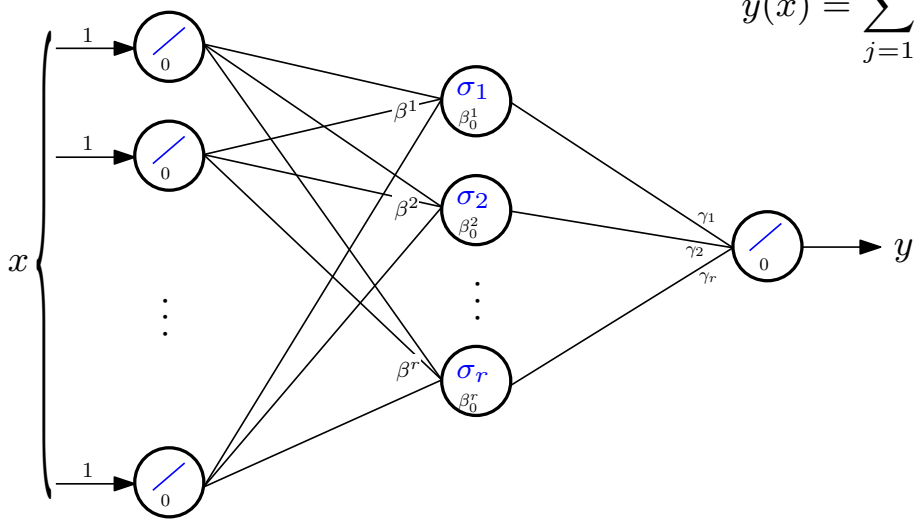
- Connectionist machine learning: principles & historical landmarks
- Rosenblatt's perceptron algorithm
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- Approximation power of MLP
- **Training of MLP, gradient back-propagation in neural networks**
- Regularization of neural networks
- Convolutional neural networks (CNN)

Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$

$$y(x) = \sum_{j=1}^{r} \gamma_j \, \sigma_j \left(x^T \beta^j - \beta_0^j \right)$$

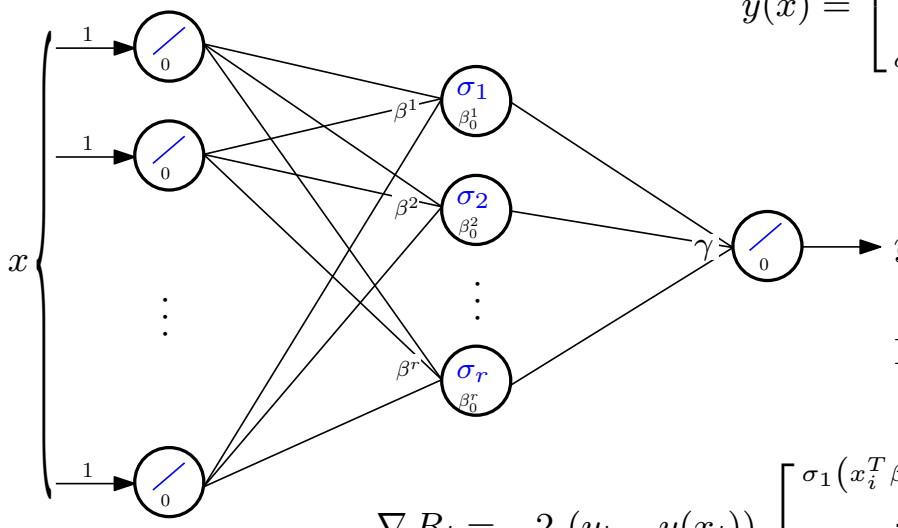
arbitrary activation function



Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$

$$y(x) = \begin{bmatrix} \sigma_1\left(x^T\beta^1 - \beta_0^1\right) \\ \vdots \\ \sigma_r\left(x^T\beta^r - \beta_0^r\right) \end{bmatrix}^T \gamma$$



$$\mathrm{RSS} = \sum_{i=1}^{n} \underbrace{(y_i - y(x_i))^2}_{R_i}$$

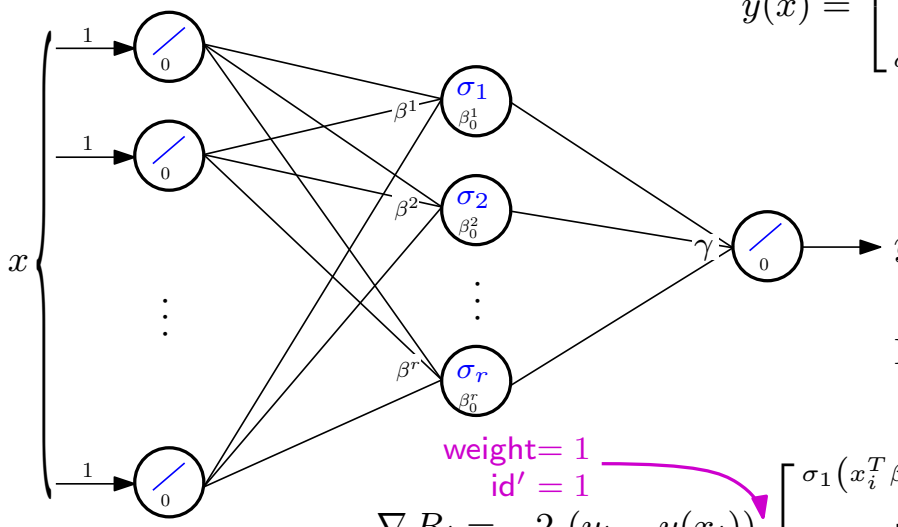
$$\nabla_\gamma R_i = \underbrace{-2\,(y_i - y(x_i))}_{\text{error at current neuron}} \begin{bmatrix} \sigma_1\left(x_i^T\beta^1 - \beta_0^1\right) \\ \vdots \\ \sigma_r\left(x_i^T\beta^r - \beta_0^r\right) \end{bmatrix} \leftarrow \text{impulse from previous layer}$$

$$\nabla_{\beta^j} R_i = -2\,(y_i - y(x_i))\,\gamma_j\,\sigma_j'\left(x_i^T\beta^j - \beta_0^j\right) x_i$$

Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$

$$y(x) = \begin{bmatrix} \sigma_1\left(x^T\beta^1 - \beta_0^1\right) \\ \vdots \\ \sigma_r\left(x^T\beta^r - \beta_0^r\right) \end{bmatrix}^T \gamma$$



$$\mathrm{RSS} = \sum_{i=1}^{n} \underbrace{(y_i - y(x_i))^2}_{R_i}$$

$$\nabla_\gamma R_i = \underbrace{-2\,(y_i - y(x_i))}_{\text{back-propagated error}} \begin{bmatrix} \sigma_1\left(x_i^T\beta^1 - \beta_0^1\right) \\ \vdots \\ \sigma_r\left(x_i^T\beta^r - \beta_0^r\right) \end{bmatrix}$$

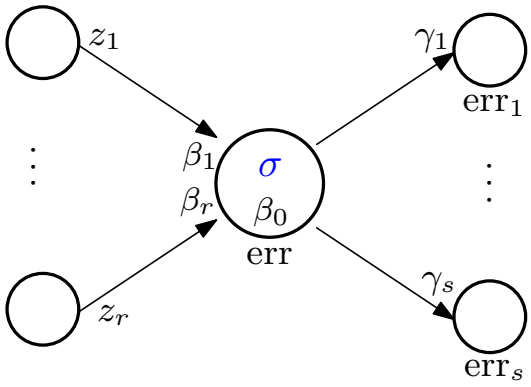
weight$= 1$, id$' = 1$

$$\nabla_{\beta^j} R_i = -2\,(y_i - y(x_i))\,\gamma_j\,\sigma_j'\left(x_i^T\beta^j - \beta_0^j\right) x_i$$

γ_j: weight of j-th hidden neuron in output neuron

Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$



gradient at each neuron:

$$\nabla_\beta R_i = \text{err} \cdot z$$

back-propagation equation:

$$\text{err} = \left(\sum_{j=1}^{s} \gamma_j \cdot \text{err}_j \right) \sigma' \left(z^T \beta - \beta_0 \right)$$

Forward-backward procedure for each (x_i, y_i):

- forward: compute activations & impulses
- backward: back-propagate error and update

Training epoch (stoch. gradient pass):

- sweep through the training set
- apply forward-backward update for each (x_i, y_i)

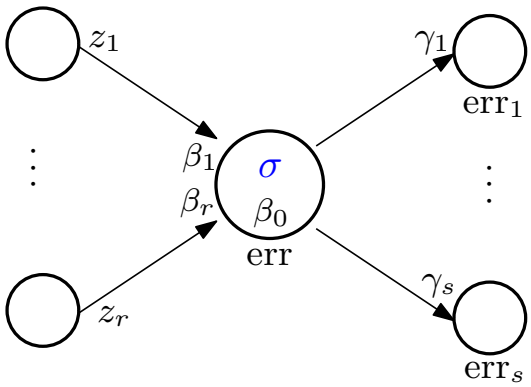
$$\begin{aligned} \beta &\longleftarrow \beta - \varrho \nabla_\beta R_i \\ \beta_0 &\longleftarrow \beta_0 - \varrho \nabla_{\beta_0} R_i \end{aligned}$$

learning rate

- **Guarantee:** computes the correct gradient step

Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$



gradient at each neuron:

$$\nabla_\beta R_i = \text{err} \cdot z$$

back-propagation equation:

$$\text{err} = \left(\sum_{j=1}^{s} \gamma_j \cdot \text{err}_j \right) \sigma' \left(z^T \beta - \beta_0 \right)$$

Forward-backward procedure for each (x_i, y_i):

- forward: compute activations & impulses
- backward: back-propagate error and update

Online learning:

- perform multiple training epochs
- update (reduce) ϱ between epochs

until cvgence or early stop

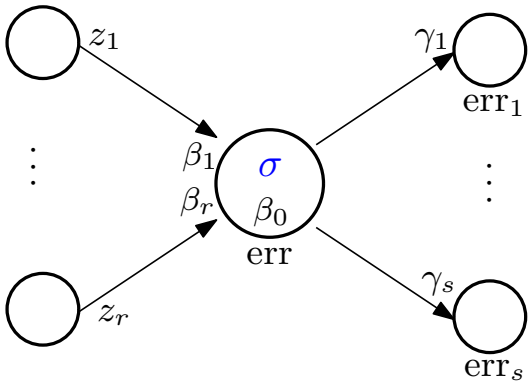
$$\begin{aligned} \beta &\longleftarrow \beta - \varrho \nabla_\beta R_i \\ \beta_0 &\longleftarrow \beta_0 - \varrho \nabla_{\beta_0} R_i \end{aligned}$$

learning rate

- **Guarantee:** cvges to a local min. under proper reduction scheme for ϱ

Training

Input: $(x_1, y_1), \cdots, (x_n, y_n)$



Online learning:

- scales up well
- can handle new training data

gradient at each neuron:

$$\nabla_\beta R_i = \text{err} \cdot z$$

back-propagation equation:

$$\text{err} = \left(\sum_{j=1}^{s} \gamma_j \cdot \text{err}_j \right) \sigma' \left(z^T \beta - \beta_0 \right)$$

Forward-backward procedure for each (x_i, y_i):

- forward: compute activations & impulses
- backward: back-propagate error and update

$$\beta \longleftarrow \beta - \varrho \nabla_\beta R_i$$
$$\beta_0 \longleftarrow \beta_0 - \varrho \nabla_{\beta_0} R_i$$

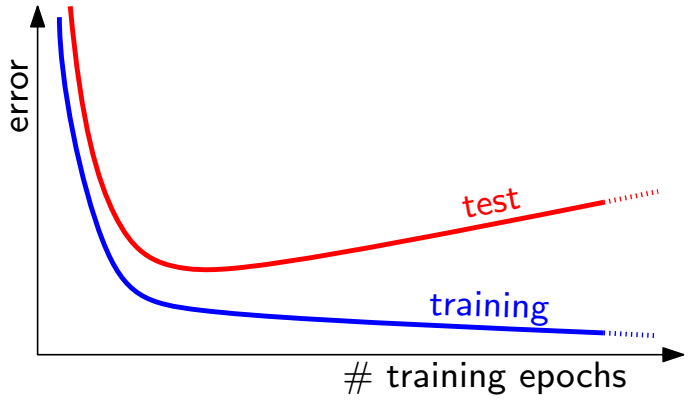
learning rate

Outline

- Connectionist machine learning: principles & historical landmarks
- Rosenblatt's perceptron algorithm
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- Approximation power of MLP
- Training of MLP, gradient back-propagation in neural networks
- **Regularization of neural networks**
- Convolutional neural networks (CNN)

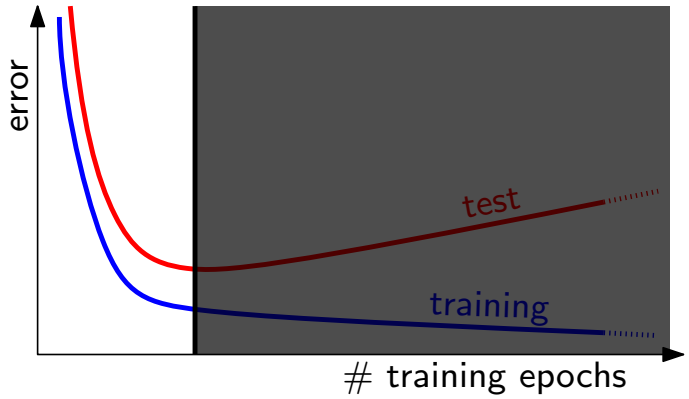
Regularization

The high number of parameters in neural networks usually leads to **overfitting.**



Regularization

The high number of parameters in neural networks usually leads to **overfitting.**

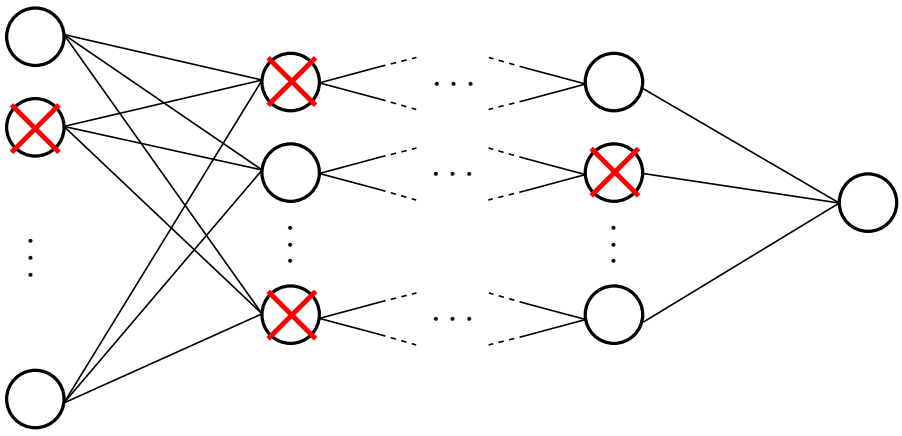


Approach 1: **early stop**:

- stop gradient descent after k epochs
- select k by cross-validation

Regularization

The high number of parameters in neural networks usually leads to **overfitting.**

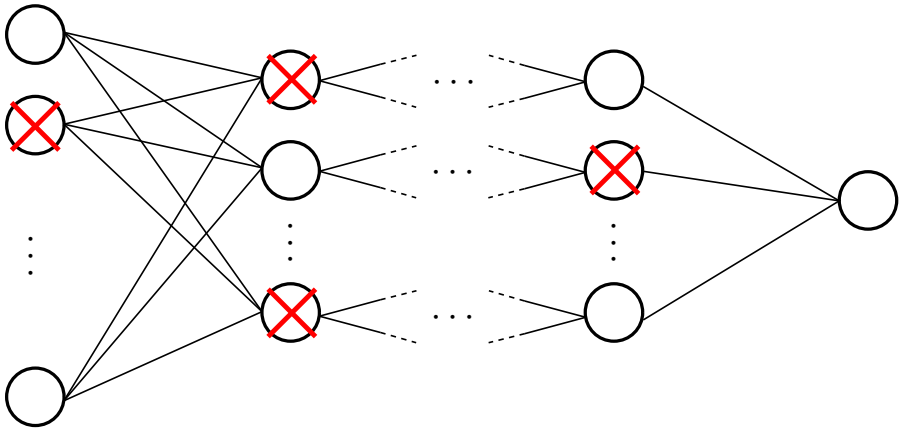


Approach 2: **dropout**:

- at each training epoch, randomly switch off a fraction of neurons in each layer
- replaces the full model by a series of random simplified models
- select fraction of switched-off neurons by cross-validation

Regularization

The high number of parameters in neural networks usually leads to **overfitting.**

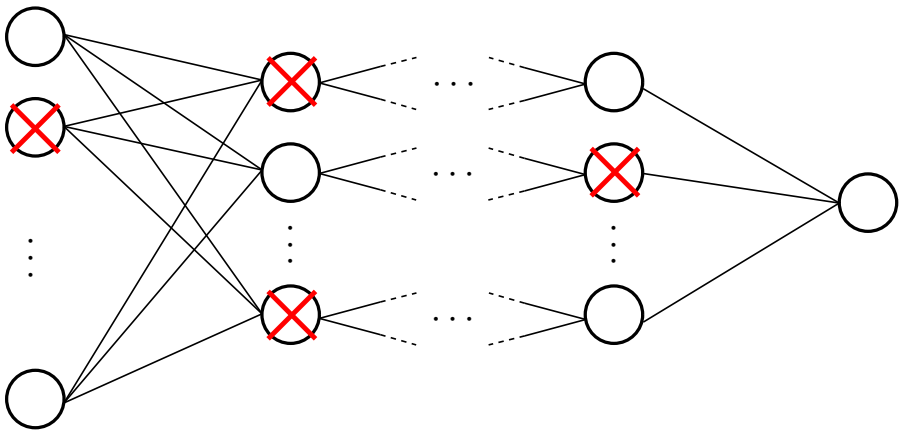


Approach 3: **weight decay**:

- penalize ℓ^2-norm of the parameter vector: $\mathrm{RSS} + \lambda \sum_{\text{neuron } j} (\beta_0^j)^2 + \|\beta^j\|_2^2$
- adds a term $2\lambda\,\beta^j$ to the gradient $\nabla_{\beta^j} R_i$
- λ selected by cross-validation

Regularization

The high number of parameters in neural networks usually leads to **overfitting.**

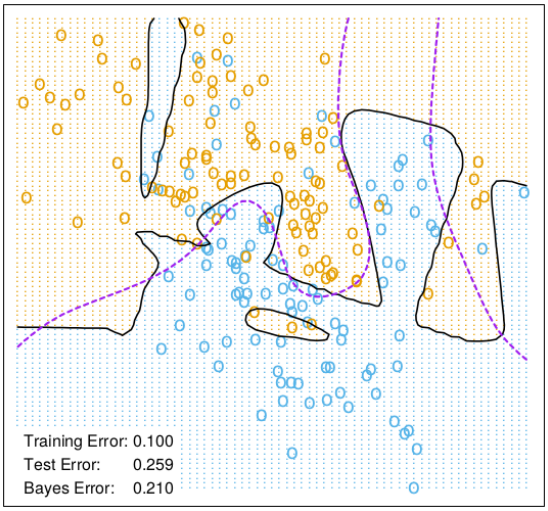


Approach 3': **weight elimination**:

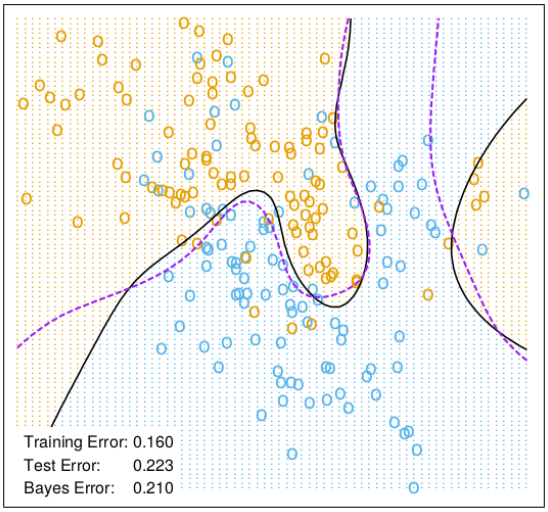
- penalize ℓ^2-norm of the parameter vector: $\mathrm{RSS} + \lambda \sum_{\text{neuron } j} \dfrac{(\beta_0^j)^2 + \|\beta^j\|_2^2}{1 + (\beta_0^j)^2 + \|\beta^j\|_2^2}$
- shrinks smaller weights more drastically
- λ selected by cross-validation

Regularization

Example: $n = 100 + 100$ (mixture of 2 Gaussians), $d = 2$



10 hidden units
err. rate $\approx 25.9\%$



10 hidden units + weight decay ($\lambda = 0.02$)
err. rate $\approx 22.3\%$

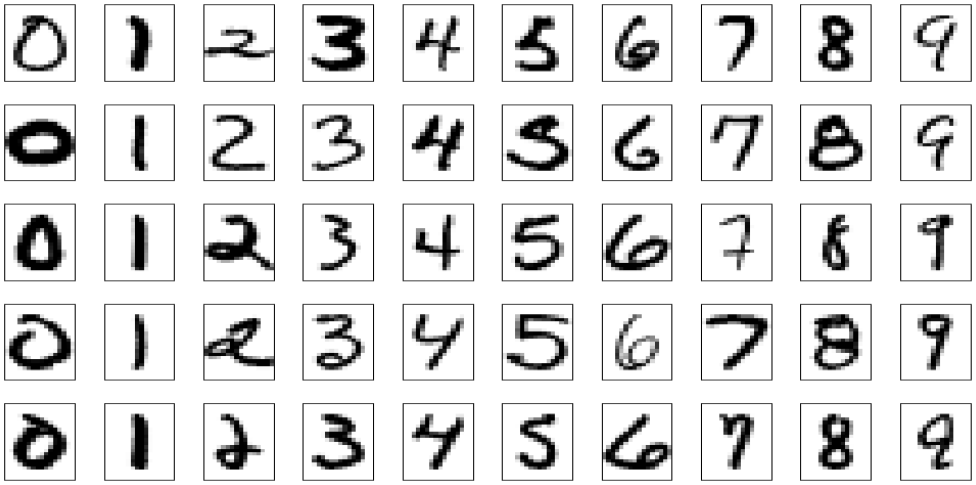
predictor	Bayes classif.	7-NN	Gauss. ker logistic reg.	Gauss. ker SVM
error rate	21%	22.5%	22.1%	21.8%

Outline

- Connectionist machine learning: principles & historical landmarks
- Rosenblatt's perceptron algorithm
- Connectionist view on perceptron, multi-layer perceptron (MLP)
- Approximation power of MLP
- Training of MLP, gradient back-propagation in neural networks
- Regularization of neural networks
- **Convolutional neural networks (CNN)**

From MLP to convolutional networks

ZIP code dataset:

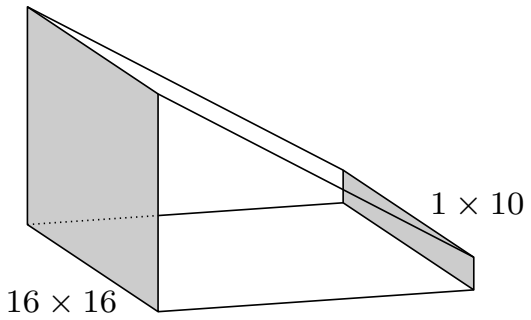


- 16×16 binary patches extracted from handwritten digits on envelopes in US
- dataset down-sized to 320 images for training and 160 for testing

From MLP to convolutional networks

Proposed networks [Le Cun 1989]:

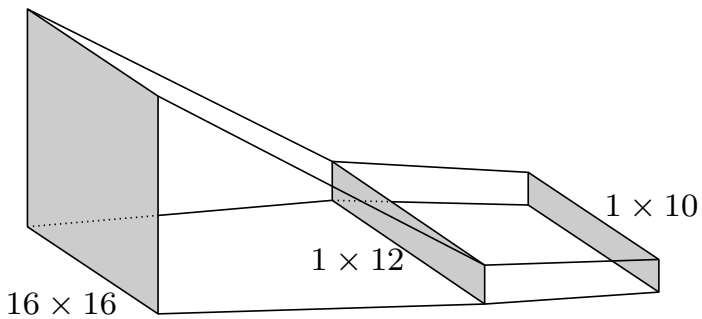
- Net-1: no hidden layer, full connectivity with sigmoid output units



From MLP to convolutional networks

Proposed networks [Le Cun 1989]:

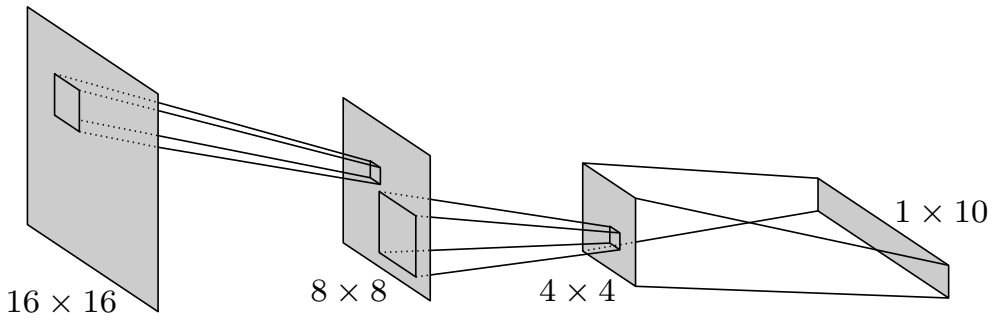
- Net-1: no hidden layer, full connectivity with sigmoid output units
- Net-2: 1 hidden layer with 12 sigmoid units, full connectivity



From MLP to convolutional networks

Proposed networks [Le Cun 1989]:

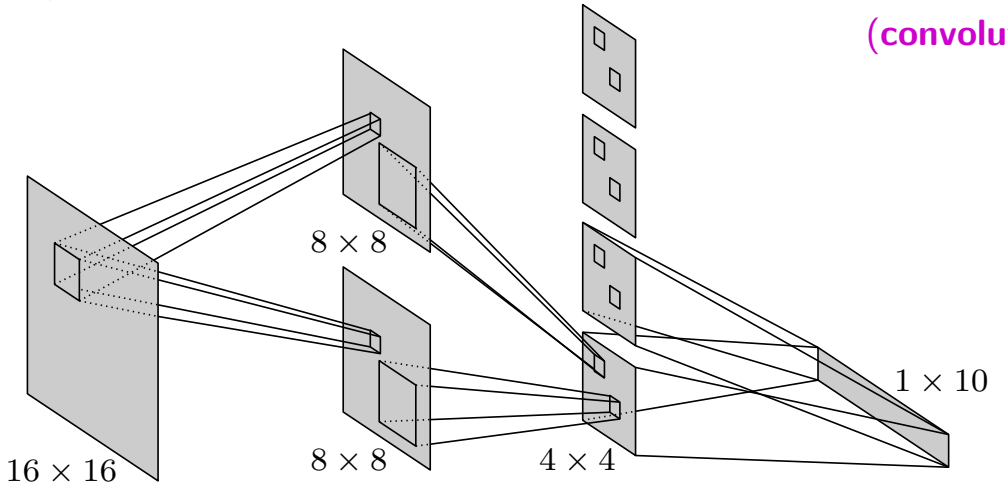
- Net-1: no hidden layer, full connectivity with sigmoid output units
- Net-2: 1 hidden layer with 12 sigmoid units, full connectivity
- Net-3: 2 hidden layers, **local connectivity**
  - 3×3 patches for first hidden layer
  - 5×5 patches for second hidden layer
  - full connectivity at output layer



From MLP to convolutional networks

Proposed networks [Le Cun 1989]:

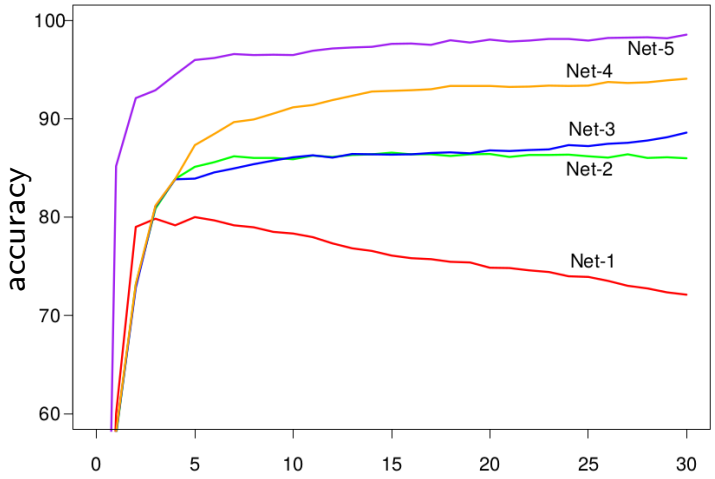
- Net-1: no hidden layer, full connectivity with sigmoid output units
- Net-2: 1 hidden layer with 12 sigmoid units, full connectivity
- Net-3: 2 hidden layers, **local connectivity**
- Net-4/5: 2 hidden layers, local connectivity, **shared weights** within each frame (**convolutional layer**)



From MLP to convolutional networks

Proposed networks [Le Cun 1989]:

- Net-1: no hidden layer, full connectivity with sigmoid output units
- Net-2: 1 hidden layer with 12 sigmoid units, full connectivity
- Net-3: 2 hidden layers, **local connectivity**
- Net-4/5: 2 hidden layers, local connectivity, **shared weights** within each frame (**convolutional layer**)



network	accuracy
Net-1	80%
Net-2	87%
Net-3	88.5%
Net-4	94%
Net-5	98.4%

What you should know

- Concepts: artificial neuron & neural network, deep learning, AI winter
- Rosenblatt's perceptron: linear model, stochastic gradient descent, Rosenblatt-Novikoff thm, limitations
- Multi-layer perceptron (MLP): architecture, universal approximation thm
- Gradient back-propagation: general formula, special case of MLP
- Regularization: early stop, dropout, weight decay & elimination
- Convolutional neural network: principles

Feature Extraction

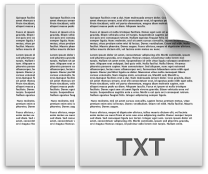
Outline:

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

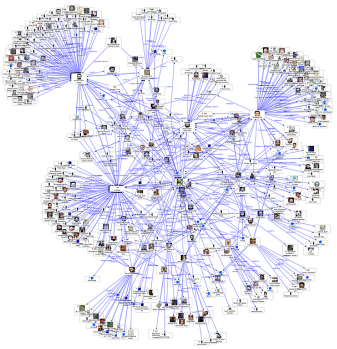
Outline

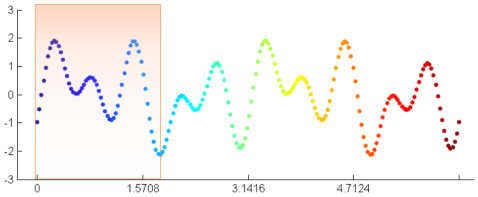
- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

Feature extraction in machine learning



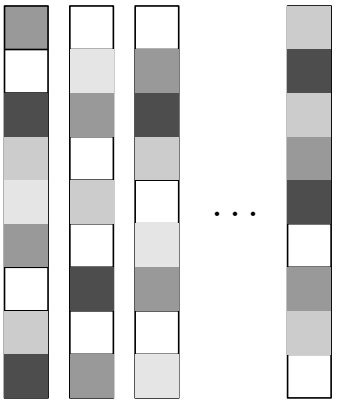
Data





feature design
or learning

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



Build features for

- ML on non-vectorial data
- improved learning rates

Feature extraction in machine learning

Input: data space D (can be $\mathbb{R}^d$, space of graphs, of 3d shapes, etc.)

Goal: find a mapping $\Phi : D \to \mathbb{R}^k$ (vectorization) or $\Phi : D \to \mathcal{H}$ (kernel) that:

- is **meaningful**, preserving relationships (e.g. proximity) between data in D
- extracts structural information that is **useful for learning** tasks
- can be inverted (**pre-image** problem)

2 classes of approaches:

- **feature engineering**: new representation is designed using expert knowledge
- **feature learning**: new representation is learnt from the data

Feature extraction in machine learning

Input: data space D (can be $\mathbb{R}^d$, space of graphs, of 3d shapes, etc.)

Goal: find a mapping $\Phi : D \to \mathbb{R}^k$ (vectorization) or $\Phi : D \to \mathcal{H}$ (kernel) that:

- is **meaningful**, preserving relationships (e.g. proximity) between data in D
- extracts structural information that is **useful for learning** tasks
- can be inverted (**pre-image** problem)

An old, vast, rich, and (still) hot topic:

- area-specific
- deserves an entire course
- overview: 1-2 approaches per data type (cf. specialized 3A courses)

Outline

- Principles of feature extraction
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- Curse of dimensionality and dimensionality reduction
- Vector quantization
- Principal component analysis (PCA)

Text features

Input: text data on a fixed dictionary $\mathcal{T}$ (e.g. English, French, scientific)

Bag-of-words model:

- distribution (histogram) $X \in \mathbb{N}^{\mathcal{T}}$ of word occurrences in the text
- normalized distribution $X' \in \mathbb{R}^{\mathcal{T}}$ to handle texts of different sizes
- "bag" means that the order of words is ignored
- remove stop-words
- use base forms of words (lemmatization)

Example:

"Humans come down from the apes, the ape comes down from the tree."

ape: ~~1~~ 2 ~~apes: 1~~	come: ~~1~~ 2 ~~comes: 1~~	down: 2 from: 2	humans: 1 ~~the: 3~~	tree: 1

Text features

Input: text data on a fixed dictionary $\mathcal{T}$ (e.g. English, French, scientific)

Bag-of-words model:

- distribution (histogram) $X \in \mathbb{N}^{\mathcal{T}}$ of word occurrences in the text
- normalized distribution $X' \in \mathbb{R}^{\mathcal{T}}$ to handle texts of different sizes
- "bag" means that the order of words is ignored
- remove stop-words
- use base forms of words (lemmatization)

Typical application: spam filtering (email $\equiv$ bag of keywords)

Generalization: n-gram model (sequences of words of length n)

Complement: word2vec (neural net trained to predict word from context BoW)

Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

Graph features

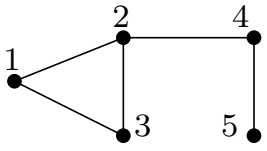
Input: graph (network) data, represented via adjacency matrices $A \in \mathbb{R}^{n \times n}$

- ▶ reprensentation not invariant to vertex relabeling

(unweighted: $A_{ij} \in \{0, 1\}$)

(undirected: $A = A^T$)

Example: $(n = 5)$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Graph features

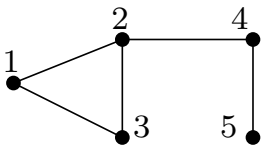
Input: graph (network) data, represented via adjacency matrices $A \in \mathbb{R}^{n \times n}$

Graphlets:

- dictionary $\mathcal{T}$ of unlabeled graphs of fixed vertex size k (graphlets)
- count the number of occurrences of each graphlet as an induced subgraph
- spectrum $X \in \mathbb{N}^{\mathcal{T}}$: distribution of occurrences (**bag-of-features** model)

Example: $(n = 5,\ k = 3)$

$X = (1, 3, 6, 0)$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Graph features

Input: graph (network) data, represented via adjacency matrices $A \in \mathbb{R}^{n \times n}$

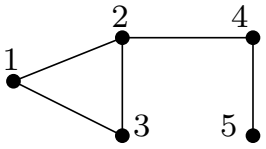
Graphlets:

- dictionary $\mathcal{T}$ of unlabeled graphs of fixed vertex size k (graphlets)
- count the number of occurrences of each graphlet as an induced subgraph
- spectrum $X \in \mathbb{N}^{\mathcal{T}}$: distribution of occurrences (**bag-of-features** model)

- normalized spectrum $X' \in \mathbb{R}^{\mathcal{T}}$ to handle graphs of different vertex sizes n

Example: $(n = 5,\ k = 3)$

$X' = (\frac{1}{10}, \frac{3}{10}, \frac{6}{10}, 0)$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Graph features

Input: graph (network) data, represented via adjacency matrices $A \in \mathbb{R}^{n \times n}$

Graphlets:

- dictionary $\mathcal{T}$ of unlabeled graphs of fixed vertex size k (graphlets)
- count the number of occurrences of each graphlet as an induced subgraph
- spectrum $X \in \mathbb{N}^{\mathcal{T}}$: distribution of occurrences (**bag-of-features** model)
- normalized spectrum $X' \in \mathbb{R}^{\mathcal{T}}$ to handle graphs of different vertex sizes n

Props:

- $G_1 \simeq G_2 \Rightarrow X'(G_1) = X'(G_2)$
- the converse holds for $n = k + 1 \leq 11$ but not in general

Graph features

Input: graph (network) data, represented via adjacency matrices $A \in \mathbb{R}^{n \times n}$

Graphlets:

- dictionary $\mathcal{T}$ of unlabeled graphs of fixed vertex size k (graphlets)
- count the number of occurrences of each graphlet as an induced subgraph
- spectrum $X \in \mathbb{N}^{\mathcal{T}}$: distribution of occurrences (**bag-of-features** model)
- normalized spectrum $X' \in \mathbb{R}^{\mathcal{T}}$ to handle graphs of different vertex sizes n

Computation:

- exhaustive enumeration of size-k subgraphs in size-n graph takes $O(n^k)$ time
- sample the space of graphlets to speed up the calculations in practice

Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

Image features

Input: Images via intensity maps $I : \mathbb{Z}^2 \to \mathbb{R}$, one for each color channel

- reprensentation not invariant to rigid transforms and rescaling
- local features designed to characterize the 'local shape' of the image
- used for salient point detection & matching
  (e.g. stereoscopic vision, panorama building)
- can be combined into global features for image comparison

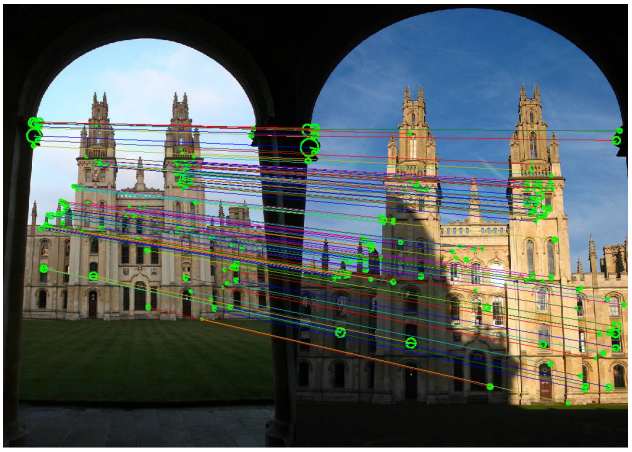


Image features

Input: Images via intensity maps $I : \mathbb{Z}^2 \to \mathbb{R}$, one for each color channel

SIFT (Scale-Invariant Feature Transform) at pixel $x \in \mathbb{Z}^2$:

a) **choose scale** σ:

- compute convolutions at x with Gaussian kernels of various bandwidths σ_i
- compute the differences between the convolutions at bandwidths σ_i, σ_{i+1}
- select scale(s) σ with maximum difference

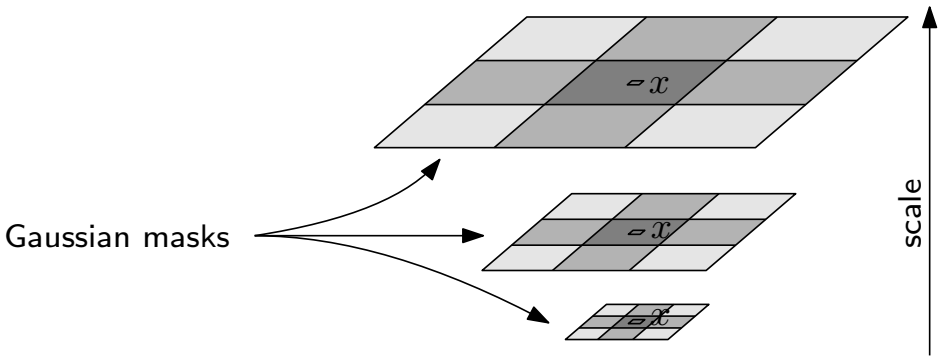


Image features

Input: Images via intensity maps $I : \mathbb{Z}^2 \to \mathbb{R}$, one for each color channel

SIFT (Scale-Invariant Feature Transform) at pixel $x \in \mathbb{Z}^2$:

b) **choose orientation**:

- compute intensity gradient at each pixel y in a window of size $\propto \sigma$ around x
- build histogram of gradient directions (36 bins, 10 degrees each)
- assign orientation corresponding to highest peak in histogram
- rotate image so assigned orientation is vertical

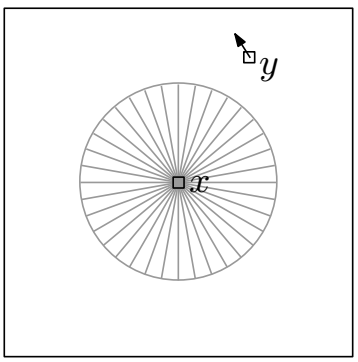


Image features

Input: Images via intensity maps $I : \mathbb{Z}^2 \to \mathbb{R}$, one for each color channel

SIFT (Scale-Invariant Feature Transform) at pixel $x \in \mathbb{Z}^2$:

c) **compute feature**:

- subdivide 16×16 window around x into 16 patches of size 4×4
- compute histogram of gradient orientations (8 bins) in each 4×4 patch
- collect the $8 \times 4 \times 4 = 128$ values (weighted by Gaussian at x) into a vector

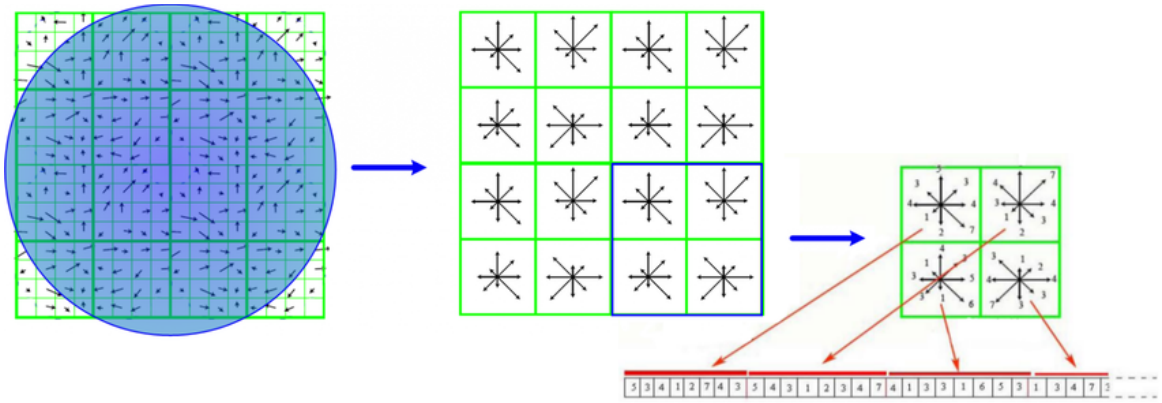


Image features

Input: Images via intensity maps $I : \mathbb{Z}^2 \to \mathbb{R}$, one for each color channel

SIFT (Scale-Invariant Feature Transform) at pixel $x \in \mathbb{Z}^2$:

c) **compute feature**:

- subdivide 16×16 window around x into 16 patches of size 4×4
- compute histogram of gradient orientations (8 bins) in each 4×4 patch
- collect the $8 \times 4 \times 4 = 128$ values (weighted by Gaussian at x) into a vector

From local to global features:

- sample the image domain uniformly
- compute SIFT feature at each sample
- concatenate feature vectors or collect them in a set (more on this later)

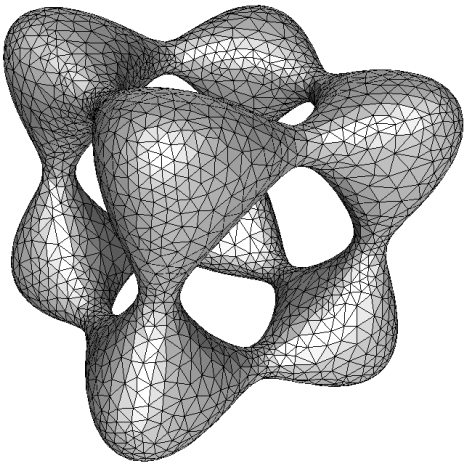
Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

3d shape features

Input: 3d shapes via (triangular) meshes

- non-canonical reprensentation
- local features designed to characterize the shape's local geometry
- used for salient point detection
- can be combined into global features
  for e.g. shape comparison and matching

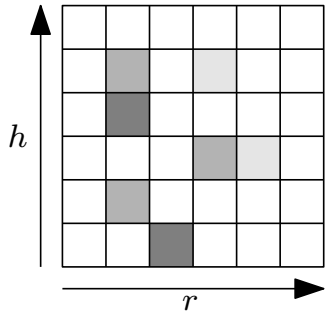


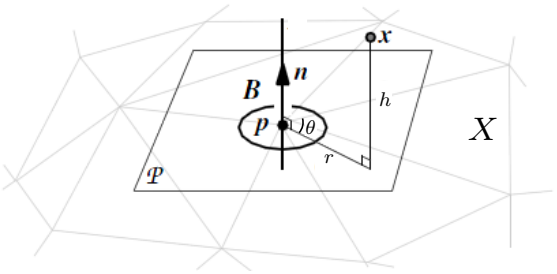
3d shape features

Input: 3d shapes via (triangular) meshes

Spin image at vertex p on a triangulated shape X:

- estimate tangent plane $\mathcal{P}$ and oriented unitary normal $\vec{n}$ at p,
- compute cylindric coordinates (r, θ, h) for every other vertex of X
- keep only those vertices with small distance & normal deviation from p
- record histogram of the distribution of radii & heights (r, h) (invariant to spin)





3d shape features

Input: 3d shapes via (triangular) meshes

Spin image at vertex p on a triangulated shape X:

- estimate tangent plane $\mathcal{P}$ and oriented unitary normal $\vec{n}$ at p,
- compute cylindric coordinates (r, θ, h) for every other vertex of X
- keep only those vertices with small distance & normal deviation from p
- record histogram of the distribution of radii & heights (r, h) (invariant to spin)

From local to global features:

- sample the shape uniformly
- compute spin image at each sample
- collect the spin images in a set (more on this later)

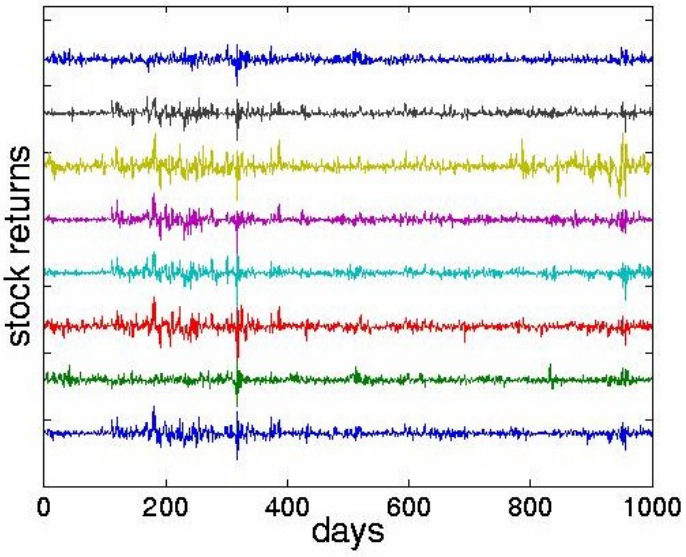
Outline

- Principles of feature extraction
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- Curse of dimensionality and dimensionality reduction
- Vector quantization
- Principal component analysis (PCA)

Time series features

Input: time series $f : \mathbb{N} \to \mathbb{R}^d$

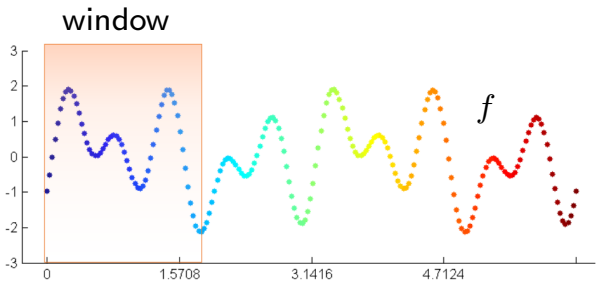
▶ may be chaotic, irregularly sampled, multivariate, hard to realign, etc.



Time series features

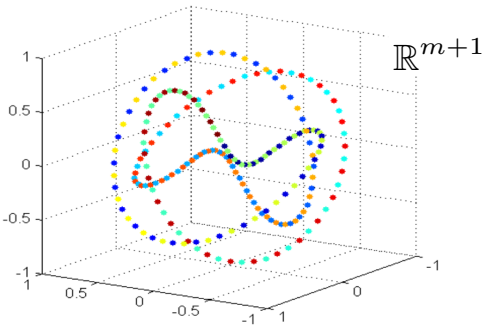
Input: time series $f : \mathbb{N} \to \mathbb{R}^d$

Time-delay embedding (a.k.a. sliding-window embedding):



$\mathrm{TD}_{m,\tau}$

(time-delay embedding)



$$\mathrm{TD}_{m,\tau}(f,t) := \begin{bmatrix} f(t) \\ f(t+\tau) \\ \vdots \\ f(t+m\tau) \end{bmatrix}$$

- τ: step / delay
- $m\tau$: window size
- $m+1$: embedding dimension

- point cloud in $\mathbb{R}^{m+1}$ (time is forgotten about)

Time series features

Input: time series $f : \mathbb{N} \to \mathbb{R}^d$

Time-delay embedding (a.k.a. sliding-window embedding):

window

$\mathrm{TD}_{m,\tau}$

(time-delay embedding)

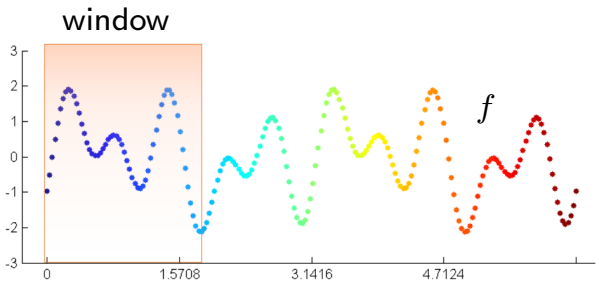
$\mathbb{R}^{m+1}$

signal	embedded data
periodicity	circularity
max. frequency (ν)	min. ambient dimension ($m \geq 2\nu$)
# non-commensurate freq.	intrinsic dimension ($\mathbb{S}^1 \times \cdots \times \mathbb{S}^1$)

Time series features

Input: time series $f : \mathbb{N} \to \mathbb{R}^d$

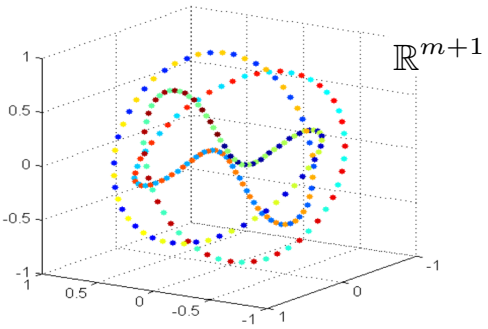
Time-delay embedding (a.k.a. sliding-window embedding):



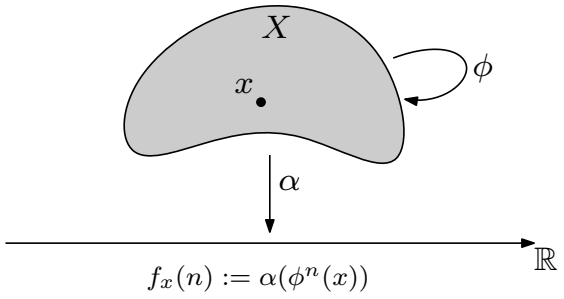
$\mathrm{TD}_{m,\tau}$

$\Longrightarrow$

(time-delay embedding)



Motivation from dynamical systems:



Thm: [Nash, Takens]
Given a Riemannian manifold X of dimension $\frac{m}{2}$, it is a **generic property** of $\phi \in \mathrm{Diff}_2(X)$ and $\alpha \in C^2(X, \mathbb{R})$ that

$$X \to \mathbb{R}^{m+1}$$
$$x \mapsto (\alpha(x), \alpha \circ \phi(x), \cdots, \alpha \circ \phi^m(x))$$

is an embedding.

Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

Curse of dimensionality

Most features live in **high to very high dimensions**:

- size of dictionary for text
- number of graphlets of a given size for graphs
- 128 for SIFT features, multiplied by # sampled pixels
- area of neighborhood for spin images, multiplied by # sampled vertices
- sliding window size over stepsize for time series

Curse of dimensionality: set of phenomena occurring in high dimensions, which are nefastous to the analysis of data.

- poor algorithmic performances of NN-search algorithms
- poor convergence rates of nonparametric density estimators
- poor learning rates of k-NN predictor
- etc.

Curse of dimensionality

Most features live in **high to very high dimensions**:

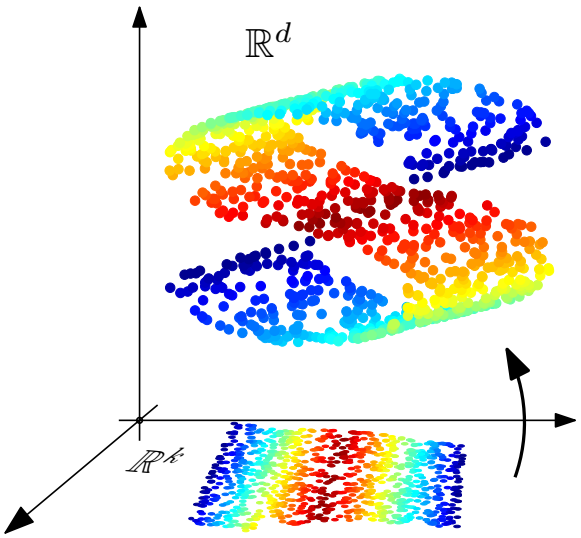
- size of dictionary for text
- number of graphlets of a given size for graphs
- 128 for SIFT features, multiplied by # sampled pixels
- area of neighborhood for spin images, multiplied by # sampled vertices
- sliding window size over stepsize for time series

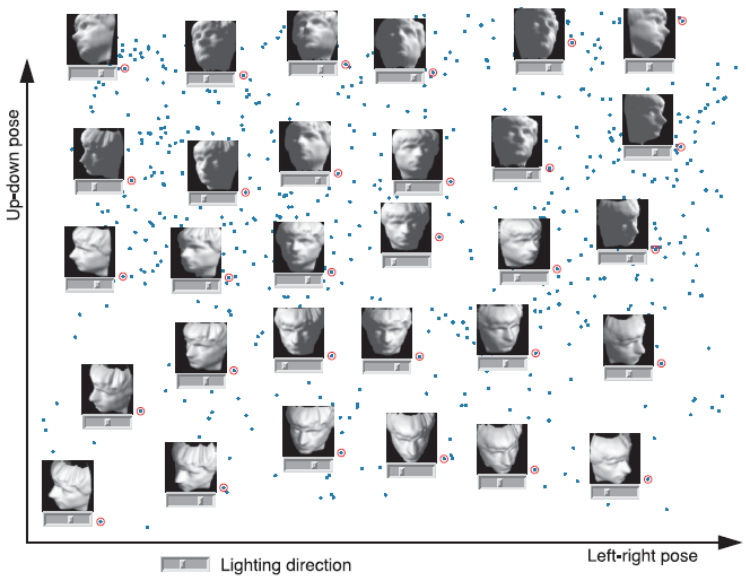
Curse of dimensionality: set of phenomena occurring in high dimensions, which are nefastous to the analysis of data.

- workaround: **dimensionality reduction**

Dimensionality reduction

Hypothesis: **data can be described by a small set of intrinsic (latent) variables**.





Dimensionality reduction

Hypothesis: **data can be described by a small set of intrinsic (latent) variables**.

Contributions of dimensionality reduction:

- prevents the curse of dimensionality
- identifies the latent variables
- resolves degenerate cases (cf. linear regression with dependent variables)

Dimensionality reduction

Hypothesis: **data can be described by a small set of intrinsic (latent) variables**.

A wealth of approaches:

- linear (PCA, MDS, NMF)
- non-linear (kernel-PCA, Isomap, t-SNE, UMAP)
- clustering based (vector quantization)
- neural network based (autoencoders)

Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

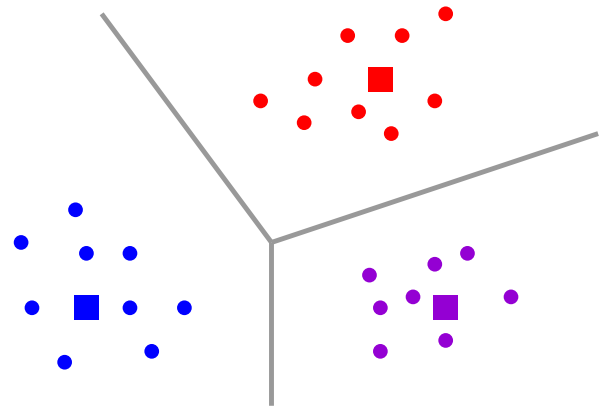
Vector quantization

Hypothesis: **data gather into a small number of clusters**.

- compute clusters $C_i, \cdots, C_k$ and cluster centers $c_1, \cdots c_k$
- compute coordinates $d_1(x_i), \cdots, d_k(x_i)$ w.r.t. cluster centers for each data point x_i
- map data points to $\mathbb{R}^k$ via $x \mapsto [\, d_1(x) \, \cdots \, d_k(x) \,]^T$

Computing the C_j's, c_j's and $d_j(x_i)$'s:

- k-means:
  $d_j(x_i) = \mathbb{1}_{x_i \in X_j}$
- Gaussian mixture models:
  for each x_i, proba. distrib. on the C_i's



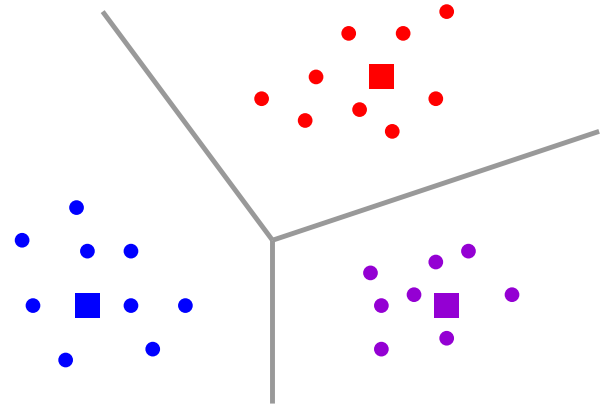
Vector quantization

Hypothesis: **data gather into a small number of clusters**.

- compute clusters $C_i, \cdots, C_k$ and cluster centers $c_1, \cdots c_k$
- compute coordinates $d_1(x_i), \cdots, d_k(x_i)$ w.r.t. cluster centers for each data point x_i
- map data points to $\mathbb{R}^k$ via $x \mapsto \left[d_1(x) \cdots d_k(x) \right]^T$

codebook

code

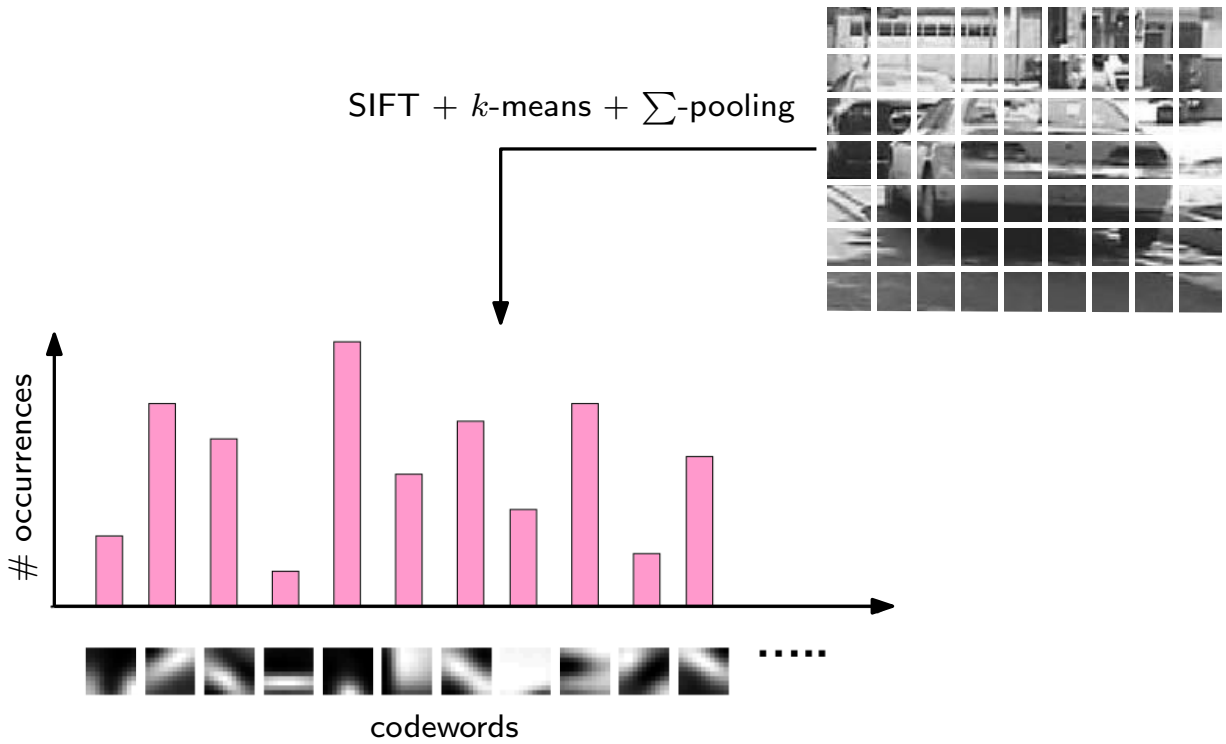


Applications in feature extraction:

- reduce dimensionality of feature space
- encode set of features as a distribution over the codebook (**bag-of-features**)
  - ▷ pooling of word functions $d_j(\cdot)$

Vector quantization

Example with image data (same idea applies to 3d shapes):



Outline

- **Principles of feature extraction**
- **Handcrafted features for:**
  - text data
  - graph data
  - image data
  - 3d shapes data
  - time series data
- **Curse of dimensionality and dimensionality reduction**
- **Vector quantization**
- **Principal component analysis (PCA)**

Linear dimensionality reduction

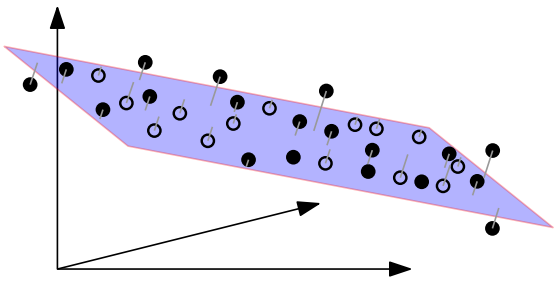
Hypothesis: **data lie on (or close to) some k-dimensional affine subspace**.

Input: $\mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $k \in \mathbb{N}$

Goal: find H minimizing the residual variance:

$$H := \underset{\dim E = k}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(x_i)\|_2^2$$

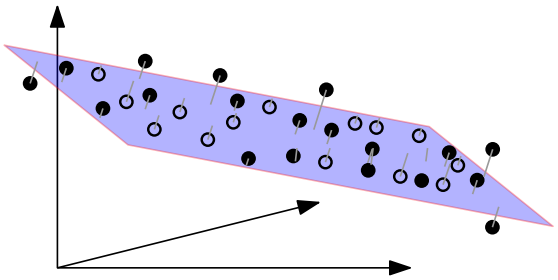
ortho. proj. onto E



Linear dimensionality reduction

Hypothesis: **data lie on (or close to) some k-dimensional affine subspace**.

Input: $\mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $k \in \mathbb{N}$



Goal: find H minimizing the residual variance:

$$H := \underset{\dim E = k}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(x_i)\|_2^2$$

Prop: H contains the centroid $\bar{x} := \frac{1}{n} \sum_{i=1}^{n} x_i$

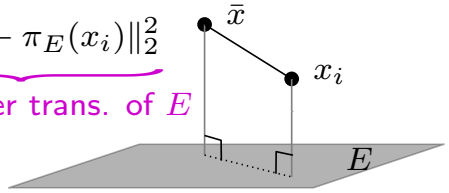
proof:

$$\frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(x_i)\|_2^2 = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(\bar{x})\|_2^2 - \|\pi_E(\bar{x}) - \pi_E(x_i)\|_2^2$$

inv. under trans. of E

$$\frac{1}{n} \sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2 + \|\bar{x} - \pi_E(\bar{x})\|_2^2$$

min. when $\bar{x} \in E$

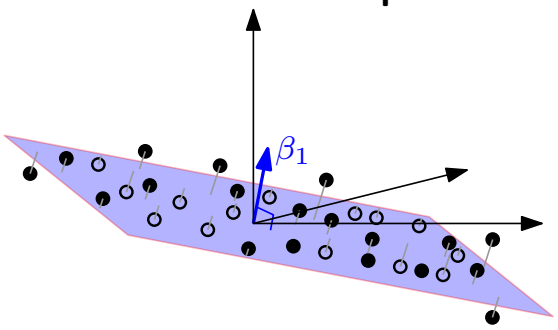


□

Linear dimensionality reduction

Hypothesis: **data lie on (or close to) some k-dimensional affine subspace**.

Input: $\mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $k \in \mathbb{N}$



Goal: find H minimizing the residual variance:

$$H := \underset{\dim E = k}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(x_i)\|_2^2$$

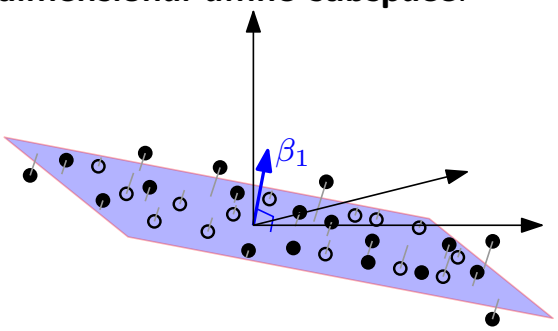
Hyp: dataset is centered ($\bar{x} = 0$): Data centering: $\forall i, \quad x_i \longmapsto x_i - \bar{x}$

$$H : \begin{cases} x^T \beta_1 = 0 \\ \cdots \\ x^T \beta_{d-k} = 0 \end{cases}$$ where $(\beta_1, \cdots, \beta_{d-k})$ is orthonormal

Linear dimensionality reduction

Hypothesis: **data lie on (or close to) some k-dimensional affine subspace**.

Input: $\mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $k \in \mathbb{N}$



Goal: find H minimizing the residual variance:

$$H := \underset{\dim E = k}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|x_i - \pi_E(x_i)\|_2^2$$

Hyp: dataset is centered ($\bar{x} = 0$): Data centering: $\forall i, \quad x_i \longmapsto x_i - \bar{x}$

$$H : x^T \mathbf{B} = 0 \text{ where } \mathbf{B} = \begin{bmatrix} \beta_1 & \cdots & \beta_{d-k} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{d \times d} \text{ s.t. } \mathbf{B}^T \mathbf{B} = \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix}$$

- ideally, $\mathbf{B}$ is such that $\mathbf{X}\,\mathbf{B} = 0$
- in practice, find $\hat{\mathbf{B}}$ minimizing $\|\mathbf{X}\,\mathbf{B}\|_F^2 \quad \left(= \sum_{i=1}^n \|x_i - \pi_H(x_i)\|_2^2\right)$
  $= \sum_{i,j} (\mathbf{X}\,\mathbf{B})_{ij}^2$

Resolution of least-squares problem

$$\hat{\mathbf{B}} := \underset{\mathbf{B}}{\operatorname{argmin}} \|\mathbf{X}\,\mathbf{B}\|_F^2$$

where $\mathbf{B} = \begin{bmatrix} \beta_1 & \cdots & \beta_{d-k} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{d \times d}$ s.t. $\mathbf{B}^T \mathbf{B} = \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix}$

- $\mathbf{B} = \mathbf{W} \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix}$ where $\mathbf{W}^T \mathbf{W} = \mathbf{I}_d$
- Singular value decomposition (SVD) of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}^T \mathbf{D}\,\mathbf{V} \quad \text{where} \begin{cases} \mathbf{U}^T \mathbf{U} = \mathbf{I}_n \\ \mathbf{V}^T \mathbf{V} = \mathbf{I}_d \\ \mathbf{D} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \\ \hline & 0 & \end{bmatrix} \in \mathbb{R}^{n \times d} \quad \text{s.t. } 0 \leq \lambda_1 \leq \cdots \leq \lambda_d \end{cases}$$

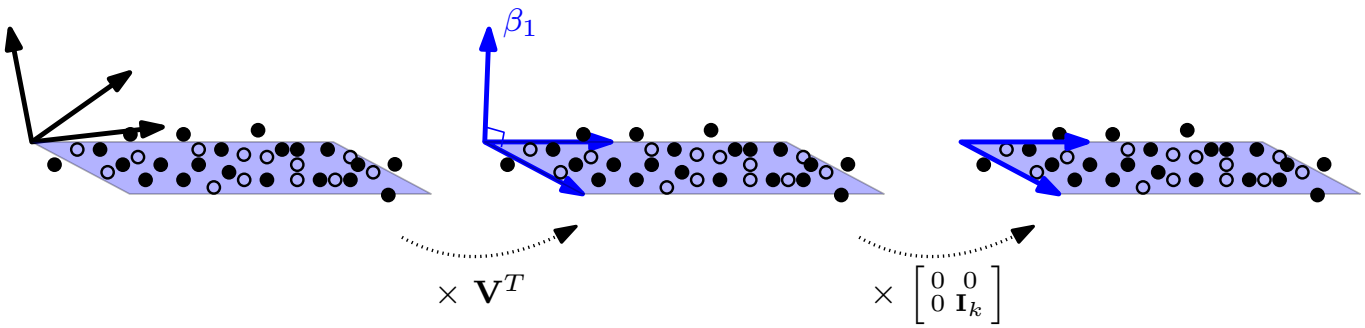
$$\implies \|\mathbf{X}\,\mathbf{B}\|_F^2 = \left\|\mathbf{U}^T \mathbf{D}\,\mathbf{V}\,\mathbf{W} \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix}\right\|_F^2 \text{ is minimum when } \mathbf{W} = \mathbf{V}^T.$$

Resolution of least-squares problem

$$\hat{\mathbf{B}} := \underset{\mathbf{B}}{\operatorname{argmin}} \|\mathbf{X}\,\mathbf{B}\|_F^2 \quad = \mathbf{V}^T \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix} \quad \Longrightarrow \begin{cases} H = \ker \mathbf{V}^T \begin{bmatrix} \mathbf{I}_{d-k} & 0 \\ 0 & 0 \end{bmatrix} \\ \pi_H(x_1, \cdots, x_n) = \mathbf{X}\,\mathbf{V}^T \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_k \end{bmatrix} \end{cases}$$

Geometric interpretation:

- $\mathbf{V}^T$ aligns the frame with the principal directions of the covariance matrix
- $\begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_k \end{bmatrix}$ projects onto the principal directions of largest eigenvalues (variances)



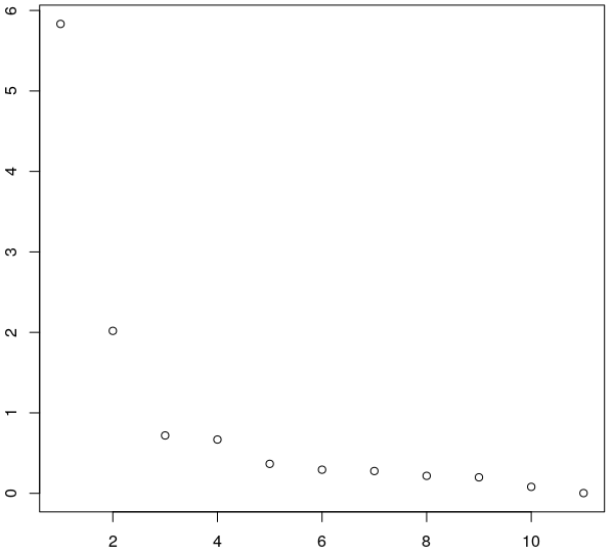
Experimental result

Dataset: 1988 Olympics decathlon (10 variables, 34 observations)

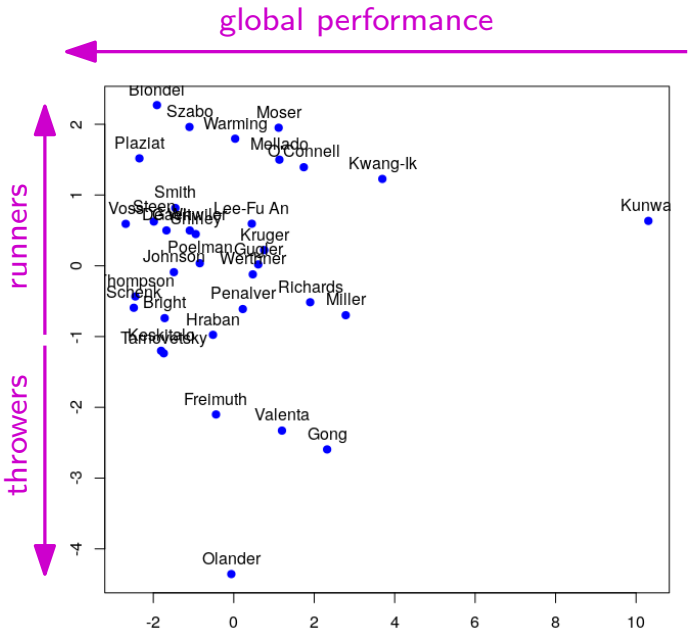
	m100	lgjp	shot	hijp	m400	hrdl	disc	plvlt	jvlin	m1500
Schenk	11.25	7.43	15.48	2.27	48.90	15.13	49.28	4.7	61.32	268.95
Voss	10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.1	61.76	273.02
Steen	11.18	7.44	14.20	1.97	48.29	14.81	43.66	5.2	64.16	263.20
Thompson	10.62	7.38	15.02	2.03	49.06	14.72	44.80	4.9	64.04	285.11
Blondel	11.02	7.43	12.92	1.97	47.44	14.40	41.20	5.2	57.46	256.64
Plaziat	10.83	7.72	13.58	2.12	48.34	14.18	43.06	4.9	52.18	274.07
Bright	11.18	7.05	14.12	2.06	49.34	14.39	41.68	5.7	61.60	291.20
De Wit	11.05	6.95	15.34	2.00	48.21	14.36	41.32	4.8	63.00	265.86
Johnson	11.15	7.12	14.52	2.03	49.15	14.66	42.36	4.9	66.46	269.62
Tarnovetsky	11.23	7.28	15.25	1.97	48.60	14.76	48.02	5.2	59.48	292.24
Keskitalo	10.94	7.45	15.34	1.97	49.94	14.25	41.86	4.8	66.64	295.89
Gaehwiler	11.18	7.34	14.48	1.94	49.02	15.11	42.76	4.7	65.84	256.74
Szabo	11.02	7.29	12.92	2.06	48.23	14.94	39.54	5.0	56.80	257.85
Smith	10.99	7.37	13.61	1.97	47.83	14.70	43.88	4.3	66.54	268.97
Shirley	11.03	7.45	14.20	1.97	48.94	15.44	41.66	4.7	64.00	267.48
Poelman	11.09	7.08	14.51	2.03	49.89	14.78	43.20	4.9	57.18	268.54
Olander	11.46	6.75	16.07	2.00	51.28	16.06	50.66	4.8	72.60	302.42
Freimuth	11.57	7.00	16.60	1.94	49.84	15.00	46.66	4.9	60.20	286.04
Warming	11.07	7.04	13.41	1.94	47.97	14.96	40.38	4.5	51.50	262.41
Hraban	10.89	7.07	15.84	1.79	49.68	15.38	45.32	4.9	60.48	277.84
Werthner	11.52	7.36	13.93	1.94	49.99	15.64	38.82	4.6	67.04	266.42
Gugler	11.49	7.02	13.80	2.03	50.60	15.22	39.08	4.7	60.92	262.93
Penalver	11.38	7.08	14.31	2.00	50.24	14.97	46.34	4.4	55.68	272.68
Kruger	11.30	6.97	13.23	2.15	49.98	15.38	38.72	4.6	54.34	277.84
Lee-Fu An	11.00	7.23	13.15	2.03	49.73	14.96	38.06	4.5	52.82	285.57
Mellado	11.33	6.83	11.63	2.06	48.37	15.39	37.52	4.6	55.42	270.07
Moser	11.10	6.98	12.69	1.82	48.63	15.13	38.04	4.7	49.52	261.90
Valenta	11.51	7.01	14.17	1.94	51.16	15.18	45.84	4.6	56.28	303.17
O'Connell	11.26	6.90	12.41	1.88	48.24	15.61	38.02	4.4	52.68	272.06
Richards	11.50	7.09	12.94	1.82	49.27	15.56	42.32	4.5	53.50	293.85
Gong	11.43	6.22	13.98	1.91	51.25	15.88	46.18	4.6	57.84	294.99
Miller	11.47	6.43	12.33	1.94	50.30	15.00	38.72	4.0	57.26	293.72
Kwang-Ik	11.57	7.19	10.27	1.91	50.71	16.20	34.36	4.1	54.94	269.98
Kunwar	12.12	5.83	9.71	1.70	52.32	17.05	27.10	2.6	39.10	281.24

Experimental result

Dataset: 1988 Olympics decathlon (10 variables, 34 observations)



Spectrum of SVD diagonal matrix $\mathbf{D}$
(ordered by decreasing value)



embedding into $\mathbb{R}^k$ with $k = 2$

What you should know

- Concepts: feature extraction / engineering / learning
- Examples of features for text and graph data
- Curse of dimensionality and principles of dimensionality reduction
- Vector quantization
- PCA: linear model, least squares problem, resolution by SVD