Algorithmic Data Analysis

Esther Galbrun

Spring 2024

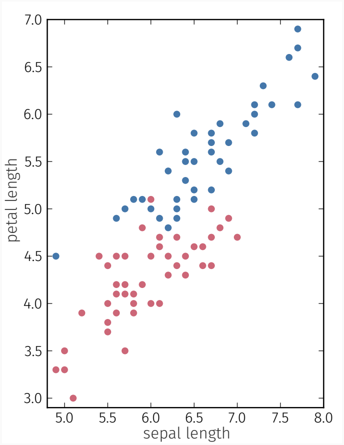
UNIVERSITY OF
EASTERN FINLAND

Last updated on 2023-12-28 13:06

Part I

Classification variants

A simple example



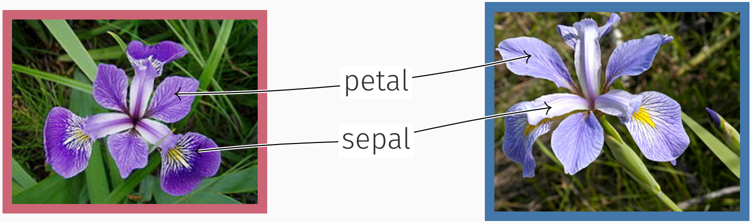
A dataset with two classes

A simple example

data points: Iris flowers

attributes: physical properties,
length of the petal and length of the sepal in *cm*

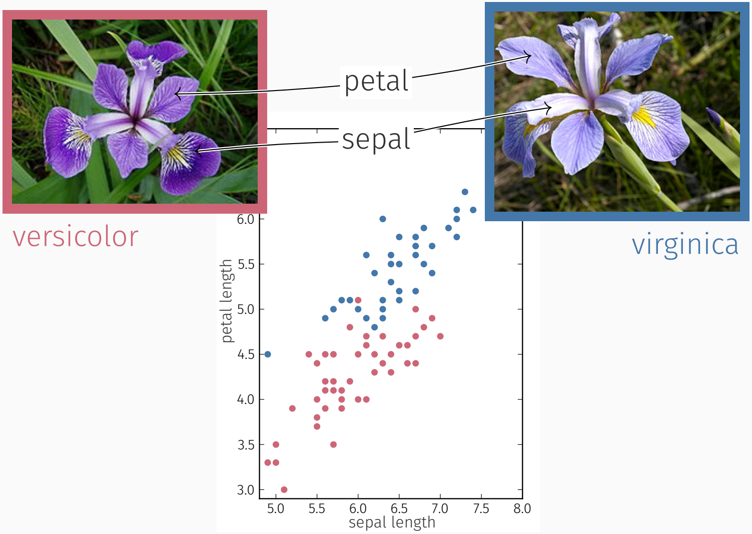
class: species, *versicolor* vs. *virginica*



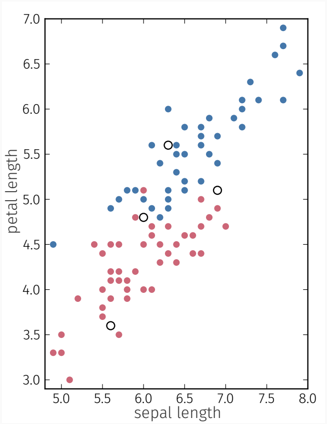
versicolor

virginica

A simple example



A simple example

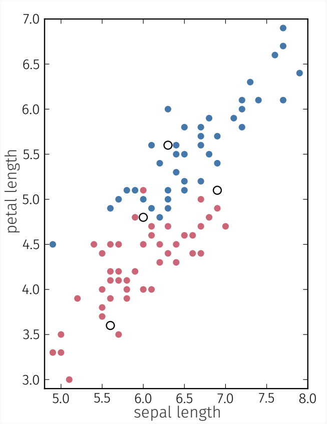


?

Class information, i.e. species, is absent for some points

Can we use the available information to predict it?

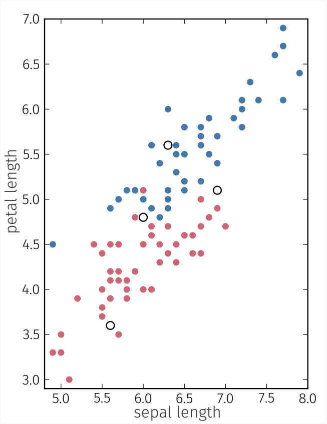
A simple example



?

classification aim to assign a class label to each instance

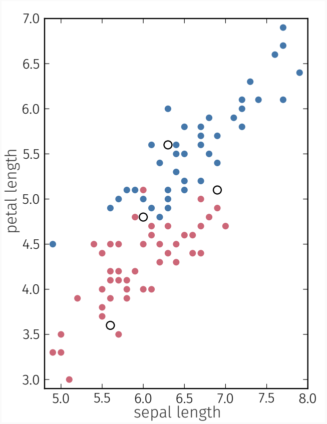
A simple example



?

binary there are two classes to choose from

A simple example



?

supervised labelled training instances are available

A simple example

supervised	labelled training instances are available
binary	there are two classes to choose from
classification	aim to assign a class label to each instance

A typical supervised binary classification problem

Some notations

The data set, denoted as $\mathcal{D}$, contains n data points and m attributes, i.e. it is a $n \times m$ matrix

A data point is a m-dimensional vector $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_m \rangle$
We denote $\boldsymbol{x}^{(j)}$ the j^{th} data point of $\mathcal{D}$, i.e. the j^{th} row
Data points are sometimes called *instances* or *examples*

Class labels are arranged into a n-dimensional vector $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle \in \mathcal{L}^n$, where $l = |\mathcal{L}|$ is the number of classes
That is, y_j is the class label associated with data point $\boldsymbol{x}^{(j)}$
In binary classification, class labels take value -1 or $+1$ (sometimes 0 or 1 instead), i.e. $\mathcal{L} = \{-1, +1\}$ (respectively $\mathcal{L} = \{0, 1\}$) and the two classes might be referred to as negative and positive, respectively

Different methods

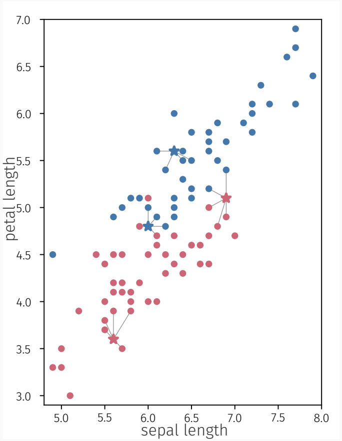
A typical supervised binary classification problem

Various classification methods are available to tackle it

Different methods

Look at the most similar data points

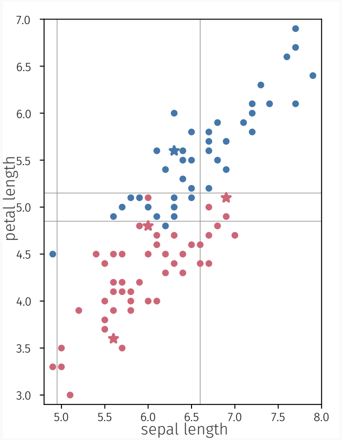
→ k nearest neighbors (k-NN)

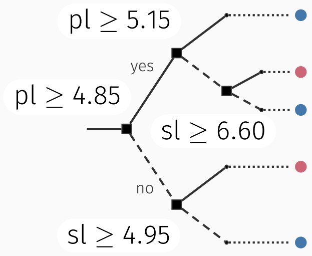


majority class
among k nearest neighbors

Different methods

Apply a sequence of tests on attributes' values
$\rightarrow$ classification tree

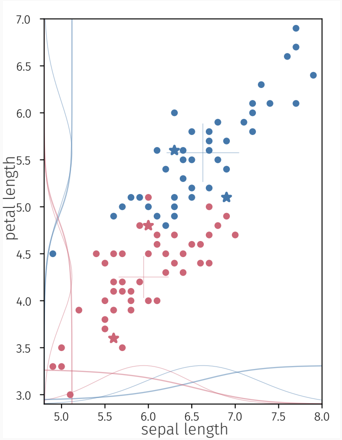




Different methods

Look at class probabilities conditioned on attributes' values

$\rightarrow$ Naive bayes



$$P(c \mid \mathrm{sl}, \mathrm{sp}) \propto P(c) \cdot P(\mathrm{sl} \mid c) \cdot P(\mathrm{sp} \mid c)$$

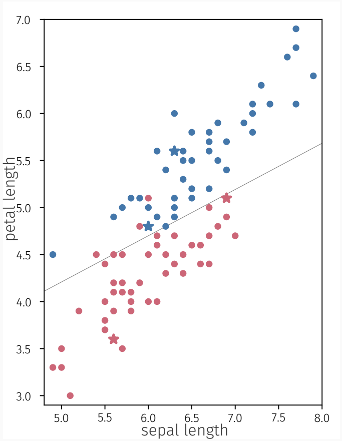
$$P(● \mid \mathrm{sl}, \mathrm{sp}) > P(● \mid \mathrm{sl}, \mathrm{sp}) \quad ●$$

$$P(● \mid \mathrm{sl}, \mathrm{sp}) \leq P(● \mid \mathrm{sl}, \mathrm{sp}) \quad ●$$

Different methods

Look at the sign of a linear combination of the attributes

$\rightarrow$ perceptron

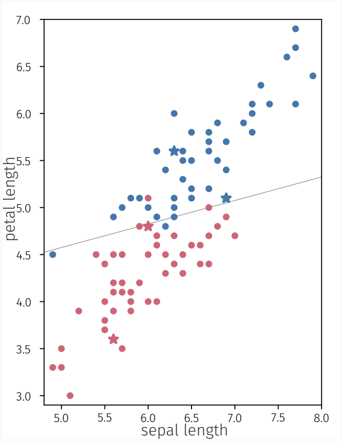


$0.671 \cdot \mathrm{sl} - 1.365 \cdot \mathrm{pl} + 2.39 < 0$ ●

$0.671 \cdot \mathrm{sl} - 1.365 \cdot \mathrm{pl} + 2.39 \geq 0$ ●

Different methods

Look at the sign of a linear combination of the attributes
$\rightarrow$ support vector machine (SVM)



$\mathrm{sl} - 4 \cdot \mathrm{pl} + 13.3 < 0$ ●

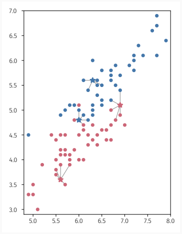
$\mathrm{sl} - 4 \cdot \mathrm{pl} + 13.3 \geq 0$ ●

Different methods

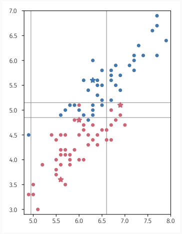
A typical supervised binary classification problem

Various classification methods are available to tackle it

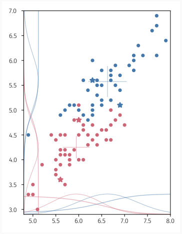
k-NN



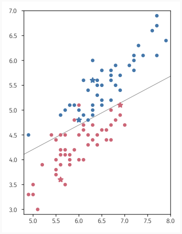
decision tree



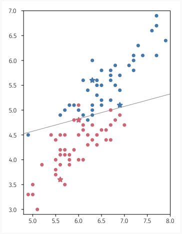
naive Bayes



perceptron



SVM



A simple example

A typical supervised binary classification problem
Various classification methods are available to tackle it

Problem variants

- What if there are more than two classes?
  $\rightarrow$ Multi-class learning
- What if the two classes are not equally represented?
  $\rightarrow$ Rare-class learning

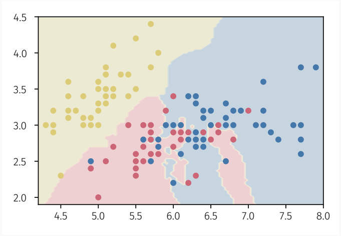
Methods

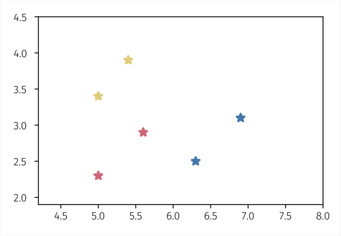
- How about combining multiple classifiers?
  $\rightarrow$ Ensemble methods

Multi-class learning

No adaptation needed

Some methods can handle multiple classes
→ k nearest neighbors (k-NN)

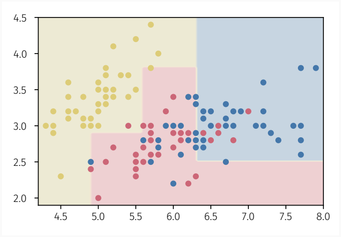


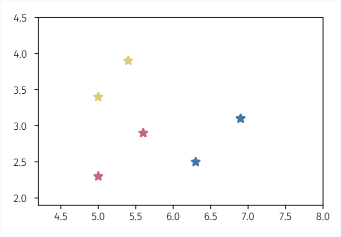


No adaptation needed

Some methods can handle multiple classes

$\rightarrow$ classification tree

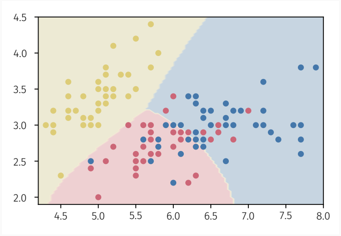


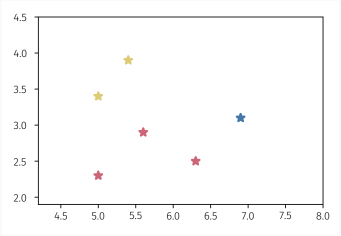


No adaptation needed

Some methods can handle multiple classes

→ Naive bayes





Adaptations needed

Other methods, like the Perceptron and SVMs are naturally designed for the binary scenario

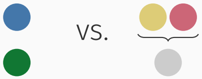
Method-specific adaptations to the multi-class scenario exist

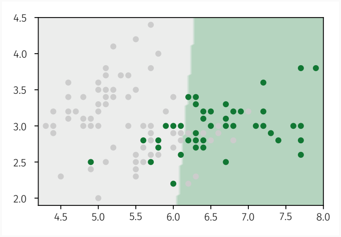
Generic, method-agnostic, meta-frameworks are helpful
Two main strategies

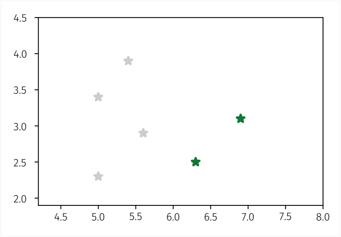
one-against-rest and one-against-one

One-against-rest

Create a new binary classification problem for each class: examples from that class are constitute positive examples the rest are negative examples

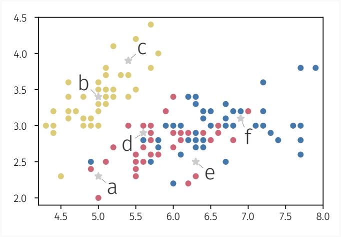






One-against-rest

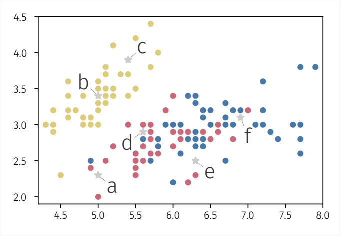
Predictions from the different problems are then combined
Might require tie-breaking,
using weighted rather than crisp votes can help



	🔵	🟡	🔴	
a	☆	☆	★	🔴
b	☆	★	☆	🟡
c	☆	★	☆	🟡
d	☆	☆	☆	🔵
e	★	☆	★	🔴
f	★	☆	☆	🔵

One-against-rest

A *k* class problem maps to *k* binary models

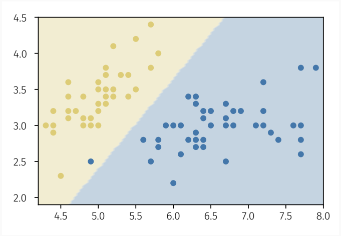


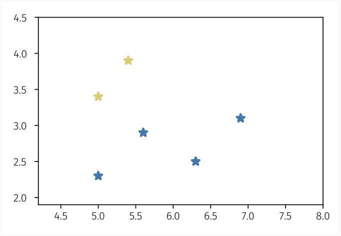
	🔵	🟡	🔴	
a	☆	☆	★	🔴
b	☆	★	☆	🟡
c	☆	★	☆	🟡
d	☆	☆	☆	?
e	★	☆	★	?
f	★	☆	☆	🔵

One-against-one

Create a new binary classification problem for each pair of classes, considering only examples from these two classes

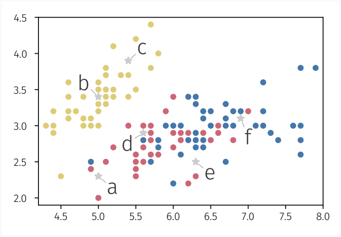
● vs. ●





One-against-one

Predictions from the different problems are then combined
Might require tie-breaking,
using weighted rather than crisp votes can help

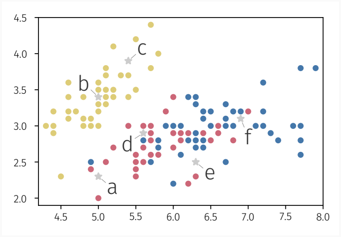


	🔵/🟡	🔵/🔴	🟡/🔴	
a	🔵	🔴	🔴	🔴
b	🟡	🔴	🟡	🟡
c	🟡	🔴	🟡	🟡
d	🔵	🔴	🔴	🔴
e	🔵	🔵	🔴	🔵
f	🔵	🔵	🔴	🔵

One-against-one

A k class problem maps to $\binom{k}{2} = k(k-1)/2$ binary models

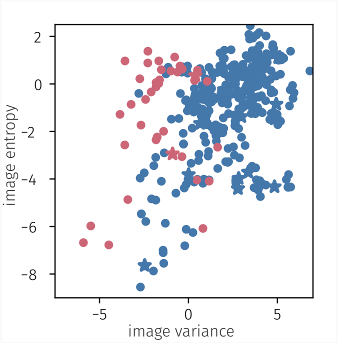
More problems than one-against-rest, but smaller



	🔵/🟡	🔵/🔴	🟡/🔴	
a	🔵	🔴	🔴	🔴
b	🟡	🔴	🟡	🟡
c	🟡	🔴	🟡	🟡
d	🔵	🔴	🔴	🔴
e	🔵	🔵	🔴	🔵
f	🔵	🔵	🔴	🔵

Rare-class learning

Rare-class scenario

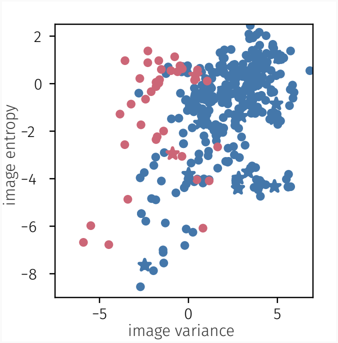


Normal banknotes
are much more common than
fraudulent banknotes
(343 to 37)

It is important to achieve high accuracy on the rare class, at the cost of reduced accuracy on the normal class

Associate different weights to the classes and try to maximize the **weighted accuracy**

Rare-class scenario



Normal banknotes
are much more common than
fraudulent banknotes
(343 to 37)

Two main strategies
example reweighting and example resampling

Rare-class scenario

Example reweighting

- weights are associated to training examples according to their missclassification cost
- algorithms require adaptations to handle these weights

Example resampling

- examples from rare class might be oversampled,
  or examples from normal class be undersampled,
  or a combination of both
- algorithms do not require any adaptation

Rare-class scenario

In effect, **resampling** and **reweighting** are almost equivalent
resampling can be understood as sampling examples in proportion to their *weights* then treating them equally

Resampling is easier to combine with other approaches

Undersampling is more efficient (smaller datasets)

Resampling has greater randomness
Reweighting is more reliable

Ensemble methods

Ensemble methods

Different classifiers might make different predictions on the same data point due to their specific characteristics or their sensitivity to random artifact in the training data

The aim of ensemble methods is to increase prediction accuracy by combining the results of multiple classifiers

Ensemble methods

For $i = 1, \ldots, \ell$, train model $\mathcal{M}^{(i)}$ on dataset $\mathcal{D}^{(i)}$
Combine the predictions of the different models into a single robust prediction

Data-centered ensembles use a single algorithm on different derivative datasets

Model-centered ensembles use different algorithms or different parameter settings of the same algorithm on a single dataset

Bucket of models

The performance of the bucket of models is only as good as the best model in the bucket for a particular dataset

Over multiple datasets the approach is able to select the model that is best suited to each case

Bagging

If the variance of a single prediction is σ, the variance of the average of ℓ independent and identically distributed (i.i.d.) such predictions is reduced to σ^2/ℓ

Derivative datasets are created using **bootstrap sampling** $\mathcal{D}^{(i)}$ is a subset of data points sampled uniformly with replacement from $\mathcal{D}$ to approximately the same size as $\mathcal{D}$

Report the majority vote among the predictions of the models as the ensemble's prediction

Bagging (a.k.a. bootstrapped aggregating) helps **reduce variance** through aggregation
Individual models should be designed so as to reduce bias as much as possible, even at the expense of variance

Random forests

A random forest is an ensemble of decision trees where randomness is added explicitely at the split selection to reduce correlation between the components

During tree construction, each split selection is preceded by the random selection of q attributes, among which the split criterion is then chosen, rather than from the entire set of m attributes

Boosting

weak learner a classifier that is only slightly correlated with the ground truth, i.e. one that performs only slightly better than random guessing

strong learner a classifier that is arbitrarily well correlated with the ground truth, i.e. one of arbitrarily high accuracy

Hypothesis boosting aims to turn a weak learner into a strong learner

Boosting

Successive models $\mathcal{M}^{(t)}$ are built by applying the same algorithm to weighted variants $\mathcal{D}^{(t)}$ of the dataset

Weights associated to every training instance are adjusted so that the model will focus more on previously missclassified instances

The prediction of the ensemble is a weighted combination of all the models' predictions

Many boosting algorithms have been proposed
AdaBoost (short for Adaptive Boosting) is most popular

Boosting

Boosting primarily focuses on **reducing bias**
It aims to combine many weak learners into a strong learner
The approach should be used with simple models having high bias but low variance

When re-weighting is done via sampling, it can also help reduce variance

The approach is vulnerable to noise
It assumes that error is caused by bias, in the presence of noise it will overtrain on low-quality portions of the data

Typically superior to bagging when noise is not excessive

Stacking

The training dataset is divided into two subsets $\mathcal{D}_A$ and $\mathcal{D}_B$
$\mathcal{D}_A$ is used to train ℓ models, the ensemble components
$\mathcal{D}_B$ is used to train a second-level classifier that combines the predictions of the ensemble components

Stacking

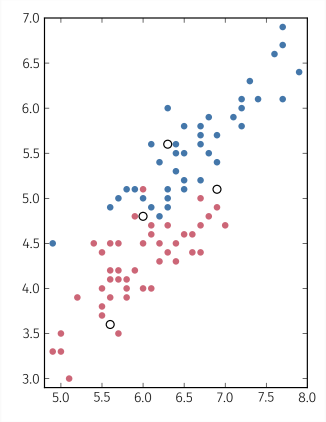
By learning from the errors of the ensemble components stacking allows to reduce both bias and variance

The power of stacking comes from the flexible learning approach of the combiner
Many other ensemble methods can be seen as special cases using less flexible, data-independent, combination procedures such as voting

Part II

Classification
Different paradigms

A simple example



?

A typical supervised binary classification problem

supervised labelled training instances are available

Learning paradigms

Supervised learning labelled training instances
$\rightarrow$ *Classification*

Learning paradigms

Supervised learning labelled training instances
$\rightarrow$ *Classification*

Unsupervised learning unlabelled training instances
$\rightarrow$ *Clustering*

Learning paradigms

Supervised learning labelled training instances
$\rightarrow$ *Classification*

Unsupervised learning unlabelled training instances
$\rightarrow$ *Clustering*

Reinforcement learning choose actions to maximize cumulative rewards
$\rightarrow$ *Exploration–exploitation trade-off*

Learning paradigms

Supervised learning labelled training instances

Unsupervised learning unlabelled training instances

Semi-supervised learning few labelled + mostly unlabelled

Learning paradigms

Supervised learning labelled training instances

- **Semi-supervised learning** few labelled + mostly unlabelled
- **Active learning** query labels selectively, at a cost
- **Online learning** data arrives and is processed iteratively
- **Transfer learning** reuse what has been learnt on one task on a different task

$\rightarrow$ *Classification*

Learning paradigms

Supervised learning labelled training instances

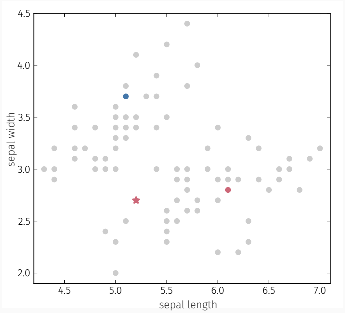
- **Semi-supervised learning** few labelled + mostly unlabelled
- **Active learning** query labels selectively, at a cost
- **Online learning** data arrives and is processed iteratively
- **Transfer learning** reuse what has been learnt on one task on a different task

$\rightarrow$ *Classification*

Semi-supervised learning

Semi-supervised learning

The aim of semi-supervised learning is to exploit both labelled and unlabelled data to improve learning



Induction vs. transduction

Inductive algorithms proceed in two well-separated phases

Training	learn a general rule from training instances
Testing	apply the general rule to test instances

Transductive algorithms use test instances for training

- require test instances to be specified at training time
- use information from test instances as unlabelled data points during training
- might not allow prediction on out-of-sample instances

Transductive Support Vector Machines

Transductive SVM

Find a separating hyperplane with maximum margin
Label unsupervised examples to maximize the margin

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{j=1}^{j=n}\xi_j \\
\text{s.t.} \quad & y_j(\boldsymbol{w}\cdot\boldsymbol{x}^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \ \forall j \in \mathcal{I}_L \\
& z_j(\boldsymbol{w}\cdot\boldsymbol{x}^{(j)} + b) \geq 1 - \xi_j \text{ and } 0 \leq \xi_j \ \forall j \in \mathcal{I}_U
\end{aligned}$$

$\mathcal{I}_L$ and $\mathcal{I}_U$ index labelled and unlabelled examples respectively

y_j are known, class labels of the supervised examples
z_j are unknown, binary integer variables to be optimized

EM clustering/Naives Bayes classification

Labelled examples deterministic assignment
initialize parameters and stabilize EM process

Unlabelled examples probabilistic assignment
estimate the cluster structure

Graph-based collective classification with random walks

Represent the data as a graph

Start random walk from unlabelled node, stop at the first encountered labelled node

Assign class at which the random walk is most likely to terminate

Key assumption: the graph must be *label-connected*

Semi-supervised learning

Two types of approaches

Method-specific adaptations

- Transductive SVM
- Semi-supervised Bayes classification with EM

Graph-based collective classification

Generic meta-algorithms

- Self-training
- Co-training

Semi-supervised learning: generic meta-algorithms

Use the smoothness assumption to incrementally expand the labelled portion of the data
$\rightarrow$ self-training

! Risk of error propagation and overfitting

Similar procedure but with two models trained on separate subsets of attributes generate labels for one another
$\rightarrow$ co-training

Semi-supervised learning

Method-specific adaptations

- Transductive SVM
- Semi-supervised Bayes classification with EM

Graph-based collective classification

Generic meta-algorithms

- Self-training
- Co-training

Working assumption:
class structure approximately matches clustering structure

Most useful when labelled examples are scarce

Active learning

Active learning

Labelled data is difficult and expensive to acquire
Cost can be evaluated or at least estimated

Not all training instances are equally useful

The aim of active learning is to train the most accurate model within a given budget
Integrate label acquisition and model building to achieve highest cost-efficiency

Active learning is sometimes known as *query learning* or *optimal experimental design*

Active learning

Active learning assumes access to an oracle, i.e. a means to obtain labels for queried instances, seen as a black-box

The querying system asks the oracle for the labels of specific instances, selected following some strategy

Active learning scenarios

Membership query synthesis generates a synthetic instance

! instance might not be realistic

Selective sampling unlabelled instances arrive one by one and the learner makes a decision to query the label from the oracle or to discard (a.k.a. *stream-based* or *sequential* AL)

Pool-based sampling a collection of interesting examples to query is sampled from a large pool of available unlabelled instances

Focus on the latter, most common scenario

Active learning process

The active learning process is iterative and starts with

- small collection of labelled instances L
- large collection of unlabelled instances U
- query budget b

$\mathrm{f}_O(x)$ is the label for data point x obtained from oracle O
$\mathrm{c}_O(x)$ is the associated cost

```
**while** $b > 0$ and accuracy improves **do**
    Train model $\mathcal{M}$ on $L$
    $C \leftarrow \{$most interesting instances from $U\}$
    $U \leftarrow U \setminus C$
    $L \leftarrow L \cup \{(x, \mathrm{f}_O(x))$ for $x \in C\}$
    $b \leftarrow b - \sum_{x \in C} \mathrm{c}_O(x)$
```

Active learning process

Clearly, the crucial part of active learning is the selection of *most interesting instances*, i.e. the querying strategy

```
**while** $b > 0$ and accuracy improves **do**
    Train model $\mathcal{M}$ on $L$
    $C \leftarrow \{$most interesting instances from $U\}$
    $U \leftarrow U \setminus C$
    $L \leftarrow L \cup \{(x, f_O(x))$ for $x \in C\}$
    $b \leftarrow b - \sum_{x \in C} c_O(x)$
```

Querying strategies

Heterogeneity-based strategies sample regions that are uncertain, heterogeneous or dissimilar to what has been seen so far

Performance-based strategies evaluate the impact of adding the queried instance on the performance of the model

Representativeness-based strategies query instances so as to obtain a distribution of instances that is representative of the underlying population

Querying strategies

uncertainty sampling

maximize label uncertainty on **queried instance**

vs.

expected error reduction

minimize label uncertainty on **remaining unlabelled instances**

Part III

Mining Data Streams

Problem

Data streams

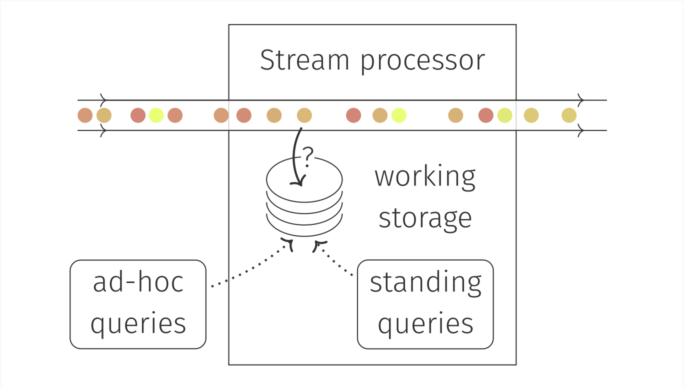
Vast amounts of data are acquired automatically

- satellite images, GPS traces
- measurements from wearable and mobile devices
- web server log traces
- user interactions on social networks
- credit card transactions

Continuous, large, rapid supply of data records
Storing all incoming data for offline processing is not possible

Algorithms must cope with amounts of incoming data many times larger than available memory
$\rightarrow$ Data stream paradigm

Data stream management system



We do not know the whole data in advance
We can think of the data as infinite and non-stationnary
How to make calculations with only limited working storage?

Constraints

Constraints encountered in the data stream paradigm include

One-pass	data records can be processed only once
Concept-drift	the data may evolve over time
Resources	the system might need to drop part of the data
Massive domain	for streams of discrete attributes, the number of distinct values might be very large

Synopsis data structures

Synopsis data structures

A synopsis is a concise representation of the data stream
maintained dynamically in the working storage
to be leveraged for answering queries

Sampling data points

Simple, flexible and generic synopsis data structure
Almost any algorithm can be applied to the sample
Unsuitable for a few specific queries such as counting distinct elements

Reservoir sampling

Maintain a *dynamically updated* sample of k data points

1. Insert n^{th} incoming data point with probability k/n
2. If new point is inserted, eject an old point at random

Theorem
After n data points have arrived, the probability of any point being included in the reservoir is the same and equal to k/n

Bias-sensitive sampling

Exponential bias function with bias rate $\lambda \in [0, 1]$

$$\mathrm{b}(r, n) = e^{-\lambda \cdot (n-r)}$$

Let $\mathrm{F}(n) \in [0, 1]$ be the fraction of the reservoir that is filled before arrival of n^{th} data point

The new point is inserted with probability $k \cdot \lambda$

A coin is flipped, with success probability $\mathrm{F}(n)$
If success, the new point replaces a randomly selected point in the reservoir, otherwise, the new point is added

Quality bounds

Having generated a sample of data points, we can use it to estimate statistical properties of the data

It is important to quantify the accuracy of these estimates, i.e. bound the quality of query answers

Probabilistic inequalities provide such bounds

Quality bounds: Markov's inequality

Theorem
Let X be a random variable that takes on only nonnegative random values. Then, for any constant α satisfying $E[X] \leq \alpha$

$$\mathrm{P}(X \geq \alpha) \leq E[X]/\alpha$$

Markov's inequality provides a bound on the *upper tail* of the probability distribution of *nonnegative values*

Quality bounds: Chebychev's inequality

Theorem
Let X be an arbitrary random variable. Then, for any constant α

$$\mathrm{P}(|X - E[X]| \geq \alpha) \leq \mathrm{var}[X]/\alpha^2$$

Chebychev's inequality provides a bound on *both tails* of the probability distribution of *arbitrary values*

Quality bounds: Chernoff bounds

Theorem
Let X be a random variable that can be expressed as the sum of n independent Bernoulli random variables with success probabilities respectively p_i.

Then, for any $\delta \in [0, 1]$

$$P(X \leq (1-\delta)E[X]) \leq e^{-E[X]\delta^2/2} \quad \textit{(lower-tail bound)}$$

and for any $\delta \in [0, 2e-1]$

$$P(X \geq (1+\delta)E[X]) \leq e^{-E[X]\delta^2/4} \quad \textit{(upper-tail bound)}$$

Chernoff bounds are tighter than Markov's and Chebychev's inequalities, for sum of independent binary random variables

Quality bounds: Hoeffding's inequality

Theorem
Let X be a random variable that can be expressed as the sum of n independent random variables, each bounded in $[l_i, u_i]$.

Then, for any $\theta \geq 0$

$$P(E[X] - X \geq \theta) \leq e^{-\frac{2\theta^2}{\sum_i (u_i - l_i)^2}} \qquad \textit{(lower-tail bound)}$$

$$P(X - E[X] \geq \theta) \leq e^{-\frac{2\theta^2}{\sum_i (u_i - l_i)^2}} \qquad \textit{(upper-tail bound)}$$

Hoeffding's inequality is stronger than Markov's and Chebychev's inequalities and applies to sum of independent bounded random variables

Massive domain scenario

In many applications, the data stream contains discrete attributes with a very large number of distinct values (IP adresses, emails, etc.)

Some simple queries can already be challenging

Has this item occurred earlier in the stream?
$\rightarrow$ Finding duplicates

Does this item occur in set $\mathcal{S}$?
$\rightarrow$ Allowing only elements with a particular property
spam filtering: the elements of the stream are sender email adresses, $\mathcal{S}$ are authorized senders (whitelisting)

Sampling schemes do not work well in such cases

Hash functions

Hash functions
crucial ingredient of probabilistic streaming algorithms
provide *reproducible randomness*

Hash function h maps every value in the input domain uniformly to a bit-string of fixed size

Bloom filters

Does this item occur in set $\mathcal{S}$?

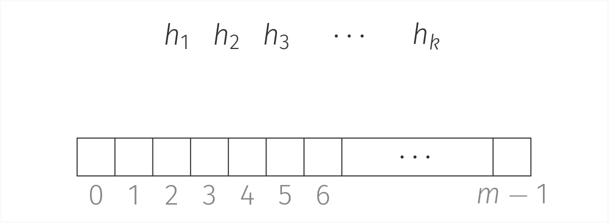
Bloom filters provide a means to answer **set-membership queries** probabilistically, when $\mathcal{S}$ cannot be stored explicitly in a hash table

! False positives are possible, false negatives are not

Bloom filters

A Bloom filter consists of

- a binary bit array B of length m, whose elements are indexed from 0 to $m-1$
- a set of k independent hash functions $h_1, \ldots, h_k$, mapping elements from the data stream to an integer in $[0, m-1]$ uniformly at random



Bloom filters

Theorem
Consider a Bloom filter B of length m with k hash functions.
Let n be the number of distinct values in $\mathcal{S}$, and $\mathbf{y} \notin \mathcal{S}$.
The probability that $\mathbf{y}$ is reported as a false positive is

$$\left(1-(1-1/m)^{kn}\right)^{k}$$

Bloom filters

Besides set-membership queries, Bloom filters can be used for alternative purposes

number of distinct values in a set

size of the union and of the intersection of different sets

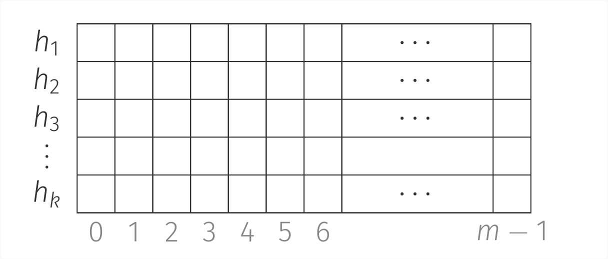
limited tracking of deletions

A generalization for tracking occurrence counts of items is known as the **count-min sketch**
A space-efficient, dedicated technique, for counting distinct values is the **Flajolet–Martin algorithm**

Count-min sketch

A count-min sketch consists of

- a numerical array C with m columns and k rows, where elements of each row are indexed from 0 to $m-1$
- a set of k pairwise-independent hash functions $h_1, \ldots, h_k$ mapping elements from the data stream to an integer in $[0, m-1]$ uniformly at random, one for each row



Count-min sketch

Theorem
Let $E(v)$ be the estimate of the occurrence count of item v from a count-min sketch of size $k \times m$. Let n_T be the sum of occurrences counts of all items (number of elements received so far), and $G(v)$ the true occurrence count of item v. Then, with probability at least $1 - e^{-k}$

$$E(v) \leq G(v) + \frac{n_T \cdot e}{m} .$$

Counting distinct values

Estimating the number of distinct values (i.e. number of items)
How many distinct email addresses appear in the data stream?

Bloom filters can be used to count distinct values

The Flajolet–Martin algorithm provides a space-efficient alternative when set-membership queries are not required

Flajolet–Martin algorithm

Hash function h maps each element to a bit-string

The number of distinct values can be estimated by choosing sufficiently large bit-strings, so that there are more possible results of the hash function than there are values in the domain

Computing moments

The **Alon–Matias–Szegedy (AMS) sketch** provides an estimate of the second-order moment when it is not possible to store the occurrence counts for all distinct values

Alon–Matias–Szegedy sketch

Each sketch component Q_i is associated with a 4-wise independent hash function mapping elements from the data stream to a binary value $r_x^{(i)} \in \{-1, 1\}$ at random

$$Q = \sum_{v \in V} c_v \cdot r_v \qquad E[Q^2] = F_2 \qquad \mathrm{var}(Q^2) \leq 2F_2^2$$

A tighter estimate can be obtained by combining the *m* sketch components using the *mean-median trick*

Synopsis data structures

reservoir sampling	generic queries
Bloom filters	testing set-membership
Flajolet–Martin algorithm	counting distinct values
count-min sketch	counting item occurrences
Alon–Matias–Szegedy sketch	estimating 2nd-order moment

Finding frequent items

What is popular?

The count-min sketch and Alon–Matias–Szegedy sketch can be used to determine frequent items (a.k.a. *heavy-hitters*)

Sketches are generally better at estimating the occurrence counts of the more frequent items as compared to rare ones

Finding frequent items: lossy counting algorithm

The stream is divided into segments of size $w = \lfloor 1/\epsilon \rfloor$
When a new element arrives, the occurrence count of the corresponding item is updated
When a segment boundary is reached, all counts are decremented by 1, items with counts of zero are pruned

When n items have been processed
$O(n/w) = O(n\epsilon)$ segments have been processed
Any count has been decremented at most $O(n\epsilon)$ times

If $\lfloor n\epsilon \rfloor$ is added to all counts, none would be underestimated
Reporting frequent items using this overestimate, might lead to some false positives but no false negatives
The amount of false positives is adjusted by tuning ϵ

Finding frequent itemsets: lossy counting algorithm

This algorithm can be generalized for finding frequent *itemsets* by batching η segments

That is, η segments are read into memory and a frequent itemset mining algorithm is applied
The counts of occurrence are maintained for *itemsets* instead of *items*, and decremented by η after each batch

The value of η can be set depending on available memory

Combining frequent itemset mining and reservoir sampling offers an alternative approach, which is better able to adjust to concept drift

Finding frequent items

What is *currently* popular?

Use a decaying window to aggregate occurrences with decaying weights, such that older occurrences are discounted

Consider a stream of elements $x_1, x_2, \ldots x_t$, where x_1 is the first element to arrive, i.e. oldest one, and x_t the most recent one
Let γ be a small constant, e.g. 10^{-6} or 10^{-9}

The **exponentially decaying window** for this stream is

$$\sum_{i=0}^{t-1} x_{t-i}(1-\gamma)^i$$

Classification

Classification

Concept-drift makes streaming classification challenging

One simple solution is to use a sample from the data stream, e.g. obtained via decay-based reservoir sampling

The challenges of streaming are addressed during sampling
Any conventional classification algorithm can be used

The accuracy of the model might deteriorate over time
Might need to retrain periodically with latest sample
Monitor the performance of the model and trigger retraining

Ensemble methods can help address concept-drift, selecting the model that is best suited for a particular portion of the data stream

Hoeffding trees

```
tree T initially consists of a single leaf (also the root)
for each new incoming element x do
    sort x to the leaf $\ell$ of $T$ where it belongs
    update the occurrence counts in $\ell$
    label $\ell$ with the majority class among its elements
    if $\ell$ contains elements from different classes then
        generate candidate splits using the occurrence counts in $\ell$
        $s_a, s_b \leftarrow$ scores of the first and second best candidate splits
        $\epsilon \leftarrow \sqrt{R^2 \cdot \ln(1/\delta)/(2n_\ell)}$
        if $s_b - s_a > \epsilon$ then
            replace $\ell$ by the best split
            for each branch of the split do
                add a new leaf with empty occurrence counts
return T
```

Mining data streams

Mining data streams presents several challenges

high volume	**massive domain**
resources constraints	**concept-drift**

A high-quality synopsis of the data stream goes a long way

Choice of synopsis depends on the application at hand

task	what queries need to be answered
data	number and expected domain size of attributes
resources	amount of memory available, desired latency

Results are estimates, it is important to quantify their accuracy

Mining temporal data

Temporal data

sequential data	(only the order matters) *gene sequences, text*
time-series	(explicit time) *stock values, network monitoring*
regularly sampled	*stock values, weather data*
irregularly sampled	*customer transactions, system logs*
real values	*stock values, population monitoring*
symbolic values	*customer transactions, text*
univariate	*electrocardiography (ECG)*
multivariate	*electroencephalography (EEG)*

Temporal data analysis

Temporal data can be viewed as contextual data

Contextual attribute(s) provide context for the measurements, reference points e.g. *date, incremental identifier*

Behavioral attribute(s) represent the actual measurements

Multivariate time-series data

A time-series of length n and dimensionality m contains m *numerical feature values* at each of n timestamps $t_1, \ldots, t_n$.

Multivariate discrete sequence data

A discrete sequence of length n and dimensionality m contains m *discrete feature values* at each of n timestamps $t_1, \ldots, t_n$.

The data point received at time t_i is $\boldsymbol{x}^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)} \rangle$

Tasks

Clustering	group together things that look alike
Classification	identify things that exhibit prototypical behavior
Outlier detection	identify things that exhibit atypical behavior
Frequent pattern mining	find frequently occurring patterns
Forecasting	predict future behavior

Thing can be

- *data point or segment* within sequence or time-series
- *sequence or time-series* within database

Distance vs. similarity

Distance

Distance function over domain $\mathcal{R}$ $\quad \mathsf{d}\colon \mathcal{R} \times \mathcal{R} \to \mathbb{R}_+$

Distance is smaller when objects are more similar

Similarity

Similarity function over domain $\mathcal{R}$ $\quad \mathsf{s}\colon \mathcal{R} \times \mathcal{R} \to [0,1]$

Similarity equals 1 when objects are identical

Induced similarity functions

$\mathsf{s}_{\mathsf{d}} = 1 - \mathsf{d}/\Delta$ where Δ is the largest possible distance

$\mathsf{s}_{\mathsf{d}} = e^{-\mathsf{d}/\delta^2}$ where parameter δ controls the decay rate

and d does not need to be bounded

Distances and metrics

A distance function d is a metric
if and only if it satisfies the following properties

non-negativity	$\mathrm{d}(x, y) \geq 0$ (a.k.a. separation axiom)
coincidence axiom	$\mathrm{d}(x, y) = 0$ if and only if $x = y$
symmetry	$\mathrm{d}(x, y) = \mathrm{d}(y, x)$
triangle inequality	$\mathrm{d}(x, y) \leq \mathrm{d}(x, z) + \mathrm{d}(z, y)$

Some tasks can be performed more efficiently if the distance function is a metric
Some algorithms expect a distance function that is a metric

Distances: Dynamic Time Warping

Pair of sequences or time-series over domain $\mathcal{R}$

$\mathcal{S}_X = \langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n_X)} \rangle \in \mathcal{R}^{n_X}$ and $\mathcal{S}_Y = \langle \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \dots, \boldsymbol{y}^{(n_Y)} \rangle \in \mathcal{R}^{n_Y}$

Distance function over domain $\mathcal{R}$ $\quad \mathsf{d} \colon \mathcal{R} \times \mathcal{R} \to \mathbb{R}_+$

Dynamic Time Warping (DTW) finds a mapping between positions that minimizes the total distance

Distances: Dynamic Time Warping

Pair of sequences or time-series over domain $\mathcal{R}$

$\mathcal{S}_X = \langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n_X)} \rangle \in \mathcal{R}^{n_X}$ and $\mathcal{S}_Y = \langle \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \dots, \boldsymbol{y}^{(n_Y)} \rangle \in \mathcal{R}^{n_Y}$

Distance function over domain $\mathcal{R}$ $\quad \mathsf{d} \colon \mathcal{R} \times \mathcal{R} \to \mathbb{R}_+$

Dynamic Time Warping (DTW) $D_{DTW}(\mathcal{S}_X, \mathcal{S}_Y) = \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(n_X, n_Y)$
where DTW is defined recursively

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,j) = \mathsf{d}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(j)}) + \min \begin{cases} \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i, j-1) & \text{repeat } \boldsymbol{x}^{(i)} \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j) & \text{repeat } \boldsymbol{y}^{(j)} \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j-1) & \text{repeat neither} \end{cases}$$

with $\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,0) = 0$,

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,0) = \infty, \forall i > 0 \text{ and } \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,j) = \infty, \forall j > 0$$

Part IV

Mining Sequences

Distances

Distances

Pair of sequences over domain $\mathcal{R}$

$\mathcal{S}_X = \langle \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n_X)} \rangle \in \mathcal{R}^{n_X}$ and $\mathcal{S}_Y = \langle \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(n_Y)} \rangle \in \mathcal{R}^{n_Y}$

Distance function over domain $\mathcal{R}$ $\quad \mathsf{d} \colon \mathcal{R} \times \mathcal{R} \to \mathbb{R}_+$

Univariate data		Multivariate data
Sequence of items	vs.	**Sequence of itemsets**
`abacacdcac`		`C,S,C,CW,C,CW,CRW,CW,C,CW`

Distances: Dynamic Time Warping

Pair of sequences $\mathcal{S}_X$ and $\mathcal{S}_Y$, distance function $\mathbf{d}$

Dynamic Time Warping (DTW) finds a mapping between positions that minimizes the total distance

Distance $D_{DTW}(\mathcal{S}_X, \mathcal{S}_Y) = \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(n_X, n_Y)$
where DTW is defined recursively

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,j) =$$
$$\mathbf{d}(x_i, y_j) + \min \begin{cases} \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i, j-1) & \text{repeat } x_i \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j) & \text{repeat } y_j \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j-1) & \text{repeat neither} \end{cases}$$

with $\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,0) = 0$,

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,0) = \infty, \forall i > 0 \text{ and } \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,j) = \infty, \forall j > 0$$

Distances: Edit distance

Pair of sequences $\mathcal{S}_X$ and $\mathcal{S}_Y$

Edit distance finds the least expensive series of operations to transform $\mathcal{S}_X$ into $\mathcal{S}_Y$

Basic edit operations are deletion, insertion and substitution each with an associated cost, respectively c_{del}, c_{ins} and c_{sub}

One can use value-specific substitution costs, e.g. $c_{\mathrm{p}\to\mathrm{b}} < c_{\mathrm{r}\to\mathrm{b}}$
For the distance to be symmetric, each operation must have a reverse with equal cost

Levenshtein distance is the most common edit distance and a metric, obtained by setting $c_{\mathrm{del}} = c_{\mathrm{ins}} = c_{\mathrm{sub}} = 1$
Damerau–Levenshtein distance is a variant with fourth operation, transposition of two adjacent elements

Distances: Longest Common Subsequence

Pair of sequences $\mathcal{S}_X$ and $\mathcal{S}_Y$

Longest Common Subsequence (LCS) finds a longest noncontiguous subsequence occurring in both $\mathcal{S}_X$ and $\mathcal{S}_Y$

! The length $S_{LCS}(\mathcal{S}_X, \mathcal{S}_Y)$ of the longest common subsequence is a *similarity measure*

Edit distance with costs $c_{\text{del}} = 1$, $c_{\text{ins}} = 1$ and $c_{\text{sub}} = 2$ satisfies

$$D_{ED}(\mathcal{S}_X, \mathcal{S}_Y) = n_X + n_Y - 2 \cdot S_{LCS}(\mathcal{S}_X, \mathcal{S}_Y)$$

Sequence alignment

Carefully match items to find optimal alignment of sequences

Multiple sequence alignment
Computationally expensive problem
Especially important in bioinformatics, to search and compare

- amino-acid sequences in proteins
- nucleotides sequences in DNA and RNA

Needleman–Wunsch global alignment algorithm

Smith–Waterman local alignment algorithm

BLAST Basic Local Alignment Search Tool

Vector-space representations

Bag of Word vector-space representation of sequences
$\rightarrow$ position information is completely lost

Instead of single items, use short contiguous subsequences
a.k.a. *n*-grams (mostly in computational linguistics)
or *k*-mers (mostly in bioinformatics)
$\rightarrow$ position information is partially preserved

Distances

Carefully match items to find optimal alignment of sequences

vs.

Look at relative proportions of different items, ignoring order

Choice of distance/similarity measure depends on application, length and number of sequences, size of the domain

Frequent pattern mining

Frequent pattern mining

The problem of **mining frequent subsequences** can be seen as the temporal analog of frequent itemset mining

Originally for market basket analysis
Can be applied to event sequences, logs, texts, gene sequences, etc.

Subsequence

Sequence of items abacacdcac

Let $\mathcal{S}_X = x_1 \ldots x_n$ and $\mathcal{S}_Y = y_1 \ldots y_k$ be two sequences of items. $\mathcal{S}_Y$ is a subsequence of $\mathcal{S}_X$ if there is $\langle i_1, i_2, \ldots, i_k \rangle$ such that $i_1 < i_2 < \cdots < i_k$ and $y_r = x_{i_r}$ for $r = 1, \ldots, k$

vs.

Sequence of itemsets C,S,C,CW,C,CW,CRW,CW,C,CW

Let $\mathcal{S}_X = X_1, \ldots, X_n$ and $\mathcal{S}_Y = Y_1, \ldots, Y_k$ be two sequences of itemsets. $\mathcal{S}_Y$ is a subsequence of $\mathcal{S}_X$ if there is $\langle i_1, i_2, \ldots, i_k \rangle$ such that $i_1 < i_2 < \cdots < i_k$ and $Y_r \subseteq X_{i_r}$ for $r = 1, \ldots, k$

Support

Database of sequences

```
W,S,W,S,W
W,S,C,S,W,CW,CRW,CR
CRW,CW,C,CW,C,W,S,W,S,C,S
⋮
```

The support of subsequence $\mathcal{S}$ in database $\mathcal{D}$ is the number of sequences in $\mathcal{D}$ that contain $\mathcal{S}$ as a subsequence

vs.

Single sequence

```
W,S,W,S,W,S,C,S,W,CW,CRW,...
```

The support of subsequence $\mathcal{S}$ in sequence $\mathcal{D}$ is the number of occurrences of $\mathcal{S}$ in $\mathcal{D}$

! Variations in terminology: support, support set, frequency

Sequential pattern mining

Given a dataset $\mathcal{D}$, which can be either a single long data sequence or a database containing multiple sequences and a minimum support threshold θ
the problem of sequential pattern mining is to determine all subsequences whose support with respect to $\mathcal{D}$ is at least θ

GSP algorithm

```
$k \leftarrow 1$
$\mathcal{F}_k \leftarrow \{\text{all frequent items}\}$
while $\mathcal{F}_k \neq \emptyset$ do
    Generate $\mathcal{C}_{k+1}$ by joining pairs of sequences from $\mathcal{F}_k$
    $\mathcal{F}_{k+1} \leftarrow \{\mathcal{S} \in \mathcal{C}_{k+1}, \text{supp}_{\mathcal{D}}(\mathcal{S}) \geq \theta\}$
    $k \leftarrow k + 1$
return $\bigcup_i \mathcal{F}_i$
```

This is a level-wise algorithm, enumerating subsequences in order of increasing length

Since support counting is expensive, candidates must be generated carefully to ensure both exhaustivity and efficiency

Markov models

Modeling sequences

Consider a sequence $\mathcal{S} = s_1 s_2 \ldots s_i \ldots s_n$

The generative probability of the sequence is

$$\mathrm{P}(s_1 \ldots s_n) = \mathrm{P}(s_1) \cdot \mathrm{P}(s_2 \,|\, s_1) \ldots \mathrm{P}(s_n \,|\, s_1 \ldots s_{n-1})$$

Short memory property

For a sequence $\mathcal{S} = s_1 \ldots s_i \ldots$, the probability $\mathrm{P}(s_i \,|\, s_1 \ldots s_{i-1})$ is well approximated by $\mathrm{P}(s_i \,|\, s_{i-k} \ldots s_{i-1})$ for some small value of k

To reliably predict the next element in the sequence, we only need to look at the last few, most recent elements

Modeling sequences

Build a model $\mathcal{M}$ that can estimate $\mathrm{P}_{\mathcal{M}}(s \mid S)$ for any element s and sequence S of size k
$\rightarrow$ Compute the probability of arbitrary sequences
$\rightarrow$ Compute the probability of arbitrary elements conditioned on recent subsequence

Markov chains

Represent the sequence generation process with state transitions in a Markov chain defined over an alphabet Σ consisting of subsequences of size k

first order model	last element in the sequence
second order model	last two elements in the sequence
k**th order model**	last k elements in the sequence

Finite number of states: special kind of *finite state automaton*

Discrete-time: at each time step the system moves from one state to the next

Markov chains

A Markov chain is defined by

Alphabet Σ, the set of distinct states of the system

Initial probabilities $\boldsymbol{\pi}$, π_i is the probability to start in state i

Transition probabilities A, a_{ij} is the probability to move from state i to state j

$\boldsymbol{\pi}$ and A are row-stochastic, $\sum_{i \in \boldsymbol{\Sigma}} \pi_i = 1$ and $\sum_{j \in \boldsymbol{\Sigma}} a_{ij} = 1$

The probabilities in $\boldsymbol{\pi}$ and A are estimated from counts over the original sequence dataset

Markov chains vs. hidden Markov models

Markov chains (MC)

The state of the system is *fully observable*

The correspondence between the current state of the system and the generated sequence element is *deterministic*

Hidden Markov models (HMM)

The state of the system is *partially observable*

The correspondence between the current state of the system and the generated sequence element is *probabilistic*

Hidden Markov models

A hidden Markov model is defined by

States alphabet Q, the set of distinct states of the system

Observations alphabet Σ, the set of distinct observations

Initial probabilities $\boldsymbol{\pi}$, π_i is the probability to start in state i

Transition probabilities A, a_{ij} is the probability to move from state i to state j

Emission probabilities B, b_{ij} is the probability that state i generates observation j

$\boldsymbol{\pi}, A$ and B are row-stochastic, $\sum_{i \in Q} \pi_i = 1$, $\sum_{j \in Q} a_{ij} = 1$ and $\sum_{j \in \boldsymbol{\Sigma}} b_{ij} = 1$

Hidden Markov models

There are three fundamental problems associated with HMMs

Evaluation

Given model $\mathcal{M}$ and observation sequence O
determine $\mathrm{P}_{\mathcal{M}}(O)$

Explanation

Given model $\mathcal{M}$ and observation sequence O
determine the most likely state sequence X

Training

Given observation sequence O and set of states Q
determine the model $\mathcal{M}$ that maximizes $\mathrm{P}_{\mathcal{M}}(O)$

Part V

Mining Time-Series

Data Preparation

Data preparation

- Linear interpolation
- Binning, a.k.a. piecewise aggregate approximation (PAA)
- Moving-average smoothing
- Exponential smoothing
- Range-based normalization
- Standardization
- Discretization
- Symbolic aggregate approximation (SAX)

Transforms

Discrete wavelet transform (DWT)

For simplicity, assume the length n of the series is a power of 2

The decomposition defines 2^{k-1} weights of order k, for $k = 1, \ldots, \log_2(n)$

Let $\Psi(k, i)$ be the i^{th} weight of order k, corresponding to the segment of the time-series between positions

$$\frac{(i-1) \cdot n}{2^{k-1}} + 1 \quad \text{and} \quad \frac{i \cdot n}{2^{k-1}}$$

Let $\Phi(k, i)$ be the average value of this segment

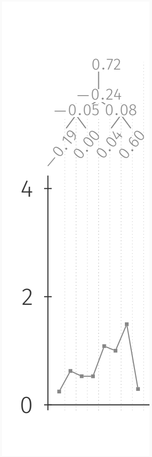
$$\Psi(k, i) = \frac{\Phi(k+1, 2i-1) - \Phi(k+1, 2i)}{2}$$

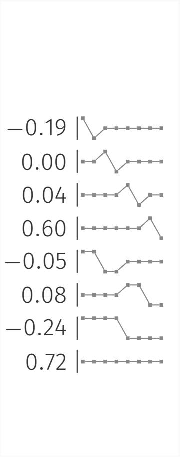
$\Phi(1, 1)$ is the global average

Discrete wavelet transform (DWT)

Each row of matrix W contains a basis vector, i.e. a wavelet

Vector $\boldsymbol{a}$ contains the weights for the different wavelets





$$\boldsymbol{a} = \begin{bmatrix} -0.19 \\ 0.00 \\ 0.04 \\ 0.60 \\ -0.05 \\ 0.08 \\ -0.24 \\ 0.72 \end{bmatrix} \quad W = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Discrete wavelet transform (DWT)

The original time-series can be reconstructed as $\boldsymbol{a}^T \cdot W$

$$\mathcal{S} = \boldsymbol{a}^T \cdot W = \sum_{i=1}^{n} a_i \boldsymbol{w}^{(i)} = \sum_{i=1}^{n} a_i \left\| \boldsymbol{w}^{(i)} \right\| \frac{\boldsymbol{w}^{(i)}}{\left\| \boldsymbol{w}^{(i)} \right\|}$$

$a_i \left\| \boldsymbol{w}^{(i)} \right\|$ are the normalized weights
$\boldsymbol{w}^{(i)} / \left\| \boldsymbol{w}^{(i)} \right\|$ are the normalized basis vectors

Discrete wavelet transform (DWT)

The original time-series can be reconstructed as $\boldsymbol{a}^T \cdot W$

$$\mathcal{S} = \boldsymbol{a}^T \cdot W = \sum_{i=1}^{n} a_i \boldsymbol{w}^{(i)} = \sum_{i=1}^{n} a_i \left\| \boldsymbol{w}^{(i)} \right\| \frac{\boldsymbol{w}^{(i)}}{\left\| \boldsymbol{w}^{(i)} \right\|}$$

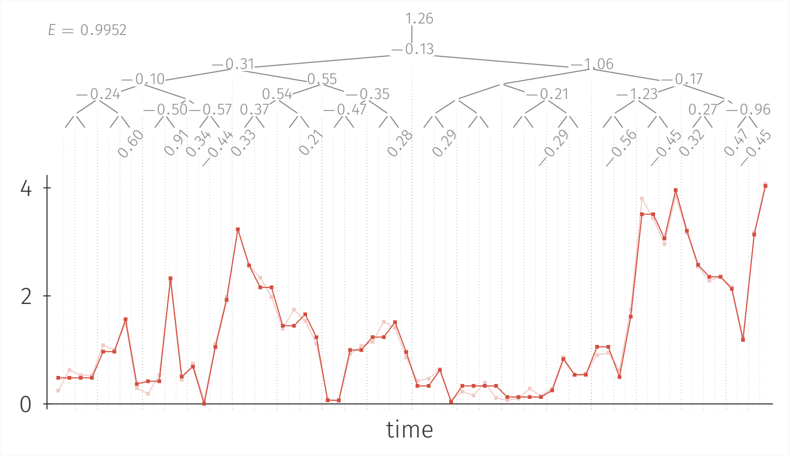
Dropping some weights reduces the dimensionality of the representation

The sum of squared normalized weights is the energy retained in the approximated time-series

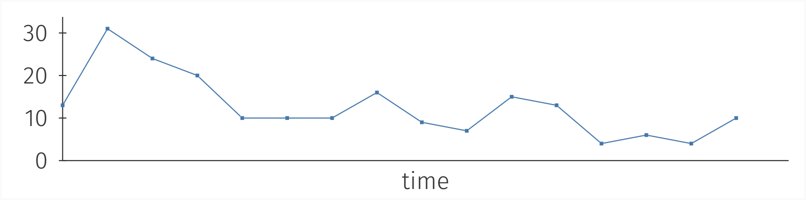
Retaining the weights with largest normalized values allows to minimize the reconstruction error

Discrete wavelet transform (DWT)

Dropping the smallest normalized weights provides a compact representation with minimum reconstruction error



Example



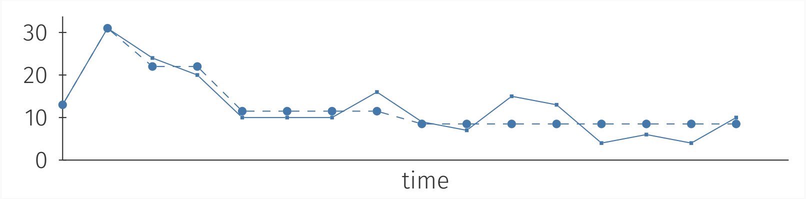
$\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle$

Discrete wavelet transform (DWT), keeping 1/4 of dimensions

Example

$$
a = \begin{bmatrix} -9.0 \\ 2.0 \\ 0.0 \\ -3.0 \\ 1.0 \\ 1.0 \\ -1.0 \\ -3.0 \\ 0.0 \\ -1.5 \\ -3.0 \\ -1.0 \\ 5.25 \\ 2.5 \\ 4.125 \\ 12.625 \end{bmatrix}
\quad
W = \begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\quad
E = \begin{bmatrix} \mathbf{162.0} \\ 8.0 \\ 0.0 \\ 18.0 \\ 2.0 \\ 2.0 \\ 2.0 \\ 18.0 \\ 0.0 \\ 9.0 \\ 36.0 \\ 4.0 \\ \mathbf{220.5} \\ 50.0 \\ \mathbf{272.25} \\ \mathbf{2550.25} \end{bmatrix}
$$

Example



$\langle 13, 31, 24, 20, 10, 10, 10, 16, 9, 7, 15, 13, 4, 6, 4, 10 \rangle$

Discrete wavelet transform (DWT), keeping 1/4 of dimensions

$$\langle 13.0, 31.0, 22.0, 22.0, 11.5, 11.5, 11.5, 11.5, \\ 8.5, 8.5, 8.5, 8.5, 8.5, 8.5, 8.5, 8.5 \rangle$$

$$\text{energy retained} = 95.55\%$$

Discrete Fourier transform (DFT)

Given a time-series $\mathcal{S}_X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$

The discrete Fourier transform decomposes the time-series into a collection of *sinusoids* with associated coefficients

Each Fourier coefficient f_k is a complex value

The original time-series can be reconstructed by summing all the weighted sinusoids

$$f_k = \sum_{r=0}^{n-1} x_r \cdot \big(\cos(2\pi rk/n) - \mathrm{i}\sin(2\pi rk/n)\big) \quad \text{for } k = 0, \ldots, n-1$$

$$x_r = \frac{1}{n} \sum_{k=0}^{n-1} f_k \cdot \big(\cos(2\pi rk/n) - \mathrm{i}\sin(2\pi rk/n)\big) \quad \text{for } r = 0, \ldots, n-1$$

i denotes the imaginary number, $\mathrm{i}^2 = -1$

Discrete Fourier transform (DFT)

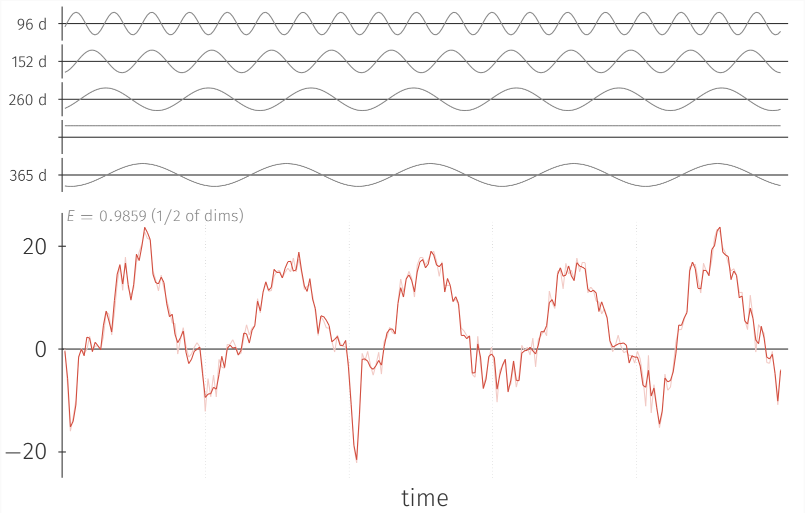
Each Fourier coefficient is a complex value $f_k = a_k + \mathrm{i}b_k$
The Fourier coefficients are such that $a_{n-k} = a_k$ and $b_{n-k} = -b_k$ for $k > 0$

Therefore, the imaginary parts in the reconstructed series cancel out
Furthermore, the $n/2$ first complex coefficients need to be retained to reconstruct the original series exactly

Dropping the coefficients with low energy $a_k^2 + b_k^2$ provides a compact approximate representation

Discrete Fourier transform (DFT)

Weekly average temperature in Kuopio from 2014 to 2018



Models for time-series

Models for univariate time-series

Given a univariate time-series $\mathcal{S}_X = \langle x_1, x_2, \ldots, x_n \rangle$, with $x_i \in \mathbb{R}$, the aim is to predict x_{n+1}

Stationarity

A stationary process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time

In a strictly stationary time-series, the probabilistic distribution of the values in any time interval $[a, b]$ is identical to that in the shifted interval $[a + \tau, b + \tau]$ for any value of the time shift τ

In a weakly stationary time-series, the mean and autocovariance are constant in time

Differencing

In some cases, the original time-series is not stationary but the difference between successive values is

Converting an original sequence into a sequence of differences is called **differencing**, e.g. first-order differencing of $\mathcal{S}_X$

$$\mathcal{S}_{X'} = \langle x'_1, x'_2, \ldots, x'_{n-1} \rangle, \text{ where } x'_i = x_i - x_{i-1}$$

Higher order differencing can also be used
e.g. second-order differencing of $\mathcal{S}_X$

$$\begin{aligned} \mathcal{S}_{X''} = \langle x''_1, x''_2, \ldots, x''_{n-2} \rangle, \text{ where } x''_i &= x'_i - x'_{i-1} \\ &= x_i - 2x_{i-1} + x_{i-2} \end{aligned}$$

For geometrically increasing series, the logarithm function is applied before differencing

Autocovariance

The **covariance** between two real-valued random variables X and Y is

$$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

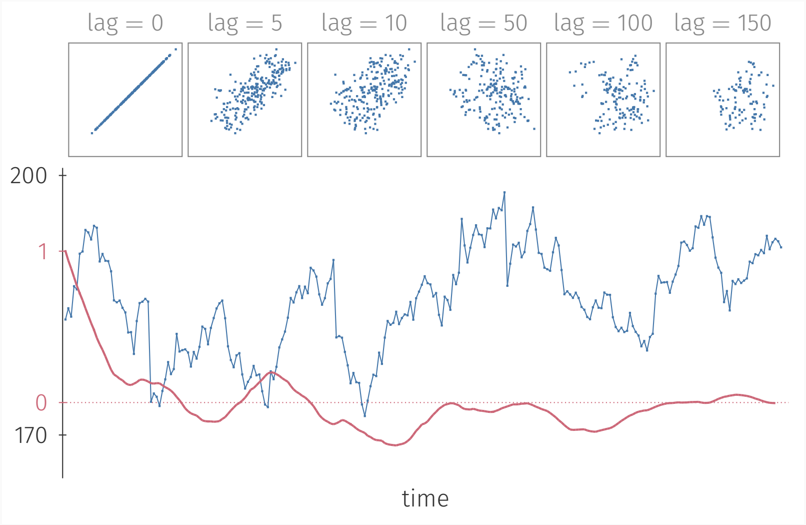
The **autocovariance** at lag τ of time-series $X = x_1, x_2, \ldots, x_n$ is the covariance between the time-series and itself shifted by τ

The **autocorrelation** at lag τ of time-series X is the normalized covariance $\text{cov}_t(X_t, X_{t+\tau}) / \text{var}_t(X_t)$ computed as

$$\text{R}_\tau(X) = \frac{(X_t - \mu_X) \cdot (X_{t+\tau} - \mu_X)}{n \cdot (X_t - \mu_X)^2}$$

Autocorrelation

IBM stock prices from Sept. 2013 to Sept. 2014



Auto-regressive models

Auto-regressive model AR(p) $$x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + c + \epsilon_i$$

Moving-average model MA(q) $$x_i = \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$

ARMA(p, q) $$x_i = \sum_{k=1}^{p} a_k \cdot x_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$

ARIMA(p, d, q) $$x'_i = \sum_{k=1}^{p} a_k \cdot x'_{i-k} + \sum_{k=1}^{q} b_k \cdot \epsilon_{i-k} + c + \epsilon_i$$

Box–Jenkins modelling procedure

Model identification

1. Use differencing to make the time-series stationary
2. Determine the most suitable model and find appropriate values for q and p
   - by looking at ACF and PACF respectively, or
   - by using Akaike's Information Criterion (AIC)

Model estimation

Estimate the parameters of the model from historical data

Model validation

Check that the model is adequate for the time-series

Models for multivariate time-series

In practice, time-series often consist of multiple variables

In addition to correlation across time, i.e. individual variables being autocorrelated, there might be significant correlations across the variables

One approach to build models for this scenario is to use *hidden variables*

The multiple input time-series are transformed into a smaller number of uncorrelated time-series, typically using principal component analysis (PCA)

A model is built for each such time-series individually
The models are used to predict hidden values, which are then mapped back into the original representation

Models for time-series

Artificial neural networks (ANN) offer a flexible alternative
e.g. long short-term Memory (LSTM) recurrent neural networks (RNN) architecture

Have fewer restrictions
Can model non-linear functions

Periodicity

Time-series might exhibit regularly recurrent, cyclic, behavior i.e. display **periodicity** (a.k.a. seasonality)

Seasonal differencing $x_i - x_{i-p}$ for some integer $p > 1$, i.e. taking the difference between values one period p apart, can be used to remove the effect of seasonality

Periodicity

Given a time-series $\mathcal{S}_X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$

The discrete Fourier transform decomposes the time-series into $n-1$ periodic sinusoidal components

$$x_r = \frac{1}{n} \sum_{k=0}^{n-1} f_k \cdot \big(\cos(2\pi rk/n) - \mathrm{i} \sin(2\pi rk/n) \big) \quad \text{for } r = 0, \ldots, n-1$$

The k^{th} component, corresponding to coefficient $f_k = a_k + \mathrm{i} b_k$, has periodicity n/k and amplitude $\sqrt{a_k^2 + b_k^2}$

If a component has a high amplitude compared to the others, the entire series will be dominated by its periodic behavior

Only components such that $k \in [\beta, n/\alpha]$ have period at least $\alpha \geq 2$ and appear at least $\beta \geq 2$ in the series

Part VI

Mining spatial data

Spatial data

grid data	(only the order matters) *image*
geo-located data	(explicit location) *demographic records*
regularly sampled	*magnetic resonance imaging (MRI), positron emission tomography (PET)*
irregularly sampled	*disease outbreaks, forest fires*
real values	*surface temperature*
symbolic values	*landcover records*
spatial	*image, topographic records*
spatio-temporal	*video, surface temperature, GPS traces*

Spatio-temporal data

Spatio-temporal data can be viewed as contextual data

Contextual attributes provide context for the measurements, reference points e.g. *date and geographic coordinates, incremental identifiers*

Behavioral attribute(s) represent the actual measurements

The dataset consists of n data points

$$\mathcal{D} = \langle (\boldsymbol{p}^{(1)}, \boldsymbol{x}^{(1)}), (\boldsymbol{p}^{(2)}, \boldsymbol{x}^{(2)}), \ldots, (\boldsymbol{p}^{(n)}, \boldsymbol{x}^{(n)}) \rangle$$

where $\boldsymbol{x}^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)} \rangle$ and $\boldsymbol{p}^{(i)} = \langle p_1^{(i)}, p_2^{(i)}, \ldots, p_c^{(i)} \rangle$ contain the values of the m behavioral attributes and of the c contextual attributes, respectively, for the i^{th} data point

Measuring distances

The distance between the locations of two data points *i* and *j*, $\mathrm{d}(p^{(i)}, p^{(j)})$, might be measured using e.g. Euclidean or Manhattan distance

Coordinates might be provided as latitude and longitude
! The length of a degree of longitude varies with the latitude
Distances might be best measured using the *great circle distance* (a.k.a. orthodromic distance)

Interpolation

Interpolation can be used to produce a dataset with equally spaced coordinates, i.e. arranged along a grid

Map datasets from different grids, e.g. with different resolutions, to common grid

Inverse distance weighting

Let v_p denote the value at the point with coordinates $\boldsymbol{p}$

Given a sample of point coordinates P for which the values are known, the value at coordinates $\boldsymbol{q}$ is estimated as

$$v_q = \begin{cases} v_p & \text{if } \exists\, \boldsymbol{p} \in P, \mathrm{d}(\boldsymbol{q}, \boldsymbol{p}) = 0 \\ \frac{\sum_{p \in P} v_p / \mathrm{d}(\boldsymbol{q}, \boldsymbol{p})}{\sum_{p \in P} 1 / \mathrm{d}(\boldsymbol{q}, \boldsymbol{p})} & \text{otherwise} \end{cases}$$

Density estimation

Considering discrete attribute j and a value a in its domain, we collect in P the coordinates of data points that are occurrences of the corresponding item, i.e.

$$P = \{\boldsymbol{p}^{(i)} \text{ for } i = 1 \dots n, \text{ such that } \boldsymbol{x}_j^{(i)} = a\}$$

Kernel density estimation methods produce density profiles, similarly to histogram techniques, but applying smoothing

The density of the item at coordinates $\boldsymbol{q}$ is estimated as

$$v_{\boldsymbol{q}} = \frac{1}{|P|} \sum_{\boldsymbol{p} \in P} K_h(\boldsymbol{q}, \boldsymbol{p})$$

using for instance the Gaussian kernel of width h

$$K_h(\boldsymbol{q}, \boldsymbol{p}) = \frac{1}{(\sqrt{2\pi} \cdot h)^c} e^{-\|\boldsymbol{q}-\boldsymbol{p}\|_2^2/(2h^2)}$$

Triangulation

The **Delaunay triangulation** and the **Voronoi diagram** of a set of points *P* can be used to find the neighbors of a point, compute interpolated values, turn the data into a graph, etc.

They have multiple applications in a wide range of fields

Contours and edges

Compute value differences across neighboring points to identify areas at which value changes sharply

Edge detection methods aim at detecting points in an image at which value changes sharply

A *contour line* or *isoline* of a function of two variables is a curve along which the function has constant value

Consider a behavioral attribute as function of the coordinates
Contours are typically plotted for values spaced regularly across the domain of the attribute
Close contours indicate steep slopes, i.e. regions where the value of the attribute changes sharply

Shapes to time-series

The contour of a shape can be transformed into a time-series
Measure the distance from the centroid of the shape to its boundary, doing a clockwise sweep
E.g. taking 360 different regularly spaced angular samples produces a series of 360 numerical values

The time-series is referred to as the **centroid distance signature** of the shape

Rotations of the shape result in cyclic translation of the series
Mirror images of the shape result in a reversal of the series
Need to be taken into account in the analysis process

Discrete wavelet transform (DWT)

For numerical data arranged into a grid, spatially adjacent values are often very similar, storing all the values is wasteful, redundant

The discrete wavelet transform can be generalized to multiple contextual attributes
Differencing is applied across contiguous areas of the grid
Division is performed while alternating between the axes of the grid, i.e. the contextual attributes

Trajectory data

Object tracking

The position of a vehicule, robot, person, animal, etc. can be recorded over time through a variety of means, including the global positioning system (GPS), video, wireless triangulation, radio frequency identification (RFID)

A **trajectory** is a time-series of geo-locations

Time is the contextual attribute

Spatial coordinates constitute behavioral attributes

Transform a trajectory into multidimensional data

Compute the discrete wavelet transform coefficient for each spatial coordinate separately

Combine coefficients vectors across the different coordinates

Trajectory data

Like other time-series, trajectories can be compared using the **dynamic time warping distance (DTW)**

$D_{DTW}(\mathcal{S}_X, \mathcal{S}_Y) = \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(n_X, n_Y)$
where DTW is defined recursively

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,j) = \\ \mathrm{d}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(j)}) + \min \begin{cases} \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i, j-1) & \text{repeat } \boldsymbol{x}^{(i)} \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j) & \text{repeat } \boldsymbol{y}^{(j)} \\ \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i-1, j-1) & \text{repeat neither} \end{cases}$$

with $\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,0) = 0$,

$$\mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(i,0) = \infty, \forall i > 0 \text{ and } \mathrm{DTW}_{\mathcal{S}_X,\mathcal{S}_Y}(0,j) = \infty, \forall j > 0$$

where $\mathrm{d}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(j)})$ is the distance between the position at time i in trajectory $\mathcal{S}_X$ and the position at time j in trajectory $\mathcal{S}_Y$

Frequent trajectory patterns

A key problem in analysing trajectories is to **identify frequent sequential paths**

1. Transform a trajectory into a univariate discrete sequence through grid-based discretization
2. Apply a sequential pattern mining algorithm (e.g. GSP) to the sequence(s)

Spatial tile transformation
Discretize each coordinate and assign a symbol to each interval
Each tile is identified by the combination of symbols along the different dimensions
Build the sequence associated to a trajectory by listing the identifiers of the tiles it traverses

Frequent trajectory patterns

A key problem in analysing trajectories is to **identify frequent sequential paths**

1. Transform a trajectory into a univariate discrete sequence through grid-based discretization
2. Apply a sequential pattern mining algorithm (e.g. GSP) to the sequence(s)

Spatio-temporal tile transformation
Divide the time range into intervals and assign them identifiers
For a given trajectory, list for each time interval the identifiers of the tiles in which at least a chosen amount of the interval was spent, tagged with the corresponding interval identifier

Part VII

Outlier Analysis

Basics

What is an outlier?

An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.

D. M. Hawkins, 1980

What is an outlier?

Outliers can be seen as a complementary concept to clusters

Clusters are groups of data points that are similar
Outliers are individual data points that are not similar to the rest of the data

Outliers are also known as anomalies, abnormalities, discordants or deviants

Applications

Credit card fraud detection

Quality control and fault detection

Web log analytics and intrusion detection

Medicine and public health

Sports statistics

...

Swamping and masking

Swamping happens when the number of normal instances increases or they become scattered so that normal instances are wrongly identified as outliers

Masking happens when the number of outliers increases, forming dense clusters of anomalous data points and concealing their own presence

Both issues are consequences of too large amounts of data used for the detection of outliers
This can be solved by using subsampling

Analysis approaches

Reference set with respect to which normality is evaluated

Global approaches

The reference set contains all other data points

Assumption: single normal generating mechanism

Drawback: other outliers in the reference set may falsify results

Local approaches

The reference set consists of a selected subset of data points

No *assumption* on number of normal generating mechanisms

Drawback: relies on appropriate choice of reference subset

Some approaches let the reference set vary from a single data point (local) to the entire dataset (global) automatically or depending on a user-defined parameter

Various detection methods

Depth-based methods	
Deviation-based methods	
Information-theoretic methods	
Density-based methods	using histograms
	using fixed radius neigborhood
Statistical tests	extreme values
Distance-based models	k-NN distance
	local outlier factor (LOF)
	instance-specific Mahalanobis distance

Univariate extreme values

Assuming a univariate Gaussian distribution, the parameters are estimated as the mean μ and standard deviation σ over all data points in $\mathcal{D}$

The probability density function of the Gaussian distribution is

$$f_{\mathcal{D}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

For a data point x the *standardized* value $z = (x - \mu)/\sigma$ is called its z-**number**

Points in the lower tail correspond to large negative z-numbers
Points in the upper tail correspond to large positive z-numbers

Univariate extreme values

The probability density function can be written in terms of the z-number

$$f_{\mathcal{D}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-z^2}{2}}$$

Hence, the cumulative Gaussian distribution can be used to determine the area of the tail that is more extreme than z
When the number of available data points n is limited, Student t-distribution with n degrees of freedom is used instead

Points are typically declared outliers if the absolute value of their z-number is greater than 3
i.e. if they deviate more than 3 times the standard deviation from the mean

Mahalanobis distance

The **Mahalanobis distance** from data point $\boldsymbol{x}$ to a distribution with mean $\boldsymbol{\mu}$ and covariance Σ is

$$D_{\Sigma}(x, \boldsymbol{\mu}) = \sqrt{(x - \boldsymbol{\mu})^T \Sigma^{-1} (x - \boldsymbol{\mu})}$$

Can be seen as a multidimensional extension of the z-number, measuring the number of standard deviations by which the data point differs from the mean of the distribution

Computing the Mahalanobis distance is equivalent to computing the Euclidean distance after rotating the data to the principal directions and dividing each of the transformed coordinate by the corresponding standard deviation

Multivariate extreme values

The probability density function can be written in terms of the Mahalanobis distance

$$f_{\mathcal{D}}(\boldsymbol{x}) = \frac{1}{\sqrt{\det(\Sigma) \cdot (2\pi)^m}} e^{-(D_\Sigma(\boldsymbol{x},\boldsymbol{\mu}))^2/2}$$

Each of the independent component of the Mahalanobis distance can be modeled as a one-dimensional standard normal distribution $\mathcal{N}(0, 1)$

The sum of squares of m such variables follows a χ^2 distribution with m degrees of freedom

The cumulative probability of the region of the χ^2 distribution with m degrees of freedom for which the value is greater than $D_\Sigma(\boldsymbol{x}, \boldsymbol{\mu})$ can be reported as the extreme value probability of $\boldsymbol{x}$

Clustering models

Assumption: clustering aims at finding groups of similar points, whereas outliers are not similar to the rest of the data

Assuming that k clusters have been detected
The Mahalanobis distance from point $\boldsymbol{x}$ to the j^{th} cluster, having mean μ_j and covariance matrix Σ_j, is

$$D_{\boldsymbol{\Sigma}_j}(\boldsymbol{x}, \boldsymbol{\mu}_j) = (\boldsymbol{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)$$

Report $\min_{j=1,\ldots,k} D_{\boldsymbol{\Sigma}_j}(\boldsymbol{x}, \boldsymbol{\mu}_j)$ as outlier score of point $\boldsymbol{x}$

Distance-based models: k-NN distance

Assumption: outliers are not similar to the rest of the data, i.e. they are far apart from their neighbors

Report the distance from a point to its k-nearest neighbor as the outlier score

Distance-based models: k-NN distance

Distance-based models have a finer granularity than clustering models, but it comes at the cost of higher computational complexity

Computing the k-nearest neighbor distance requires $O(n)$ time for each data point when a sequential scan is used,
i.e. $O(n^2)$ time for the entire dataset, which is not scalable

Early termination

Two steps method with sample

Distance-based models: k-NN distance

The k-NN distance is sensitive to the neighborhood density
Need for corrections to account for local variations in density

Local outlier factor (LOF)
Normalizes distances with average local density
Sometimes seen as a density-based method
Sometimes as a distance-based method
Both types of methods rely on proximity

Local outlier factor

Let $\Delta_k(x)$ denote the distance from $\mathbf{x}$ to its k nearest neighbor
Let $N_k(x)$ denote the points within distance $\Delta_k(x)$ of $\mathbf{x}$

$$R_k(x, x') = \max(\mathrm{d}(x, x'), \Delta_k(x'))$$

$$AR_k(x) = \frac{1}{|N_k(x)|} \sum_{x' \in N_k(x)} R_k(x, x') \quad LOF_k(x) = \frac{1}{|N_k(x)|} \sum_{x' \in N_k(x)} \frac{AR_k(x)}{AR_k(x')}$$

Typically, LOF_k values for points in a cluster are close to 1 if the points are distributed homogeneously
Points with $LOF_k \gg 1$ are reported as outliers

In practice, determine the best neighborhood size k by taking the maximum LOF_k over a range of values

Instance-specific Mahalanobis distance

Determine the k-neighborhood of point $\mathbf{x}$ following an agglomerative approach

```
$N \leftarrow \{\mathbf{x}\}$
for $i = 1, \dots, k$ do
    $N \leftarrow N \cup \{\arg\min_{\mathbf{x}' \in \mathcal{D} \setminus N} \min_{\mathbf{u} \in N} \mathrm{d}(\mathbf{x}', \mathbf{u})\}$
return $N$
```

Use $D_{\boldsymbol{\Sigma}_N}(\mathbf{x}, \boldsymbol{\mu}_N)$ as outlier score for point $\mathbf{x}$, with $\boldsymbol{\mu}_N$ and Σ_N respectively the mean and covariance matrix of the k-neighborhood N of $\mathbf{x}$, i.e. the Mahalanobis distance that accounts for the local covariance structure

High-dimensional data

High-dimensional approaches

As dimensionality increases the distances between pairs of points become more similar, outliers become increasingly more difficult to tell apart from normal points

High-dimensional approaches

Outliers typically present anomalous behavior only in a small subset of attributes while other dimensions are irrelevant to the anomaly detection process

Subspace outlier detection

An outlier is defined in association with one or more subspaces that are specific to it

Consider projections into lower dimensional subspaces to detect associated outliers

Grid-based sparsity coefficient

Partition each attribute into p bins containing each an equal fraction $f = 1/p$ of data points

Selecting k attributes and one bin from each defines a k-dimensional grid cell or cube

The sparsity coefficient for cube $\mathcal{R}$ containing $n_{\mathcal{R}}$ data points is

$$S(\mathcal{R}) = \frac{n_{\mathcal{R}} - n \cdot f^2}{\sqrt{n \cdot f^k \cdot (1 - f^k)}}$$

A negative sparsity coefficient indicates that the number of points in the cube is significantly lower than expected

Genetic algorithm for subspace outliers

- The process starts with a population of q random individuals and iteratively repeats the process of selection, crossover, mutation
- Individuals in the population progressively improve in fitness and become more similar
- A position in the encoding has converged when a predefined fraction of the population has the same value for that position
- The population has converged when all positions in the encoding have converged

- Keep track of the best solutions encountered, i.e. cubes with most negative sparsity coefficients
- Data points contained in those cubes are reported as outliers

Isolation-based methods: Isolation trees

Assumption: outliers are few, not similar to the rest of the data and located in sparse regions, hence suceptible to isolation

Grow binary decision trees at random until all distinct data points are in a node of there own

Data points that are reached via short paths are reported as outliers

Temporal data

Outliers in temporal data

In the context of temporal data, *outlier detection* is also known as *event detection*, especially when performed in real-time

A sudden change at a given timestamp of a time-series or sequence is referred to as **contextual outlier** or **point outlier**

An anomalous pattern of consecutive data points is referred to as **collective outlier**, as well as **shape outlier** in the context of time-series and **combination outlier** in the context of discrete sequences

Point outliers

The detection of point outliers is closely related to forecasting

A data point is considered an outlier if it deviates significantly from its forecasted, i.e. expected, value

Combination outliers

The aim is to identify unusual combinations of values appearing in a sequence

Small windows of a chosen size, referred to as *comparison units*, are extracted from the sequence
Distances between comparison units can be computed using e.g. dynamic time warping (DTW) distance, edit distance, etc.

The k-nearest neighbor distance can be used as outlier score

Shape outliers: HOTSAX

Shape outliers are defined over windows of the time-series
Distance to *k*-nearest neighbors is used as outlier score

1. Extract all candidates by sliding a window of length *w* over the time-series
2. Compute the Euclidean distance from each candidate to all other non-overlapping windows
3. Report candidates with highest *k*-nearest neighbor distance as outliers

Use non-overlapping windows to prevent trivial matches
Pruning and early termination are used to improve efficiency