
*This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 825215 (Topic: ICT-35-2018 Type of action: CSA). All material presented here reflects only the authors' view. The European Commission is not responsible for any use that may be made of the information it contains.*

MUNI

MUNI

# Text Mining with Machine Learning

FINancial supervision and TECHnology compliance training program – SUPTECH WORKSHOP IV

František Dařena

# Access to R Server

join at https://uem1.euba.sk/rstudio/auth-sign-in

login: **cnb##** (numbers from 02 to 15)

password: **fintech01**

MUNI

# Knowledge Discovery in Texts

Introduction to Text Mining
and
Examples of Applications

# Web 2.0

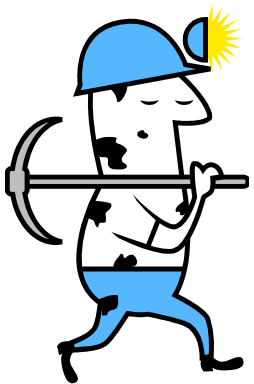
- people don’t view the content of Web pages passively, but they create it (social networks, blogs, wikis, folksonomies, …)
- usability and interoperability enable access and content creation anywhere and at any time to many people
- huge amount of data is available
- about 80% is in textual form

# Available textual content

- often freely available (newspapers, blogs, company sites, discussion boards, social networks, reviews sites, …)
- very useful for many purposes (contextual advertising, customer care, hiring personnel, opinion mining, market research, political campaigns, fight against cybercrime, …)

# Problems

- large volumes of data
- unstructured, not very suitable for machines
- processing by humans is not feasible
- automation of revealing hidden knowledge in text data => discipline known as **text mining**



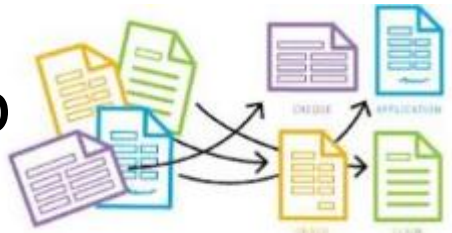
# Text mining

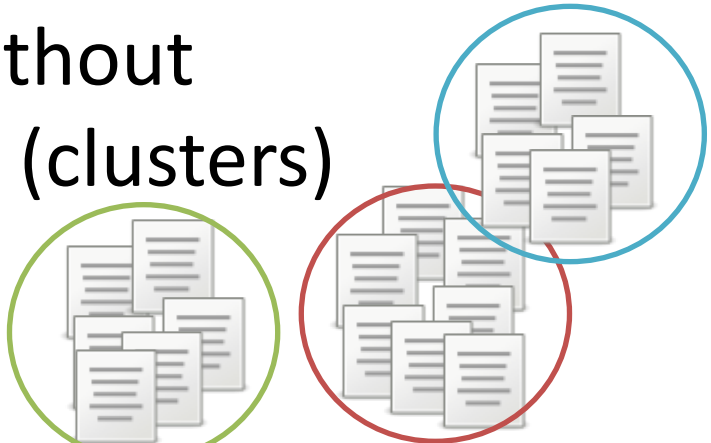
- knowledge discovery discipline
- closely related to data mining  (also known ad text data mining)

| **econometrics** | **vs**. | **data mining** |
| --- | --- | --- |
| assumes that the data are generated by a given stochastic data model | | uses algorithmic models and treats the data mechanism as unknown |
| starts with a model and tries to find its parameters | | starts with data and tries to find a model that well fits (and predicts) the data |

# Typical tasks

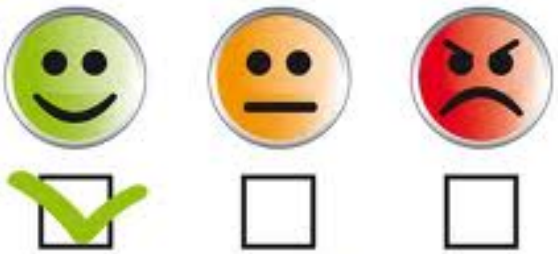
- classification – assigning documents to one or more predefined classes according to the training examples
- information retrieval – retrieving documents matching a query
- clustering – grouping documents according to their similarity without prior knowledge of the groups (clusters)





# Typical tasks

- summarization – selecting the most important information from one or more documents
- finding associations among document parts
- sentiment analysis and opinion mining

# Typical tasks

- information extraction – finding structured information in unstructured texts (e.g., accounting data in annual reports)
- question answering – understanding questions in a natural language, extracting or formulating answers
- machine translation – computers translate texts from one language to another; statistical MT with incorporation of human experts is the most popular today


# Data mining vs. Text mining

- some of the tasks are quite similar to the ones of data mining
- data mining
  - a well developed branch of science
  - focuses on finding interesting patterns and hidden knowledge in large amounts of data
  - involves the following common tasks: anomaly detection, association rule learning, clustering, classification, regression, summarization
  - works with highly structured data (tabular format)

# Data mining vs. Text mining

- text mining
  - focuses on tasks similar to those in DM – why not to use the same methods?
  - textual data is unstructured and complicated for understanding (grammar, several meanings of a word, order of words, dictionary size, …)
  - documents must be converted to a representation suitable for a selected DM algorithm

# Data mining vs. Text mining

- text mining is related to disciplines such as statistics, artificial intelligence, machine learning (like data mining), but also to natural language processing, linguistics, speech recognition and generation, …

# Specifics of text mining

- large input space – many potential examples, huge number of words and their combinations
- sparse vectors representing the documents – the number of features in one document is small compared to the number of all features
- little training data – because of demandingness of the labeling process
- noise – spelling errors, typos, wrong grammar etc., typical for natural languages, also incorrect labeling

# Specifics of text mining

- a small fraction of the content is often relevant
- the distribution of the probability with which the words appear in documents is strongly skewed

# Text Mining Process

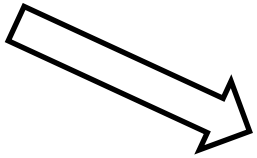
- defining the problem
- collecting the necessary data
- defining features
- analyzing the data
- interpreting the results

# Machine Learning for Text Mining

- it is not possible to cover all possible problems (inputs, situations) in advance
- past experience can be used to solve problems in the future
- machines need an ability to learn – a model to be applied in the future (in unknown situations) is trained (learned) as a generalization of past experience represented by data

# Machine Learning for Text Mining

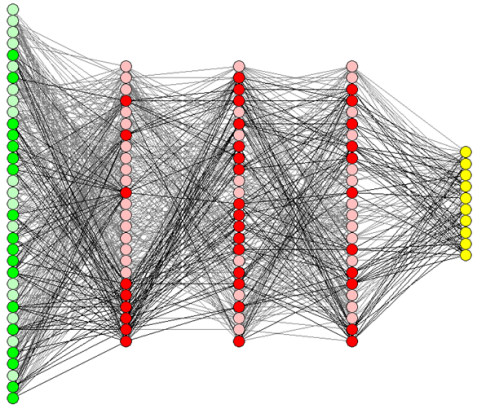
# Machine Learning for Text Mining



available documents (training data, experience)

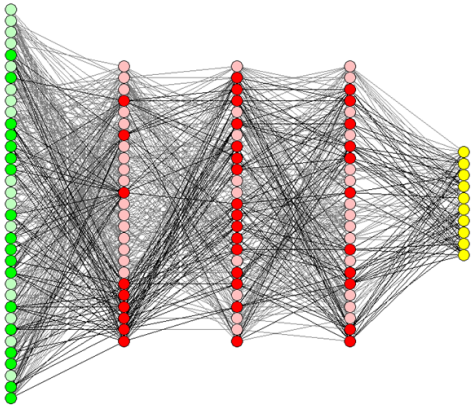
# Machine Learning for Text Mining

learning (training, generalization, induction, model creation)

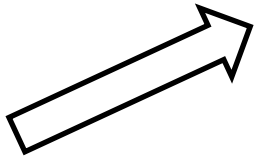


available documents (training data, experience)

# Machine Learning for Text Mining

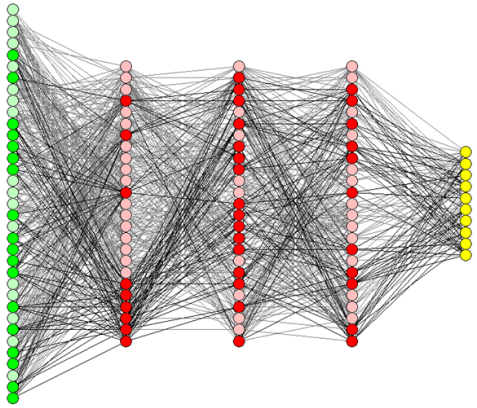


new (previously unseen) problem



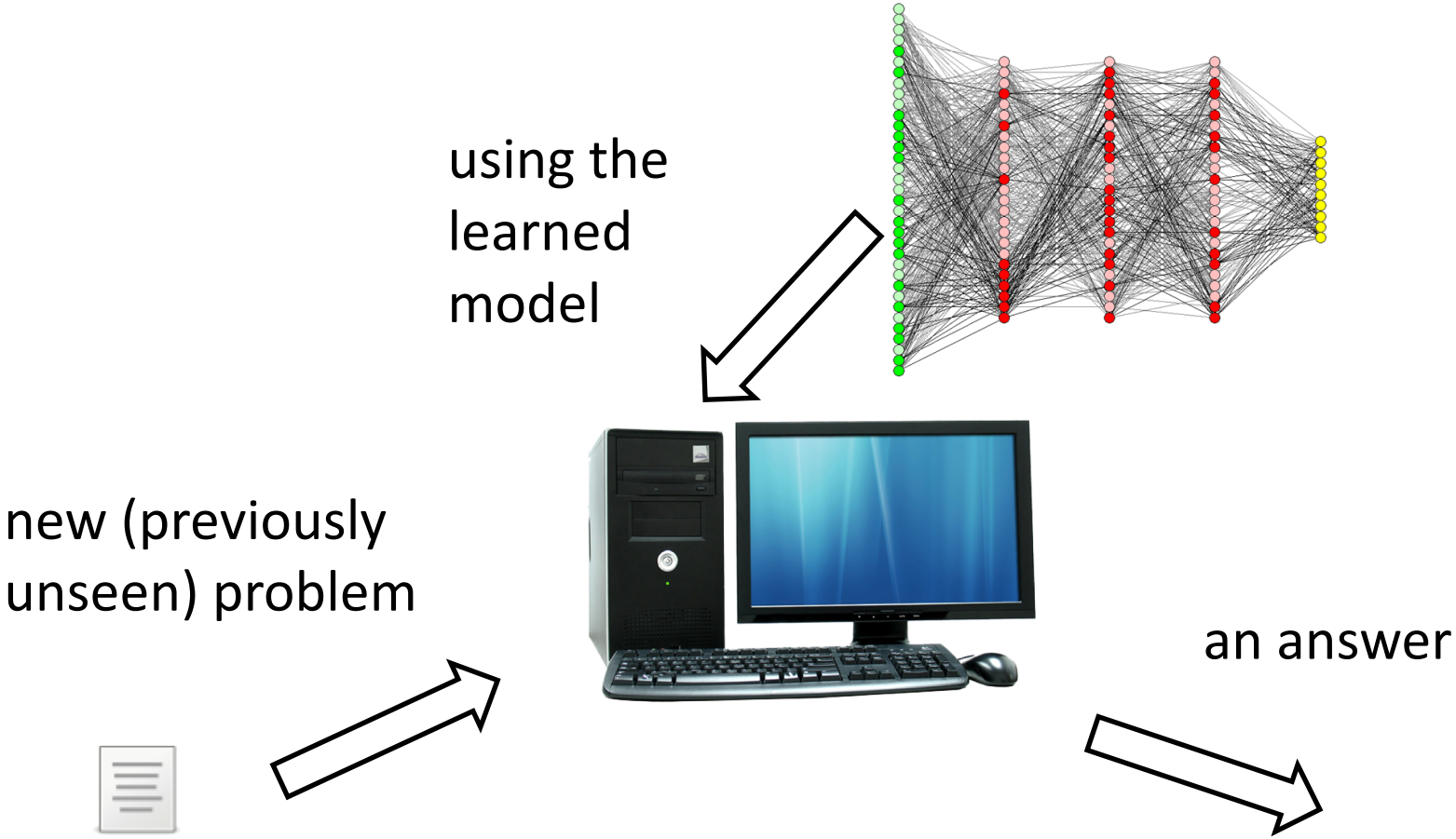
# Machine Learning for Text Mining

using the learned model



new (previously unseen) problem

# Machine Learning for Text Mining



# Machine Learning

- supervised – availability of “correct answers” (target variables, labels)
- unsupervised – no correct answers are available
- semi-supervised – a few labeled examples contain initial information for learning from many unlabeled examples

# Some typical tasks – more details

# Summarization

- producing a new text from *one or more* texts
- the new text conveys the most *important information* from the original text(s) and it is significantly *shorter*
- the new text can have the form of a set of key phrases, set of most important sentences, a linguistically correct abstract, …

# Summarization

- four basic approaches
  - extraction – identifying and reproducing most important parts
  - abstraction – producing most important parts in a new way
  - fusion – combining extracted document parts
  - compression – throwing out unimportant sections

# Summarization – extraction

- early attempts focused on
  - extracting sentences containing the most frequent words (considering also distance of the words)
  - position of sentences in paragraphs (the most informative is often at the beginning)
  - sentences containing some of the predefined words
  - parts defining document structure (e.g., title, headings)

# Summarization – extraction

- further research focused on using machine learning using features such as
  - number of words
  - first occurrence
  - frequency
  - length in characters
  - presence of a part of speech (verb, adjective, …)
- supervised approaches use existing data, such as newspaper articles, web pages, etc. with available abstracts or keywords (keyphrases)

# Summarization – abstraction

- can be characterized as ’top-down’ – they look for a set of pre-defined information types to include in the summary (e.g., an earthquake frame may contain slots for location, earthquake magnitude, number of casualties, etc.)
- the desired pieces of information are located and filled in

# Multi-document summarization

- major problems
  - identifying and coping with redundancy
  - recognizing novelty, contradictions
  - ensuring logical coherence
  - ensuring completeness

# Sentiment analysis

- not only objective information is important
- sentiment analysis (or opinion mining) focuses on sentiments, evaluations, attitudes, and emotions
- applications: product ranking, predicting sales performance, box-office revenues for movies, election results, stock market and many others

# Opinion definition

- an opinion is a quintuple ($e_i$, $a_{ij}$, $s_{ijkl}$, $h_k$, $t_l$)
  - $e_i$ is the name of an entity
  - $a_{ij}$ is an aspect of $e_i$
  - $s_{ijkl}$ is the sentiment (positive, negative, or neutral, or expressed with different strength/intensity levels, e.g., 1 to 5) on aspect $a_{ij}$ of entity $e_i$
  - $h_k$ is the opinion holder
  - $t_l$ is the time when the opinion is expressed by $h_k$
- SA objective – to identify all components

# Levels of analysis

- sentiment might be identified at
  - document/sentence level – overall sentiment for the entire piece of text
  - entity/aspect level – a sentiment is related to a target (*"Although the service is not that great, I still love this restaurant."*)
- two types of sentiment
  - regular opinions ("*Coke tastes very good*")
  - comparative opinions ("*Coke tastes better than Pepsi*")

# Sentiment lexicons

- the most straightforward method
  - words: positive (good, wonderful, amazing, …) and negative (bad, poor, terrible, …)
  - phrases: e.g., a cold fish (a person who does not seem very friendly and does not show their emotions)
- problems
  - *"This camera sucks"* vs. *"This vacuum cleaner really sucks"*
  - *"Can you tell me which Sony camera is good?"*
  - *"What a great car! It stopped working in two days."*
  - *"This washer uses a lot of water."*

# Machine translation

- Computer Aided Translation (CAT)
  - uses software tools to facilitate the translation process, spell checkers, terminologies, concordances, translation memories tools etc.
  - help a human translator
- Machine translation
  - performs the translation itself

# Machine Translation types

- rule-based
  - original form based on linguistic information and dictionaries
- example-based
  - based on analogy with previous translations (a segment is translated according to already translated sub-segment parts)
- statistical
  - the most popular MT branch
  - rules from huge bilingual content (training set) are inferred (using statistics)
  - the more training data is used the better results are produced
- hybrid
  - leverages the strengths of statistical and rule-based translation methodologies

# Information extraction

- finding structured information in unstructured data
- the desired concepts from a domain (entities, relationships) are known and IE focuses on identifying them in texts
- the output is like database records containing templates filled during an IE process
- different from complete natural language understanding => shallow linguistic analysis is usually enough

# IE problems

- one fact might be expressed by many different ways
- one fact might spread across more sentences, documents, or repositories
- some information is implicit

# IE tasks

- named entity recognition
  - identification and classification of predefined types of entities, such as organizations, persons, places, numerical and currency expressions, temporal expressions, etc.
- co-reference resolution
  - identification of multiple (co-referring) mentions of the same entity,
    - named: “EU” and “European Union”
    - pronominal: “John bought food. But he forgot to buy drinks.”
    - nominal: “Microsoft revealed its earnings. The company also unveiled future plans.”
    - implicit: “Marco è arrivato tardi. (he, Marco) Ha portato la birra.”

# IE tasks

- Relation extraction
  - identification of predefined relationships
    - “Steve Jobs works for Apple” => EmployeeOf(Steve Jobs,Apple)
    - “Mr. Smith gave a talk at the conference in New York” => LocatedIn(Smith,New York)
    - “Listed broadcaster TVN said its parent company, ITI Holdings, is considering various options for the potential sale” => SubsidiaryOf(TVN,ITI Holding)
    - etc.

# IE tasks

- Event Extraction
  - identifying events and finding *who did what to whom, when, where, through what methods (instruments), and why*
  - *"Masked gunmen armed with assault rifles and grenades attacked a wedding party in mainly Kurdish southeast Turkey, killing at least 44 people."* => perpetrators (*masked gunmen*), victims (*people*), number of killed/injured (*at least 44*), weapons and means used (*rifles and grenades*), and location (*southeast Turkey*)

# Open Information Extraction

- doesn't require the knowledge of relations of interest (the type of extracted information is not specified in advance)
- all possible relations are found and are then available, e.g., for querying
- based on the fact that that most relations (in English) can be characterized by a set of several lexicosyntactic patterns, e.g.,
  - *Noun phrase, Verb, Noun Phrase – e.g., Graham Bell, invented, telephone*
  - *Subject, Verb, Complement* – e.g., *Albert Einstein, is, a scientist*
- uses part-of-speech tags and close word classes (e.g., pronouns, conjunctions), or clause types

# Open Information Extraction

- problems
  - incoherent extractions – have no meaningful interpretation
  - uninformative extractions: “Faust made a deal with the devil” => (Faust, made, a deal) instead of (Faust, made a deal with, the devil)
  - overly-specific relation phrases: “is offering only modest greenhouse gas reduction targets at”
- solutions – better specification of linguistic constraints

# Question answering

- provides information containing answers to user questions
- related to human language understanding (semantic analysis)
- in early phases limited only to some narrow domains
- can use the IE output (structured databases) for finding answers

# Question answering

- NLP based
  - a query is thoroughly analysed and converted into a formal representation (logic, semantic networks, …)
  - the answer is found using a world model (e.g., an ontology)
  - ensures the most reliable answers
- IR based
  - focus on fact retrieval from a large text corpus
  - employ shallow or deep NLP techniques
  - are considered language and domain independent
- template based
  - question templates are matched against queries
  - after a match is found, an answer in a structured database is found
  - templates are typically created manually

# Data preprocessing and definition of features

# Data preparation – standardization

- converting the document into a suitable format, e.g., extracting text from PDF documents, removing tags from HTML and XML documents,  removing headers from e-mails, etc.
- determining the language of documents
- one document might be further split, e.g., a book into chapters, a newspaper article into paragraphs, a review into sentences

# Data preparation – preprocessing

- algorithms that work with document properties (features) that are related to the document content or other properties are used
- it is therefore necessary to define the features
- features (characters, words, terms, concepts, ) might be extracted or derived
- a document represented as a set of features and their values = structured representation

# Data preparation – preprocessing

- typical steps
  - text cleaning
  - white space removal
  - case folding
  - spelling errors corrections
  - abbreviations expanding
  - stemming
  - lemmatization
  - stop words removal
  - negation handling
  - feature selection
  - part-of-speech tagging
  - syntactical or shallow parsing

# Data preparation – preprocessing

- classification of the typical steps of linguistic processing
  - tokenize – deciding what constitutes a term
  - normalize
    - making same things looking differently look the same
    - usually increases recall, reduces precision
    - e.g., converting superficially different strings of characters to the same form (e.g., *car*, *Car*, *cars*, and *Cars* could all be normalized to *car*) or case folding
  - annotate
    - marking identical strings of characters as being different
    - usually decreases, increases precision
    - e.g., *fly* might be a verb -> *fly/VB* or a noun -> *fly/NN*)

# Data preparation – tokenization

- might seems simple at fist glance (e.g., for English) – words are separated by spaces (unlike, e.g., in Chinese)
- a good tokenizer must  handle punctuation (e.g., don't, Jane's, and/or), hyphenation (e.g., state-of-the-art versus state of the art), and recognize multi-word terms (e.g., *Barack Obama* and *ice hockey*)
- ignoring stop words – high-frequency words with relatively low information content (e.g., of, the, and, them, who, that)
- the text might be broken into character unigrams or bigrams
- the input text might be matched against a against a lexicon
- accurate tokenization is a challenging task for most human languages

# Data preparation – tokenization

- splitting the text into basic units, called tokens (typically words, numbers, currency symbols, dates, …)
- based on understanding the structure of the used language (e.g., what is a delimiter and what is not, how numbers are written, smileys)

# Data preparation – standardization

- converting every token to a standard form, e.g.,

  am, are, is => be

  car, cars, car's, cars' => car
- reduces the number of distinct tokens
- usually increases recall, reduces precision
- case folding
  - easy in English, can be problematic in some languages, e.g., in French, accents are optional for uppercase: example, PECHE -> pêche (fishing or peach) or péché (sin)
  - problems even in English, e.g., SMART (abbreviation of the name of an information retrieval system) vs. smart; Bush (a surname) vs. bush (a kind of plant)

# Data preparation – standardization

- stemming
  - the process of reducing inflected words to their stems
  - in English, affixes are simpler and more regular than in many other languages -> stemming algorithms based on heuristics work relatively well
  - Porter stemming: ED -> ‘’ (plastered -> plaster), ATIONAL -> ATE (relational -> relate), ATOR -> ATE (operator -> operate), EMENT -> ‘’ (replacement -> replac) etc.
- lemmatization
  - requires more detailed morphological analysis to convert a word into so called lemma (dictionary form), like ‘better’ -> ‘good’ (will be missed by a stemmer)

# Data preparation – annotation

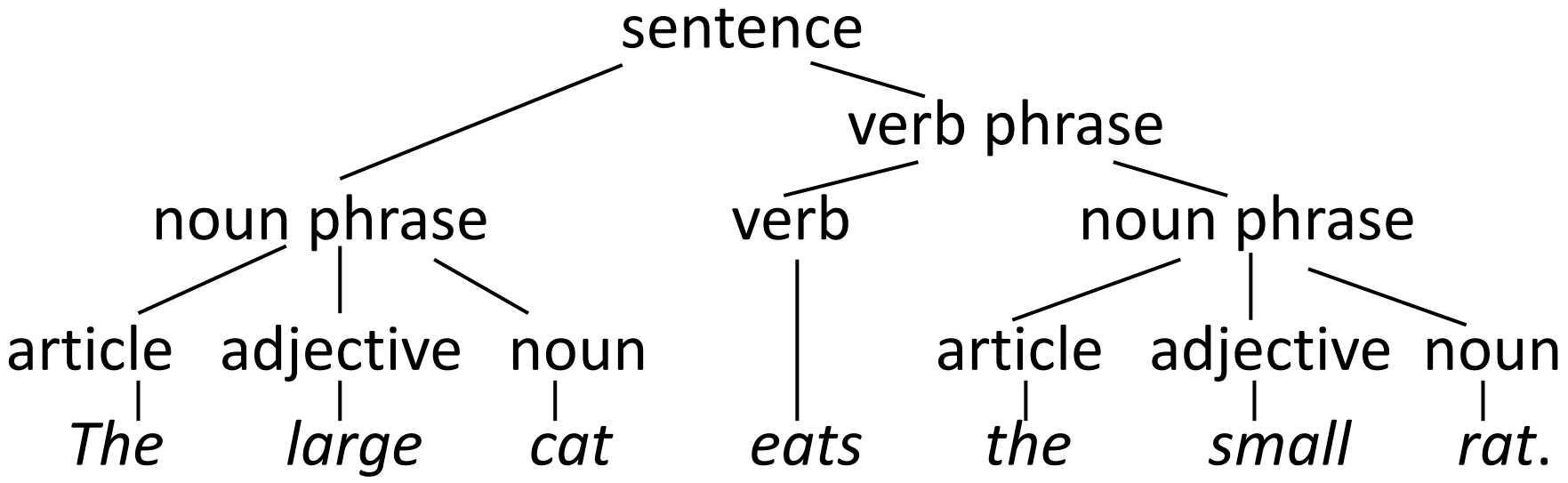
- inverse to normalization
- usually increase precision and decrease recall
- part-of-speech (POS) tagging
  - marking words according to their parts of speech; sometimes, additional linguistic features are needed
  - determining whether a word is a noun, verb, adjective, preposition, conjunction, pronoun, article, adverb, etc.
- word sense disambiguation
  - marking ambiguous words according to their intended meanings
  - some words have more meanings, e.g., *derecho* (in Spanish) is *right* or *law*

# Data preparation – annotation

- named entity recognition
  - some tokens or groups of tokens might represent an entity (a person, company, place, …)
  - e.g., the automotive company created by Henry Ford in 1903 -> Ford, European Union, EU, a politico-economic union of 28 member states that are located primarily in Europe -> European Union

# Data preparation – annotation

- parsing (syntax analysis)
  - analyzing the grammatical structure of sentences and marking the words in the sentences according to their grammatical roles
  - relations of a word to the others and the functions of the words (subject, object, etc.) become obvious



# Data preparation – annotation

- shallow parsing
  - standard parsing algorithms are too expensive for use on very large corpora and are not robust enough
  - shallow parsing produces only parts that are easy and unambiguous, typically, small and simple noun and verb phrases
  - for the purposes of information extraction, shallow parsing is usually sufficient

# Data preparation

- adding potentially relevant attributes if some additional information is available, about, e.g.
  - time
  - place
  - people
  - topic
  - sentiment
  - …

# Converting texts into a suitable format

- a document is typically broken down into some pieces (typically the words) that are not as complex as the entire document is
- these pieces might be than transformed in some way (relevant features are derived):
  - removed (e.g., rare words are eliminated)
  - changed to a different value (e.g., stemmed)
  - supplemented by additional information (e.g., part of speech)
- the features are represented in a way suitable for a particular algorithm (e.g., feature–document matrix)

# Document vector representation

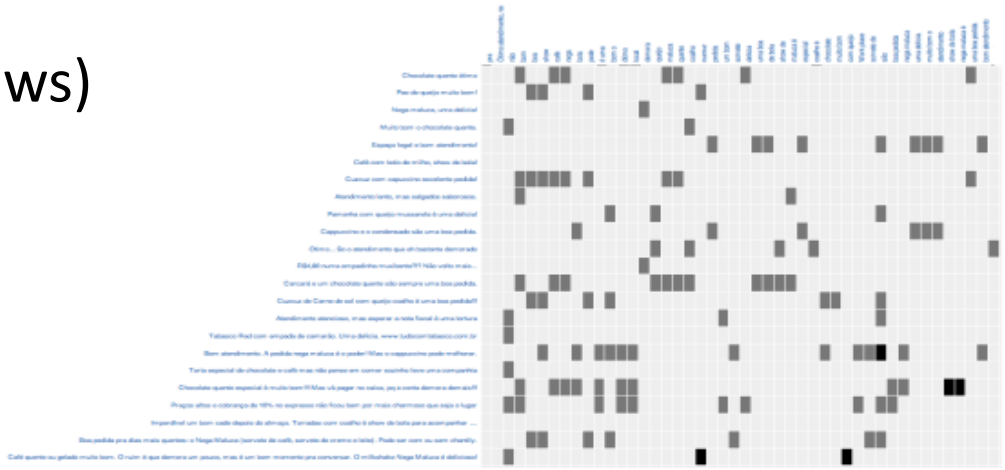
- documents are often treated as a bag of words
- bag = multiset – a set where duplicates are allowed
- not important are
  - the order of words
  - relationships between words
- the problem becomes less complex and thus manageable

# Document vector representation

- a form suitable for data mining algorithms (tabular format)
- a document is represented by a vector
  - attributes – numeric values representing properties of the document
  - properties – terms (included in the text), concepts (derived from the text)
  - attribute values – characterize, quantify the features

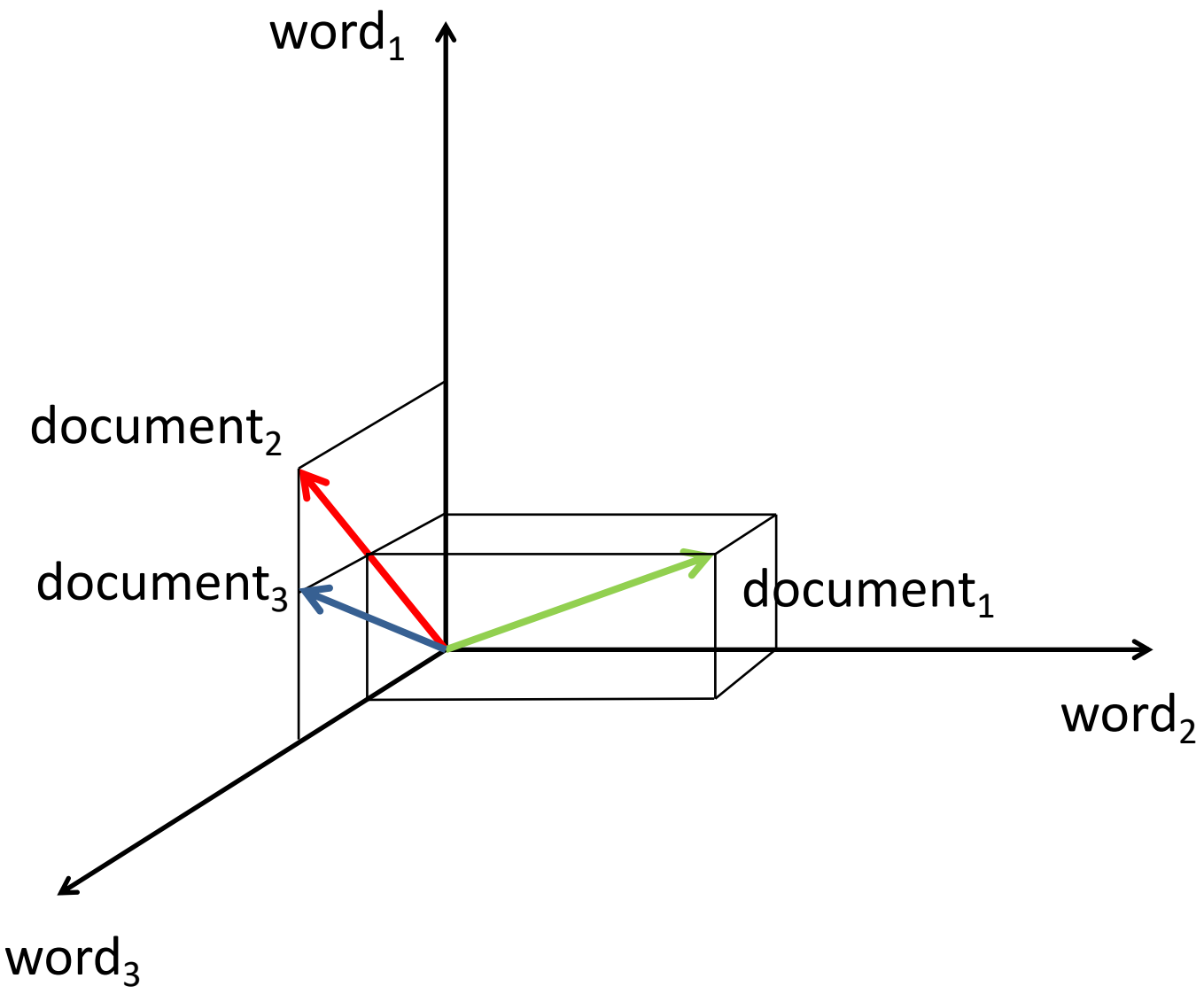
# Document vector representation – a simple example

- a set of documents (reviews)
  - very good product
  - very bad product
  - very very good
  - good
  - quite bad



- a simple document vector representation

| good | quite | product | bad | very |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 2 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |


word1
document2
document3
document1
word2
word3

# Vector components

- a *local weight* $L_{ij}$ representing the frequency $f_{ij}$ of term i in document j (in every single document)
- a *global weight* $G_i$ reflecting the discriminative ability of the term i, based on the distribution of the term over the entire document collection
- a *normalization factor* $N_j$ correcting the impact of different lengths of documents

$$w_{ij} = L_{ij} * G_i / N_j$$

$w_{ij}$ … the weight of i-th term in j-th document

i … the number of a term

j … the number of a document

# Local weights

- term presence (binary)

  $f_{ij} = 0 \Rightarrow w_{ij} = 0,\ f_{ij} > 0 \Rightarrow w_{ij} = 1$

- term frequency

  $w_{ij} = f_{ij}$

- logarithm
  - de-emphasize the effect of frequency (a term appearing 10 times is not 10 times more important than a term appearing once)

    $f_{ij} = 0 \Rightarrow w_{ij} = 0,\ f_{ij} > 0 \Rightarrow w_{ij} = 1 + \log f_{ij}$

# Local weights

- augmented normalized term frequency
  - assigns weight k (typically 0.5) to every term in a document + a bonus up to 1-k
  - for longer documents lower (e.g., 0.3), for shorter documents higher (e.g., 0.5)

    $f_{ij} = 0 \Rightarrow w_{ij} = 0,\ f_{ij} > 0 \Rightarrow w_{ij} = k + (1-k)*f_{ij}/x_j$

    ($x_j$ is the maximum frequency in document j)
- and many others

# Global weights

- inverse document frequency (the most popular)

$$idf(t_i) = \log \frac{N}{n(t_i)}$$

  N … number of documents, $n(t_i)$ … number of documents containing term $t_i$ (document frequency)

- other variations, e.g., squared inverse document frequency, probabilistic inverse document frequency, GFIDF, Entropy, …

# Normalization

- transforms the elements of vector $d_j$, i.e., the vector with local and global weighting applied:

$$d_j = (l_{1j}*g_1, l_{2j}*g_2, \ldots, l_{mj}*g_m)$$

- eliminates the problem of preferring longer documents because
  - they have higher term frequencies
  - contain more words (the probability of matching against a query is higher)

# Normalization

- cosine normalization
  - most popular

$$n_j = \sqrt{\sum_{i=1}^{m} {d_i}^2}$$

- other variants – max weight, sum of weights…

# Document classification

# Classification (categorization)

- many text mining tasks require that the data items to be processed have assigned labels => importance of classification
- a label
  - symbolic, no additional knowledge is available
  - characterizes a class (category)
- applications: document (web page, newspaper article, scientific paper, …) categorization, spam detection, word sense disambiguation, authorship attribution, language identification, or sentiment analysis

# Classification problem

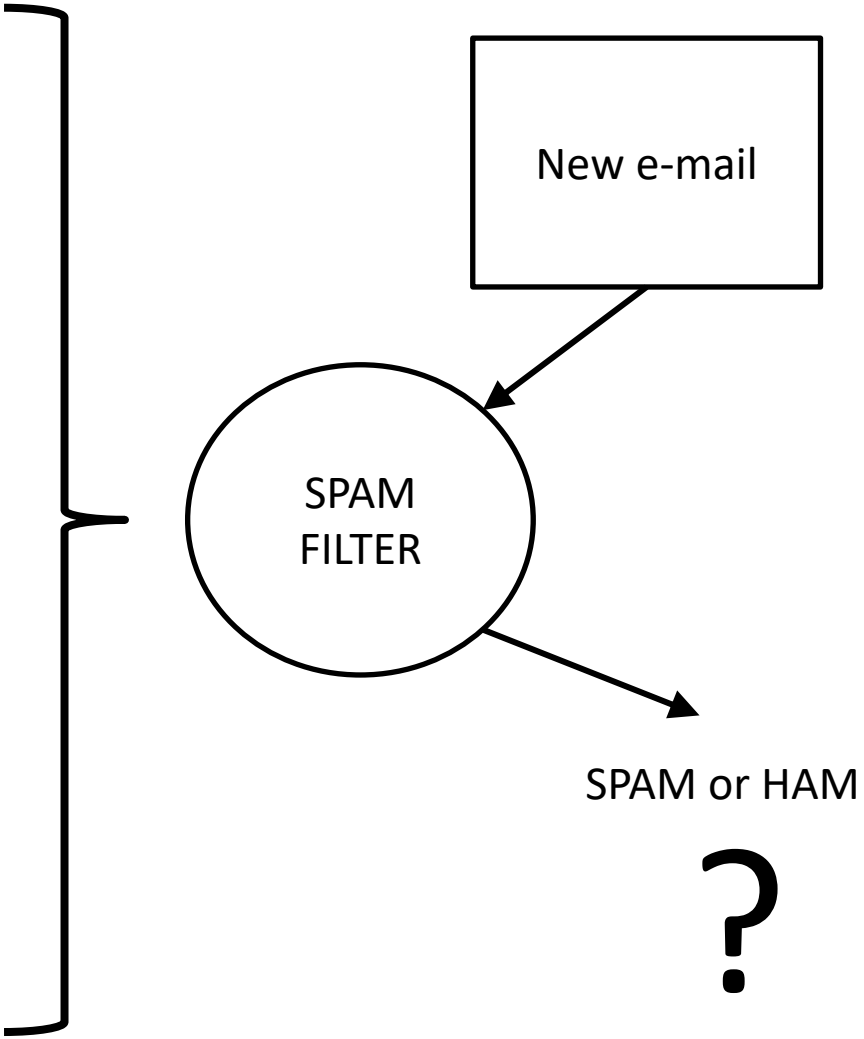
- to find a function (classifier, hypothesis) that can be, according to training data, represented by input data (training examples) paired with desired outputs (class labels), used for predicting the outputs for new, unseen data

$(I_1, O_1)$
$(I_2, O_2)$ training generalization
$(I_3, O_3)$ → $f(I) \rightarrow O$
…
$(I_n, O_n)$

# Classification problem

- having documents $d_1$, $d_2$, …, $d_n$ and classes $c_1$, $c_2$, …, $c_m$, the hypothesis assigns true or false to each pair ($d_j$, $c_i$) denoting whether document $d_j$ belongs to category $c_i$
- alternatively, the categories might be ranked according to their appropriateness and a human expert might decide on a category (categories)

| | |
|---|---|
| Would you like to loose weight? | SPAM |
| Cheap Viagra!!! | SPAM |
| Amateur teen girls | SPAM |
| Urgent money transfer | SPAM |
| ... | |
| Department meeting takes place on Tuesday | HAM |
| Registration of courses has started | HAM |
| New grant possibilities have been announced | HAM |
| Dissertation thesis defense | HAM |
| ... | |



# Classification problem

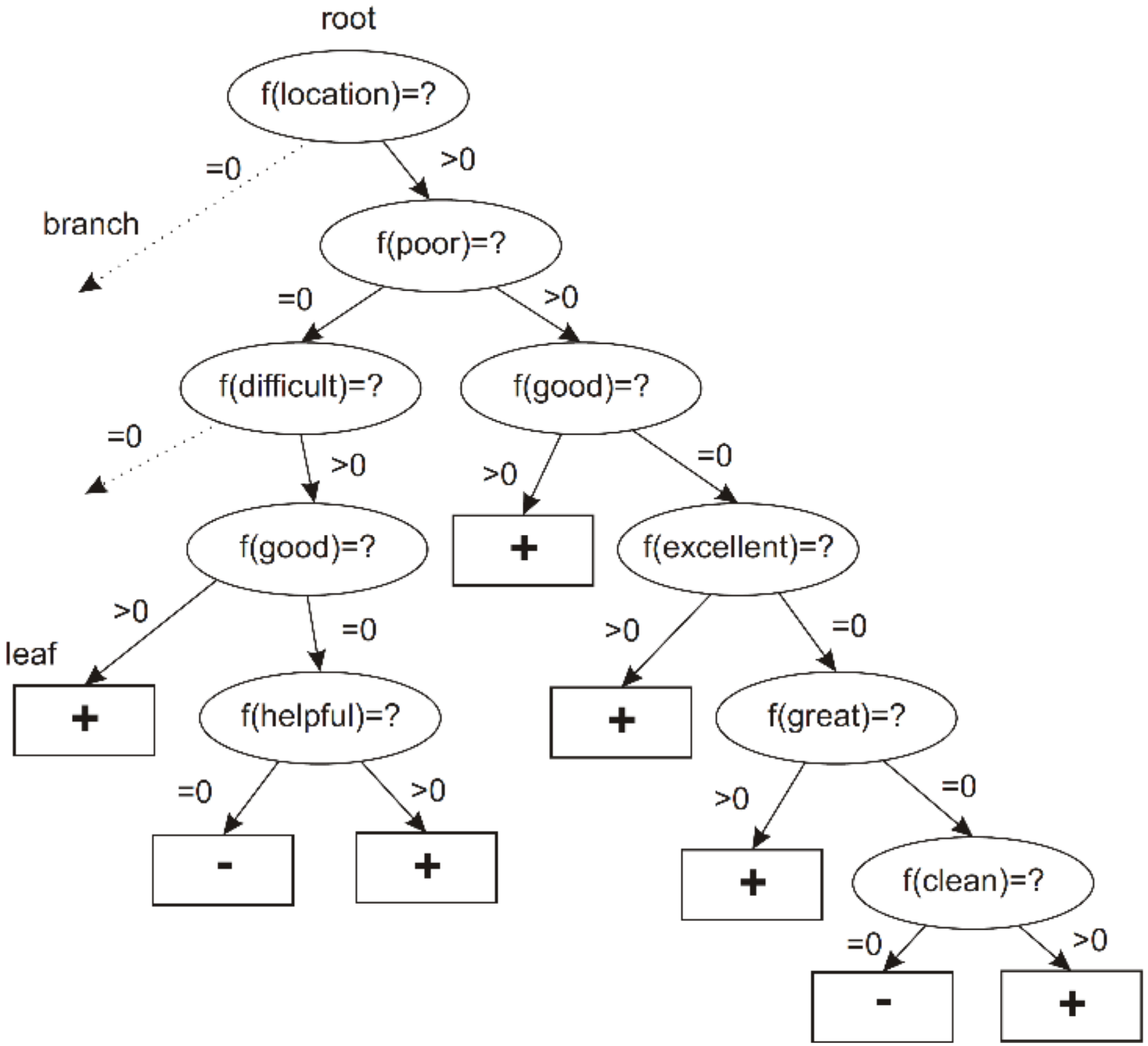
- single-label classification – only one category is assigned
- binary classification – a document belongs to $c_i$ or to the complement of $c_i$
- multi-label classification
  - a document can belong to 0..|C| categories
  - can be transformed to |C| independent binary classifications

# Commonly used classifiers

- probabilistic classifiers, decision trees, decision rules, example-based classifiers, support vectors machine, or neural networks
- it is very difficult to compare individual methods
  - different authors use different sampling, preprocessing, algorithm setting, etc.
  - generally, support vectors machine, instance based classifiers, neural networks, and decision trees bring acceptable results

# Decision trees

- the category of an object is determined by performing a sequence of tests, based on the values of attributes characterizing the object
- a DT is represented by a directed graph (with one root) where the nodes represent the questions and the leaves the classes
- able to explain *why* a certain instance should be assigned to a specific class


root
f(location)=?
=0
>0
branch
f(poor)=?
=0
>0
f(difficult)=?
f(good)=?
=0
>0
>0
=0
f(good)=?
+
f(excellent)=?
>0
=0
>0
=0
leaf
+
f(helpful)=?
+
f(great)=?
=0
>0
=0
>0
-
+
+
f(clean)=?
=0
>0
-
+

# Decision tree training

- a DT is usually build using the top-down approach:
  1. find an attribute $a_i$ that best divides T (training set)
  2. divide T into subsets $T_i$ , each characterized by a different value of $a_i$
  3. if all documents in $T_i$ belong to the same class, create a leaf labelled with that class; otherwise, recursively apply the same procedure starting with step 1 to $T_i$

# Decision tree training

- calculating the suitability of an attribute to split the training set
- a common measure is the information gain

$$IG(T, a_i) = H(T) - H(T, a_i)$$

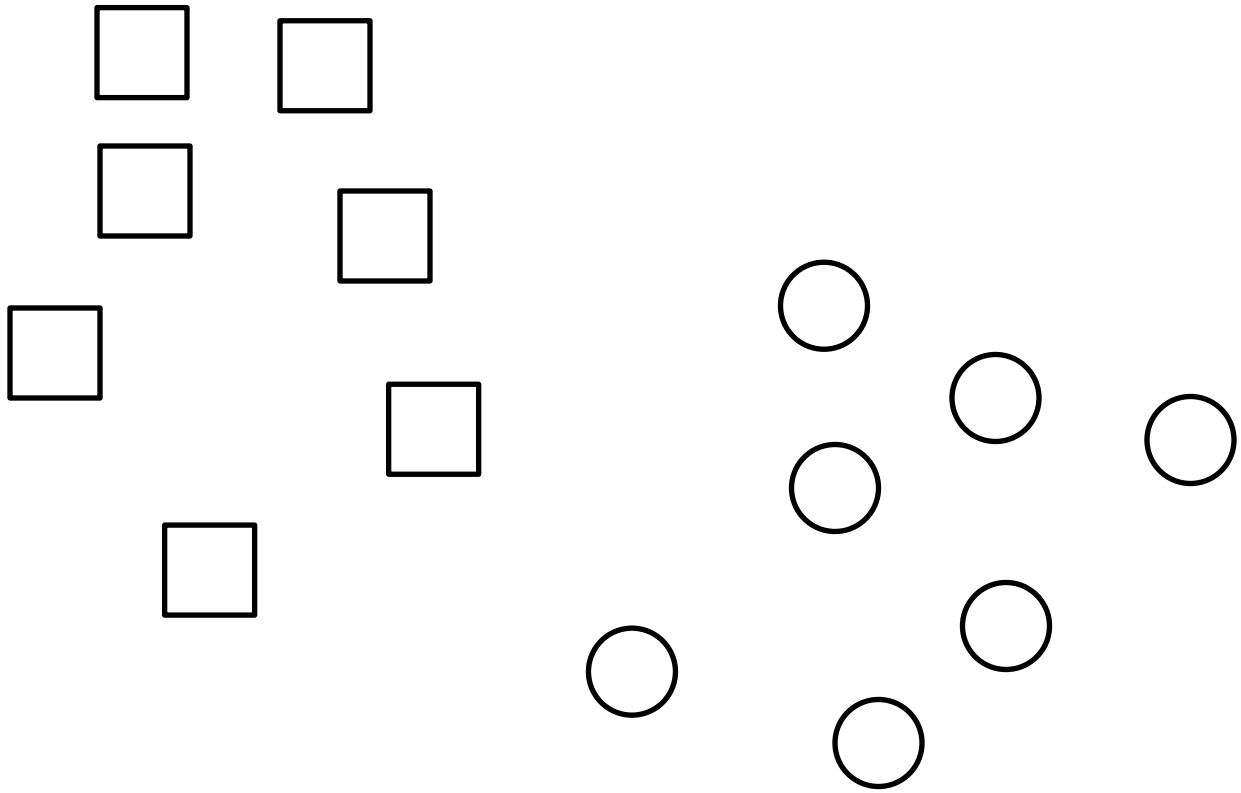
$$H(T) = -\sum_{j=1}^{n} p(c_i) \log p(c_i)$$

$$H(T, a_i) = \sum_{i} P_i H(T_i)$$

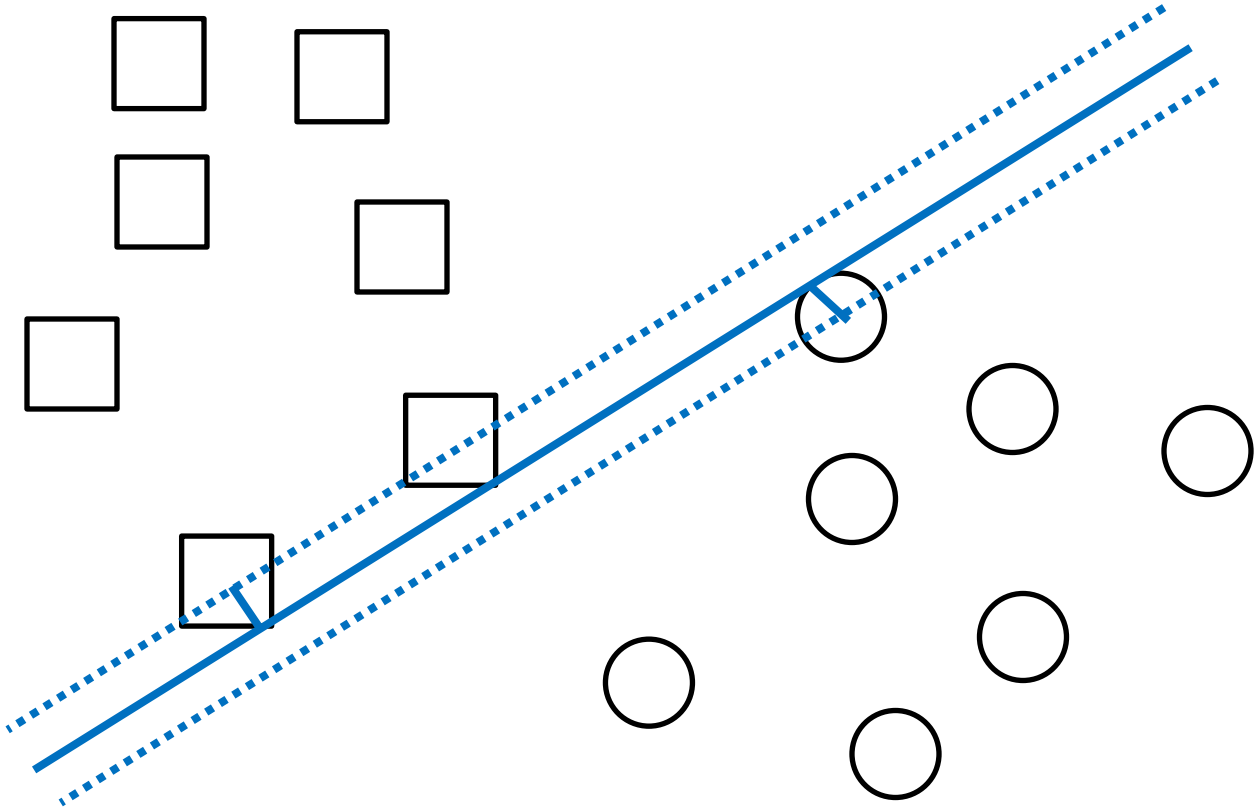
H … entropy, $P_i$ … relative size of $T_i$, n … number of classes

# Support vector machine

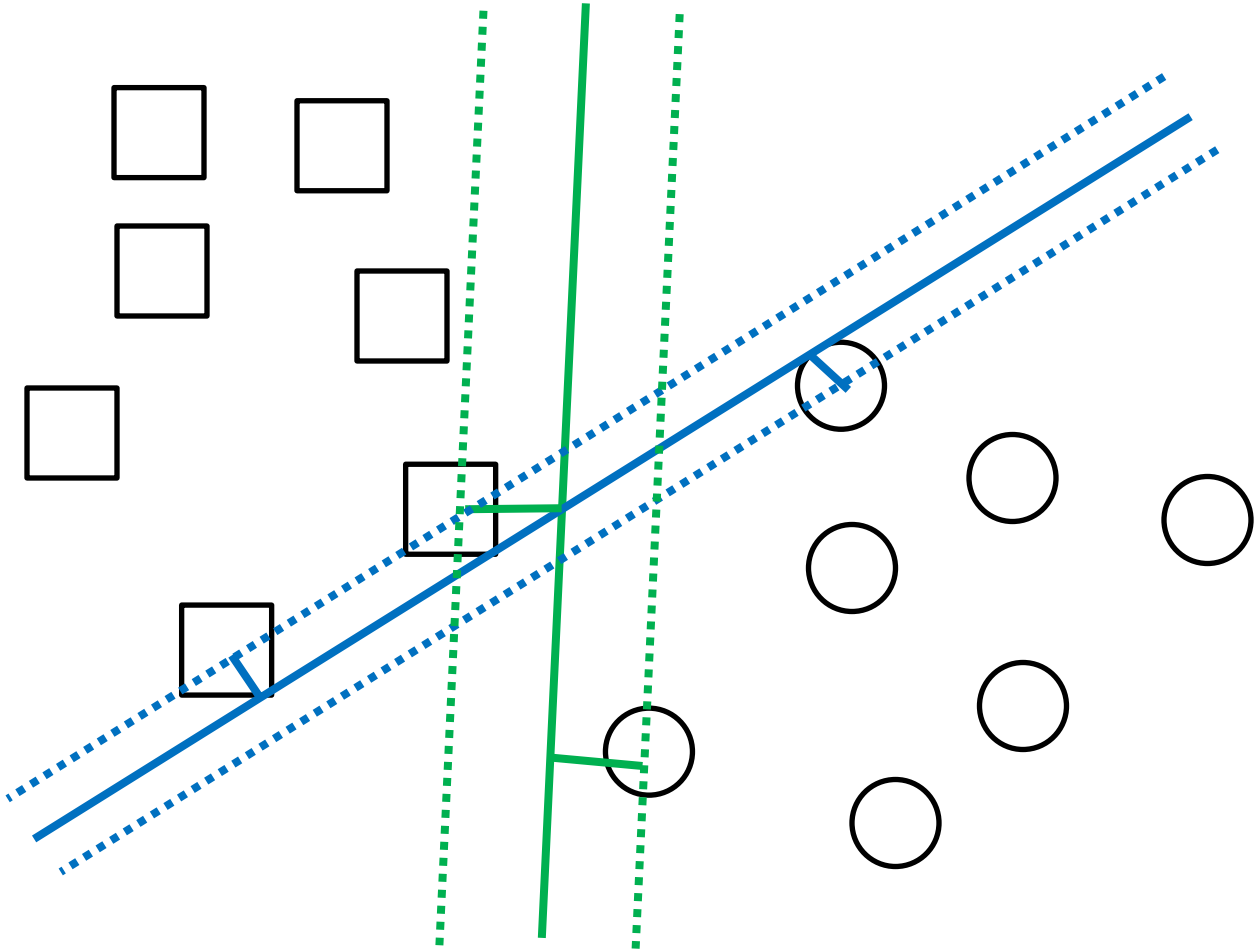
- the data is partitioned by finding a linear boundary (hyperplane) between two classes
- the margin widths between the class boundary and training patterns are maximized during the training process
- the best decision surface is determined by only a small set of training examples, called the support vectors
- robust to overfitting



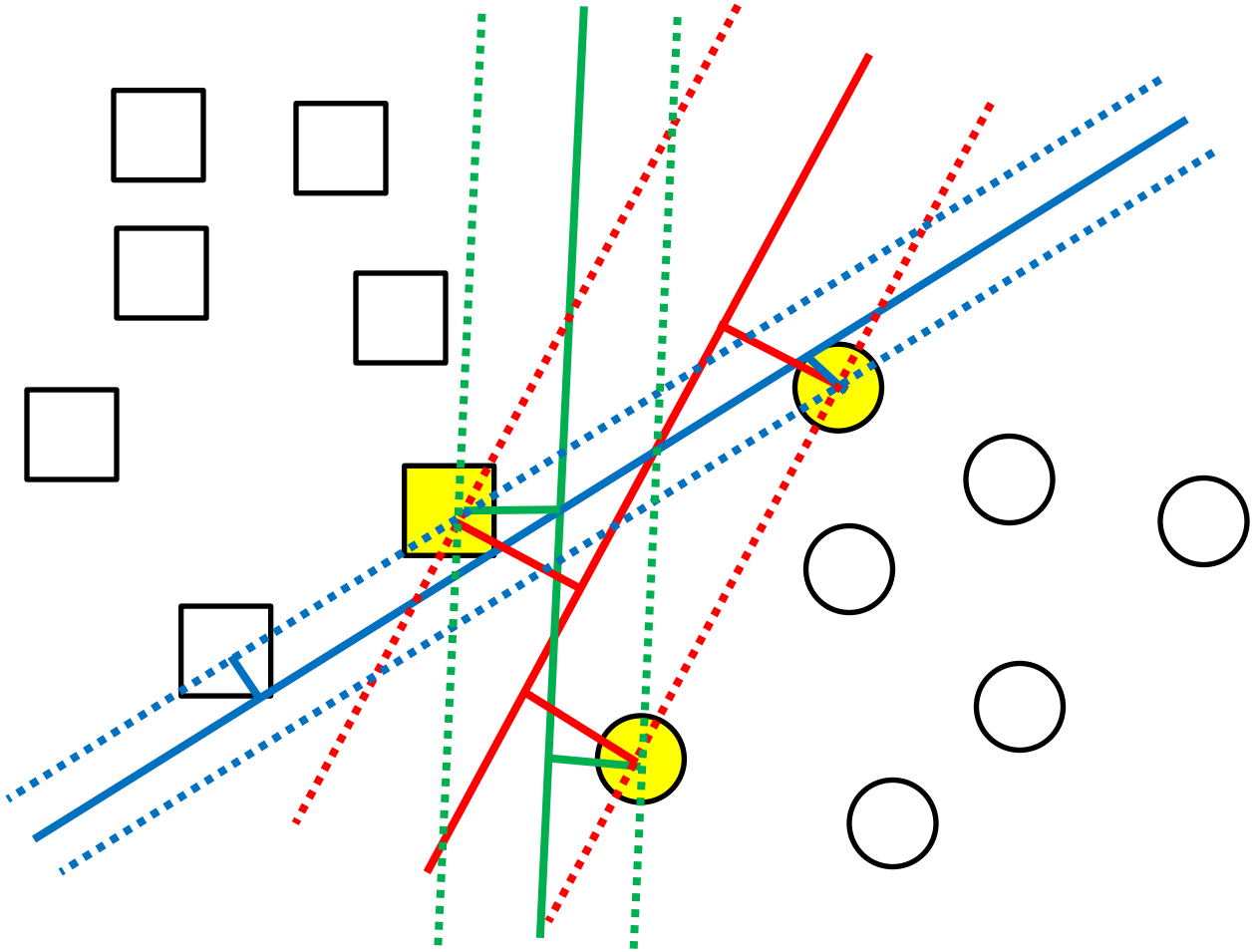
objects of two classes to be separated



the blue line separates the data with some margin



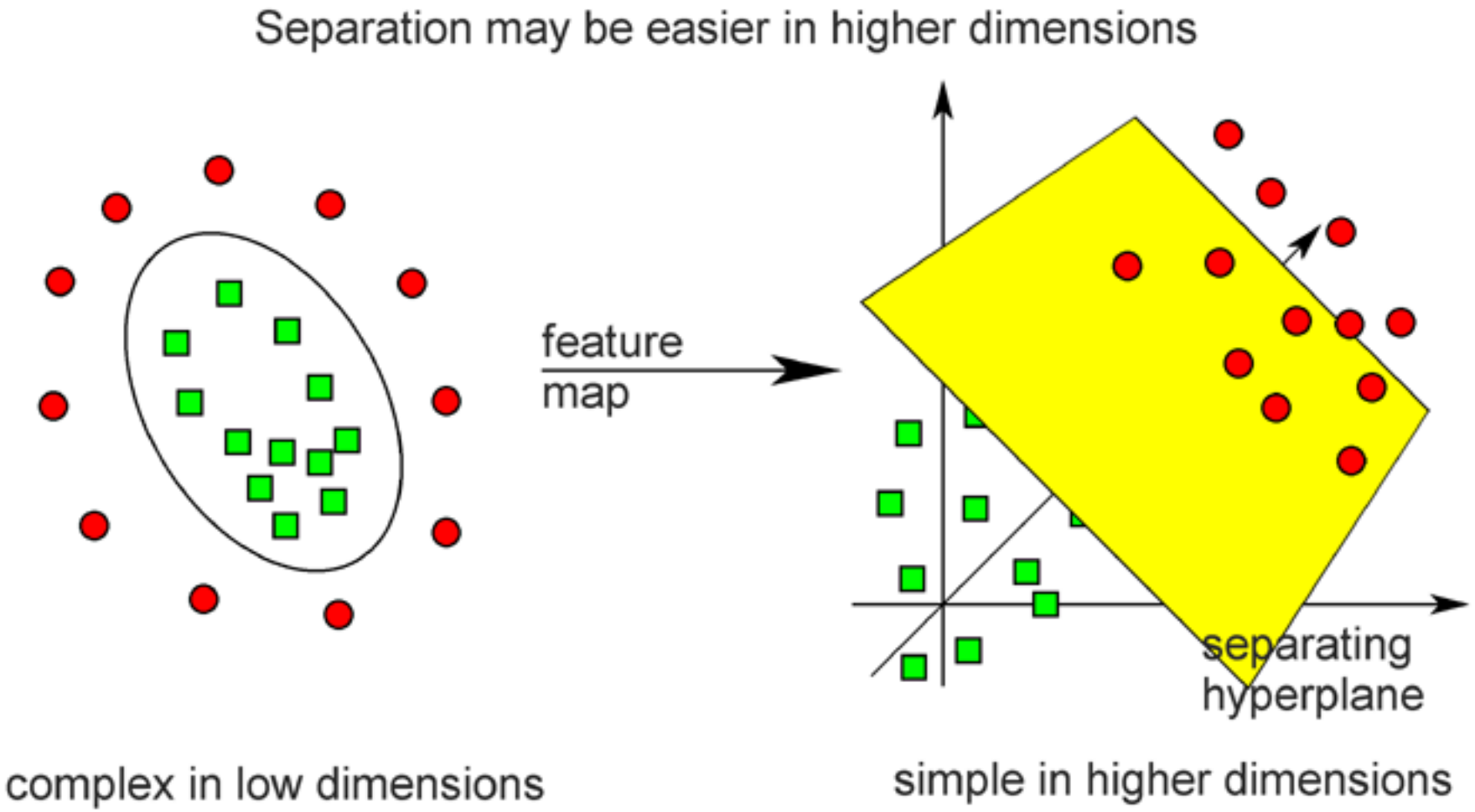
the green line separates the data with a larger margin



the red line separates the data with the largest margin
(dotted lines contain the support vectors – yellow objects)

# Support vector machine

- when it is not possible to separate the data in a given n-dimensional space linearly, a kernel function that projects the data to a space of higher dimension is used

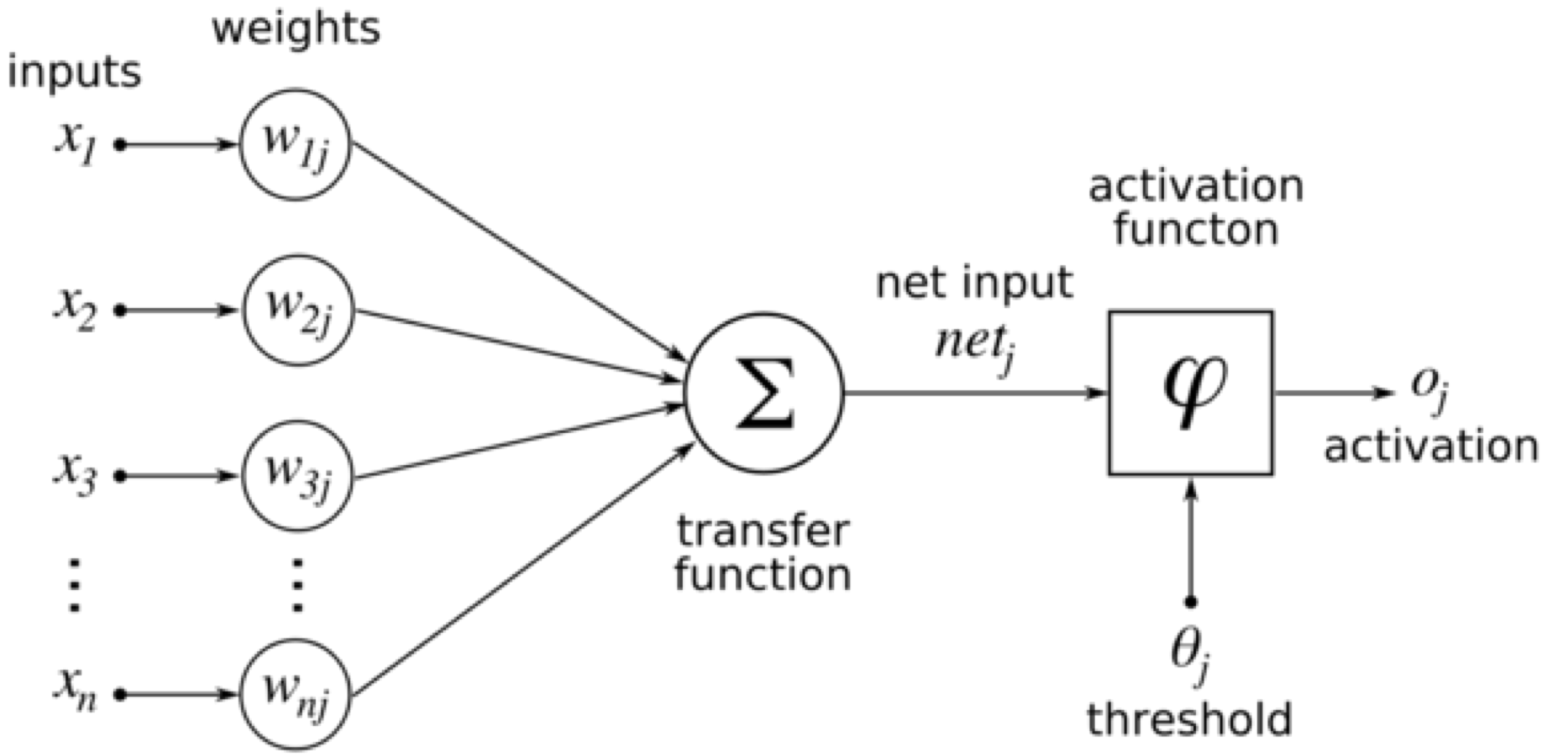



Van Looy *et al. Critical Care* 2007 **11**:R83

# Artificial neural networks

- imitate the behavior of neural networks in living organisms
- consist of elements called neurons
- neurons receive signals (all inputs are combined into a single value ) and produce responses (an activation function)
- the neurons are organized in a network (layers) with their inputs and outputs connected together
- the interconnected neurons compose a function of the whole network

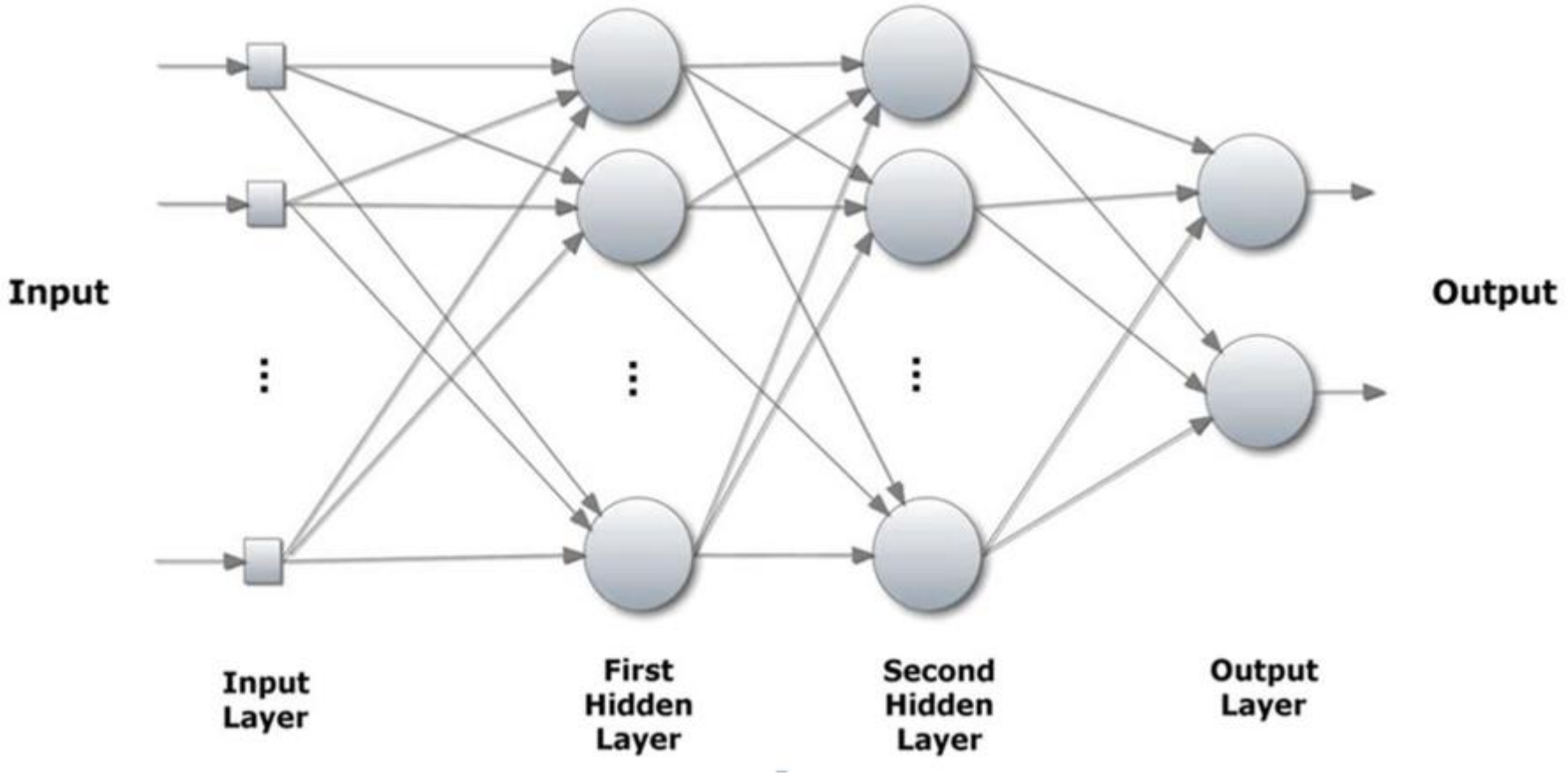
# Artificial neural networks – a neuron



# Artificial neural networks design

- the structure of neurons
- the topology of the network
- the learning algorithm (the backpropagation algorithm is one of the most used)

# Artificial neural networks design



# Artificial neural networks learning

- ANN inputs – features
- ANN outputs – classes
- learning process – updating the inter-neuronal synaptic weights during training iterations in order to maximize the correctness of assigned classes

# Instance-based classifiers

- sometimes called lazy learners or exemplar-based classifiers
- do not create a generalization of class representation, a label of an unknown document is determined according to the training documents that are similar
- fast training, slow classification

# Instance-based classifiers

- *k*-nearest neighbors classifier (IBL-1) is a typical representative
  - a decision whether a document $d_i$ belongs to class $c_j$ is made based on *k* documents that are most similar to $d_i$
  - if a large enough proportion of them have the label $c_j$, a positive decision is taken
  - *k* nearest documents can have a different weight according to the degree of their similarity to $d_i$

# Instance-based classifiers

- IBL-2
  - it is not necessary to store every instance the learner has seen to classify unseen instances
  - only instances near the boundaries in a small neighborhood of the boundary line, are needed to produce an accurate approximation of the concept boundary (instances away from the concept boundary do not really matter in classification)
  - we can save space by storing only informative instances

# Instance-based classifiers

- IBL-3
  - IB3 is a noise tolerant algorithm that reduces IB1's storage requirements
  - a "wait and see" method is employed – a classification record with each saved instance is maintained
  - only good classifiers are used to classify subsequently presented instances

# Probabilistic classifiers

- calculate the probability $P(c_j \mid d_i)$ that a document $d_i$ belongs to class $c_j$
- this can be computed using the Bayes' theorem

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{P(d_i)}$$

# Naïve Bayes Classifier

- finding the most likely class (maximum a posteriori)

$$C_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} \frac{P(c_j)P(d_i|c_j)}{P(d_i)} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j)P(d_i|c_j)$$

- $d_i = (x_1, x_2, \dots, x_m)$ so the probability $\mathrm{P}(x_1, x_2, \dots, x_m \,|c_j)$ must be calculated (possible only when a very large number of training examples is available)
- if the features are independent

$$\mathrm{P}(x_1, x_2, \dots, x_m \,|c_j) = \mathrm{P}(x_1|c_j) \cdot \mathrm{P}(x_2|c_j) \cdot \dots \cdot \mathrm{P}(x_m|c_j)$$

# Naïve Bayes Classifier

- calculating the necessary probabilities – from frequencies in the data

$$P(c_j) = \frac{number\ of\ documents\ from\ class\ c_j}{number\ of\ documents}$$

$$\mathrm{P}(x_i|c_j) = \frac{count(x_i, c_j)}{\sum_{x \in V} count(x, c_j)}$$

# Naïve Bayes Classifier

- if one of $\mathrm{P}(x_i|c_j)$ is 0 the entire formula will result in zero, no matter the other evidence
- thus, Laplace smoothing is used

$$\mathrm{P}(x_i|c_j) = \frac{count(x_i, c_j) + 1}{\sum_{x \in V}(count(x, c_j) + 1)}$$

$$= \frac{count(x_i, c_j) + 1}{(\sum_{x \in V} count(x, c_j)) + M}$$

M … size of the vocabulary

# Naïve Bayes Classifier – example

| | |
|---|---|
| good | + |
| very good | + |
| very very good | + |
| not good | - |
| very bad | - |
| bad | - |
| not very good | ? |

we need to calculate P(?|not very good), where ? id + and -

we need P(+), P(-), P(not very good | + ), and P(not very good | - )

# Naïve Bayes Classifier – example

| | |
|---|---|
| good | + |
| very good | + |
| very very good | + |
| not good | - |
| very bad | - |
| bad | - |
| not very good | ? |

| |
|---|
| P(+) = 3/6 |
| P(-) = 3/6 |
| P(good\|+) = (3+1)/(5+4) |
| P(very\|+) = (2+1)/(5+4) |
| P(bad\|+) = (0+1)/(5+4) |
| P(not\|+) = (0+1)/(5+4) |
| P(good\|-) = (1+1)/(5+4) |
| P(very\|-) = (1+1)/(5+4) |
| P(bad\|-) = (2+1)/(5+4) |
| P(not\|-) = (1+1)/(5+4) |

# Naïve Bayes Classifier – example

| | |
|---|---|
| good | + |
| very good | + |
| very very good | + |
| not good | - |
| very bad | - |
| bad | - |
| not very good | ? |

| |
|---|
| P(+) = 3/6 |
| P(-) = 3/6 |
| P(good\|+) = (3+1)/(5+4) |
| P(very\|+) = (2+1)/(5+4) |
| P(bad\|+) = (0+1)/(5+4) |
| P(not\|+) = (0+1)/(5+4) |
| P(good\|-) = (1+1)/(5+4) |
| P(very\|-) = (1+1)/(5+4) |
| P(bad\|-) = (2+1)/(5+4) |
| P(not\|-) = (1+1)/(5+4) |

P(+|not very good) = 1/2*1/9*3/9*4/9 = 0.00823
P(-|not very good) = 1/2*2/9*2/9*2/9 = 0.005487

# Naïve Bayes Classifier – example

| | |
|---|---|
| good | + |
| very good | + |
| very very good | + |
| not good | - |
| very bad | - |
| bad | - |
| not very good | ? |

| |
|---|
| P(+) = 3/6 |
| P(-) = 3/6 |
| P(good\|+) = (3+1)/(6+4) |
| P(very\|+) = (3+1)/(6+4) |
| P(bad\|+) = (0+1)/(6+4) |
| P(not\|+) = (0+1)/(6+4) |
| P(good\|-) = (1+1)/(5+4) |
| P(very\|-) = (1+1)/(5+4) |
| P(bad\|-) = (2+1)/(5+4) |
| P(not\|-) = (1+1)/(5+4) |

P(+|not very good) = 1/2*1/9*3/9*4/9 = 0.00823
P(-|not very good) = 1/2*2/9*2/9*2/9 = 0.005487

# Measuring the quality of classifiers

- classifiers trained on training data are applied to test samples (objects with known labels)
- in the two class classification, the classes might be labeled as positive and negative; correctly classified objects are referred as true positive (TP) and true negative (TN); false positive (FP) and (FN) represent misclassified objects

# Classifier effectiveness measures

- standard IR measures, like precision and recall might be used
- precision: $P( f(c_i, d_x) = true \mid h(c_i, d_x) = true )$
  - the probability that if a random document $d_x$ is classified under $c_i$, the decision is correct
- recall: $P( h(c_i, d_x) = true \mid f(c_i, d_x) = true )$
  - the probability that if a random document $d_x$ should be classified under $c_i$, this decision is taken

# Classifier effectiveness measures

- the values of precision and recall can be estimated from the contingency table (P…predicted, R…real)

| C1 | P+ | P- |
|---|---|---|
| R+ | 50 | 30 |
| R- | 15 | 110 |

| C2 | P+ | P- |
|---|---|---|
| R+ | 60 | 20 |
| R- | 20 | 105 |

| C3 | P+ | P- |
|---|---|---|
| R+ | 40 | 5 |
| R- | 20 | 140 |

Prec = 50/(50+15)
Rec = 50/(50+30)

Prec = 60/(60+20)
Rec = 60/(60+20)

Prec = 40/(40+20)
Rec = 40/(40+5)

| | P C1 | P C2 | P C3 |
|---|---|---|---|
| R C1 | 50 | 20 | 10 |
| R C2 | 10 | 60 | 10 |
| R C3 | 5 | 0 | 40 |

Prec(C1) = 50 / (50+10+5)
Rec(C1) = 50/(50+20+10)
Prec(C2) = 60 / (20+60+0)
Rec(C2) = 60/(10+60+10)
Prec(C3) = 40 / (10+10+40)
Rec(C3) = 40/(5+0+40)

# Measuring the quality of classifiers

- Accuracy
  - proportion of good results among all
  - not suitable for unbalanced data

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision (positive predictive value)
  - proportion of good results among result marked as good (retrieved results)

$$Precision = \frac{TP}{TP + FP}$$

# Measuring the quality of classifiers

- Recall (sensitivity, true positive rate)
  - proportion of retrieved good results among all good results

$$Recall = \frac{TP}{TP + FN}$$

- F-measure (F-score)
  - balances Precision and Recall
  - values between 0 (worst) and 1 (best)

$$F\text{-}measure = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall}$$

  - often β=1, $F_1$score is the harmonic mean of Precision and Recall (they are equally balanced)

# Micro- and macro- averaging

- macroaveraging
  - computes a simple average over classes
  - each class has equal weight
- microaveraging
  - aggregates per-document decisions across classes (computes a measure on the pooled contingency table)
  - classes have different weights depending on their size (bigger classes contribute to a measure value more)

# Micro- and macro- averaging

| **Class 1** | | |
|---|---|---|
| | R: yes | R: no |
| P: yes | 10 | 10 |
| P: no | 10 | 970 |

| **Class 2** | | |
|---|---|---|
| | R: yes | R: no |
| P: yes | 90 | 10 |
| P: no | 10 | 890 |

| **Overall** | | |
|---|---|---|
| | R: yes | R: no |
| P: yes | 100 | 20 |
| P: no | 20 | 1860 |

R … real value, P … predicted value

- Prec(1) = 10/(10+10) =0.5
- Prec(2) = 90/(90+10) =0.9
- microaveraged: Prec(overall) = 100/(100+20) = 0.83
- macroaveraged: (Prec(1)+Prec(2))/2 = 0.7
- microaveraged Precision is closer to 0.9 than to 0.5 because Class 2 is five times bigger than Class 1

# Documents clustering

# Documents clustering

- an answer to unavailability of labeled data – no characteristics of the groups (classes) is known
- goal = grouping data into subsets called clusters using information contained *in the data*
- the clusters are *coherent* (homogeneous) internally and must be clearly *different* from each other
- no labels = no one correct solution
- used in automatic creation of ontologies, summarizing, disambiguating, and navigating the results retrieved by a search engine, patent analysis, detecting crime patterns, and many others

# Similarity

- the only endogenous information that is available in the clustering process, used also by instance based classifiers (k-NN)
- can be measured between individual documents, between groups of documents, or between a document and a group of documents
- a vector representing a group of documents

$$C_C = \frac{\sum_{x \in C} x}{|C|} = (C_{C1}, C_{C2}, \ldots, C_{Cm})$$

# Similarity measures

- cosine similarity (based on the angle between two vectors)

$$L_C(A,B) = \frac{A \cdot B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{m} A_i * B_i}{\sqrt{\sum_{i=1}^{m} (A_i)^2} * \sqrt{\sum_{i=1}^{m} (B_i)^2}}$$

- Euclidean distance (distance in the vector space)

$$L_E(A,B) = \sqrt{\sum_{i=1}^{m} (A_i - B_i)^2}$$

- Jaccard coefficient (intersection of two sets divided by their union)

$$L_J(A,B) = \frac{A \cdot B}{\|A\| + \|B\| - A \cdot B}$$

$$= \frac{\sum_{i=1}^{m} A_i * B_i}{\sqrt{\sum_{i=1}^{m} (A_i)^2} + \sqrt{\sum_{i=1}^{m} (B_i)^2} - \sum_{i=1}^{m} A_i * B_i}$$

# Clustering algorithms

- *hard* clustering – each object belongs exactly to one cluster
- *soft* or *fuzzy* clustering – each object is associated with each cluster with a degree of membership in the interval [0, 1]
- different clustering algorithms usually lead to different clustering solutions
- clustering optimization is a very hard task (e.g., for a hard, flat clustering of $n$ elements into $k$ clusters there exist $k^n/k!$ possible solutions)

# Clustering algorithms

- *partitional (flat) clustering* seeks a k-partition $C = \{C_1, C_2, \ldots, C_k\}$, $k \leq n$ of $X$

  such that

$$C_i \neq \emptyset, i = 1, 2, \ldots, k;$$

$$\bigcup_{i=1}^{k} C_i = X;\ C_i \cap C_j = \emptyset, i, j = 1, 2, \ldots, k \text{ and } i \neq j$$

# Clustering algorithms

- *hierarchical clustering* constructs a tree like, nested structure partition of X,

  $$H = \{H_1, H_2, \ldots, H_k\},\ k < n,$$

  such that $C_i \in H_m$, $C_j \in H_l$, and $m > l$ imply $C_i \subset C_j$ or $C_i \cap C_j = \varnothing$ for all $i, j \neq i, m, l = 1, 2, \ldots, k$

# Clustering algorithms

- *graph clustering* – partitioning a *similarity graph* (vertices = documents, edges = quantified according to the similarity between the documents), i.e., minimization of the edge-cut; an alternate model considers the documents and the terms contained in them to be the vertices and weights of the edges are set using *tf-idf* measure

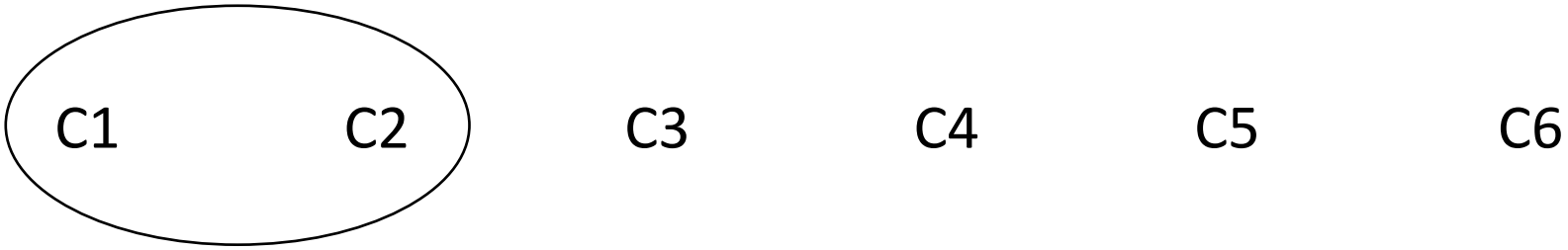
# Hierarchical clustering

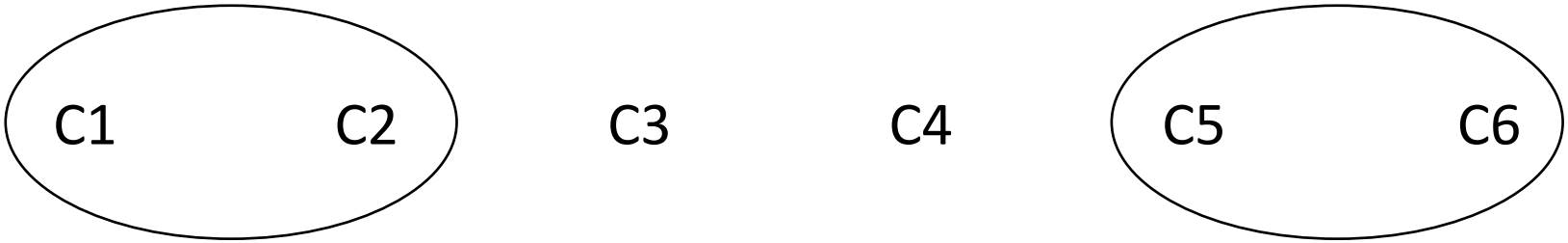
- top-down (divisive) – at the beginning is all data in one cluster; in the following steps, the data is split into several clusters (typically using a flat method); the same step is then recursively repeated for the new clusters according to some criteria
- bottom-up (agglomerative) methods – each item to be clustered is considered to be a single cluster at the beginning; in each of the following steps, two most similar clusters are joined together until all objects are in one cluster

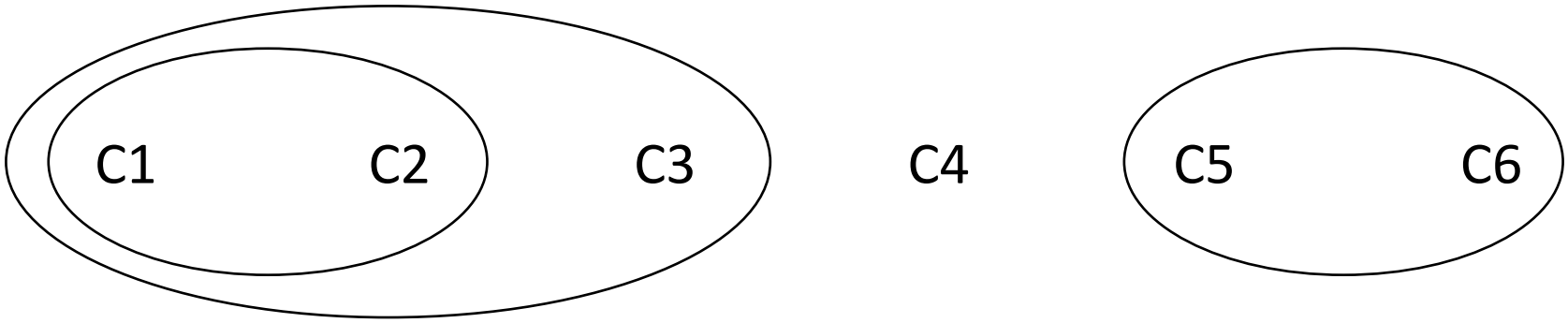
# Merging clusters in agglomerative methods

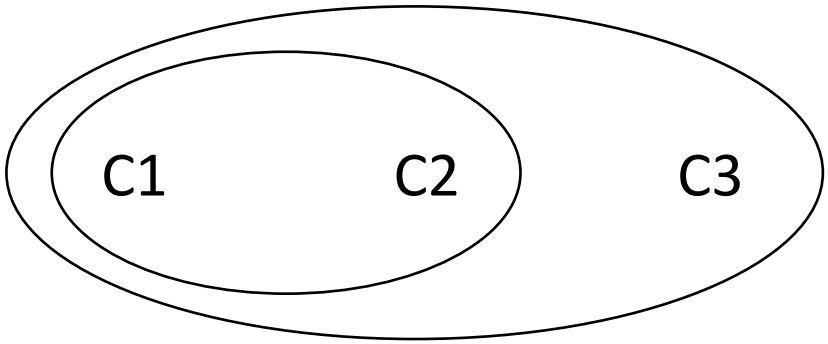
- single linkage clustering method (SLINK), the clusters are merged according to the similarity of the most similar objects from the clusters
- complete linkage clustering method (CLINK), determines the similarity according to the similarity of two most dissimilar objects
- Unweighted Pair Group Method with Arithmetic Mean (UPGMA), the distance between two clusters is calculated as the average of all distances between pairs of objects
- Weighted Pair Group Method with Arithmetic Mean (WPGMA) assigns higher importance to the objects added to the clusters later

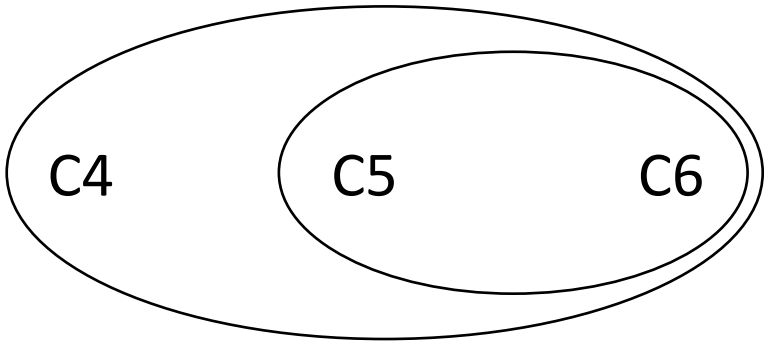
C1 C2 C3 C4 C5 C6

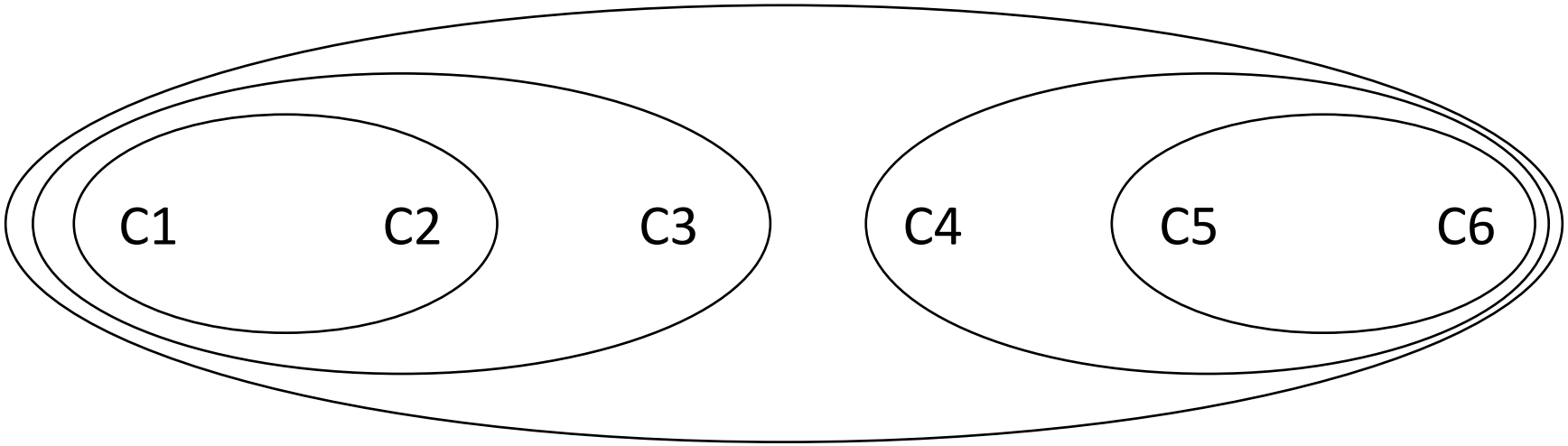

C1
C2
C3
C4
C5
C6

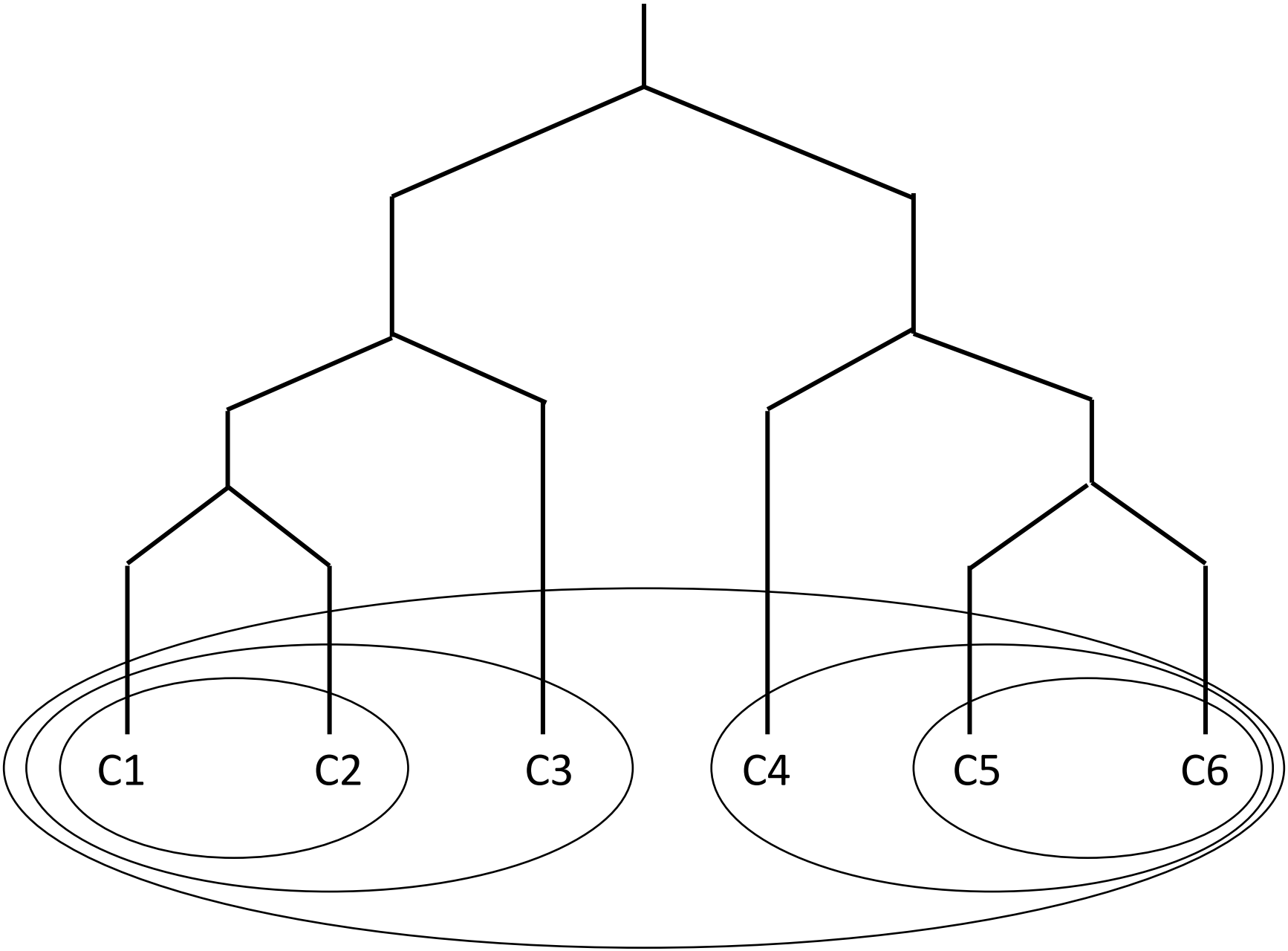

C1
C2
C3
C4
C5
C6

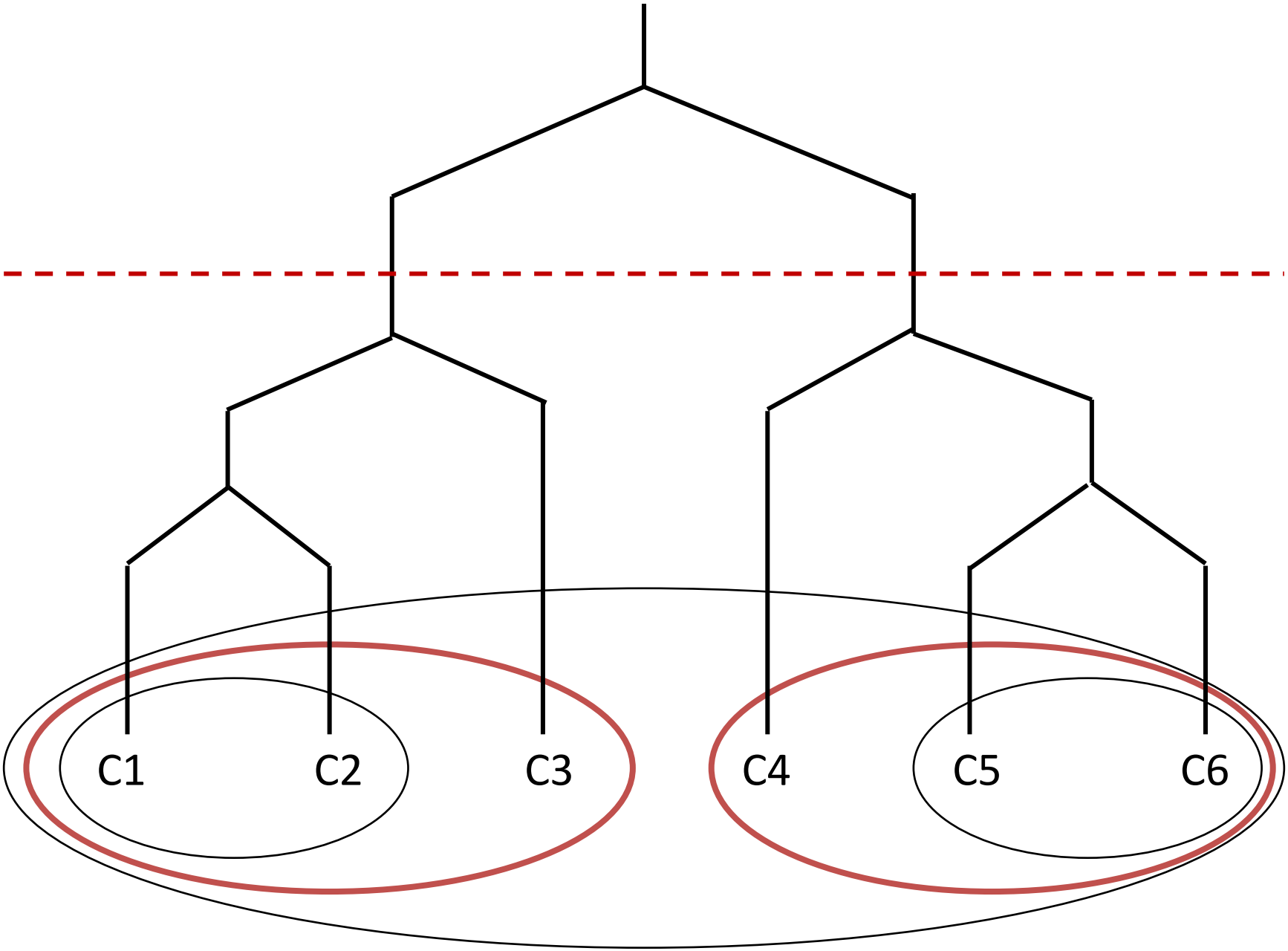

C1
C2
C3
C4
C5
C6

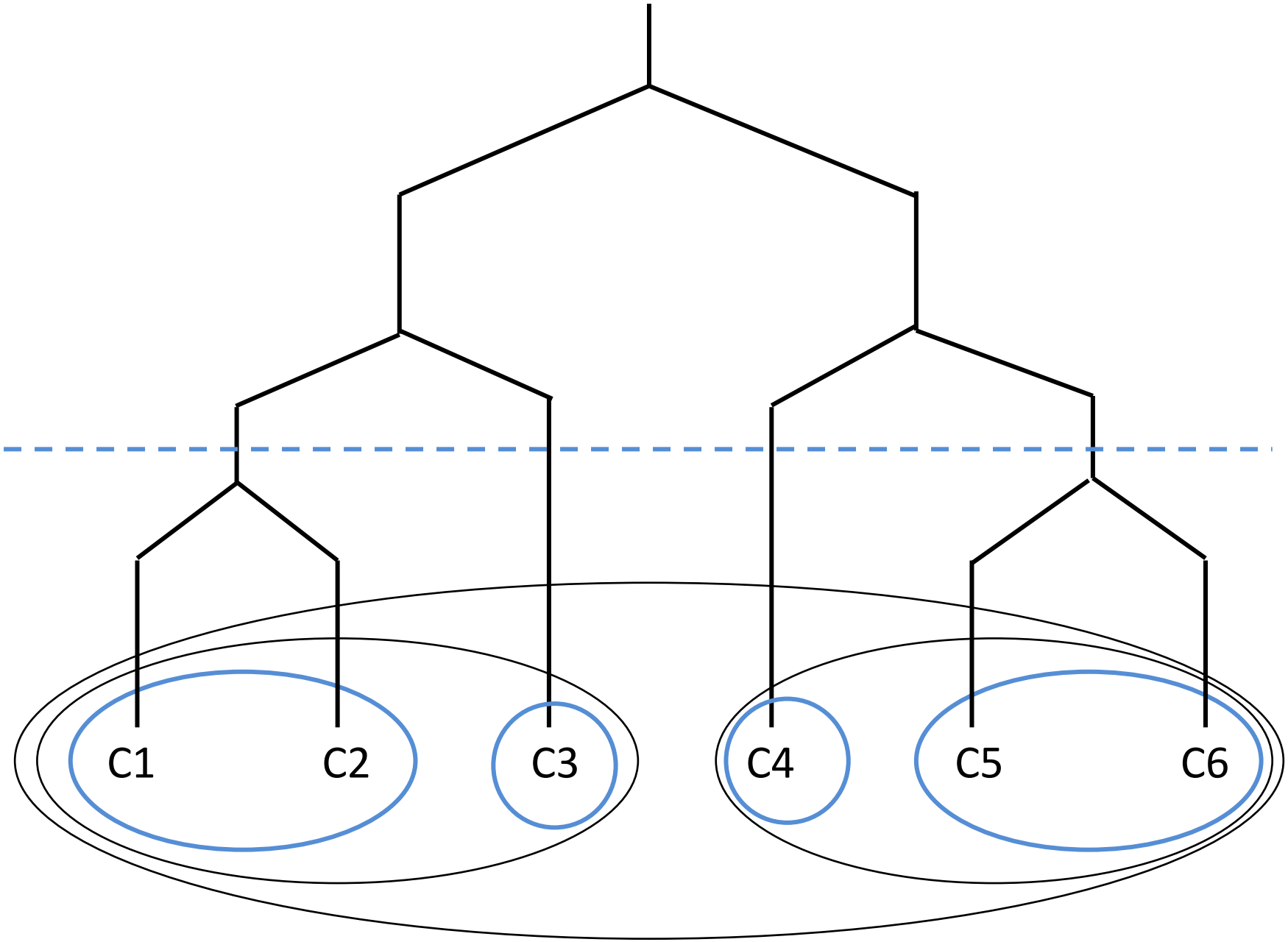

C1
C2
C3

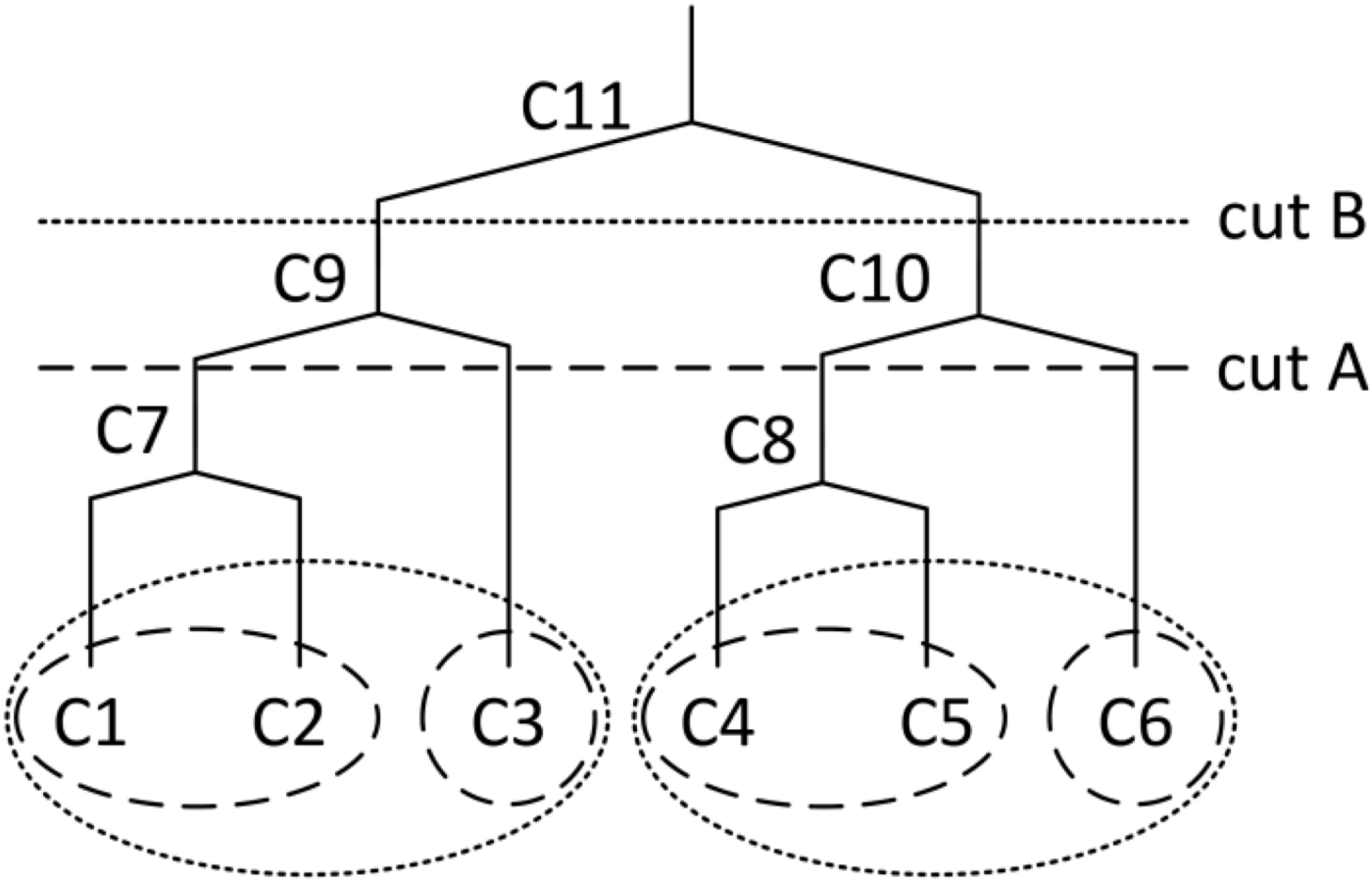

C4
C5
C6


C1
C2
C3
C4
C5
C6


C1
C2
C3
C4
C5
C6


C1
C2
C3
C4
C5
C6


C1
C2
C3
C4
C5
C6




A dendrogram as a result of agglomerative hierarchical clustering. By performing different cuts different clusters are obtained.

# Partitioning clustering

- no explicit structure between the clusters is considered
- their result is a set of *k* clusters, where *k* is given or automatically determined
- well suited for clustering large document data sets due to relatively low computational requirements

# K-means

- the most widely used flat clustering algorithm
- in the first step, *k* randomly selected cluster centers
- then, all objects are assigned to a cluster with the closest center
- in the following step, the cluster centroids are re-computed according to the positions of the objects in the clusters
- the steps of assignment of the objects to the clusters and re-computation of cluster centroids are repeated until a stopping criterion has been met
- the algorithm might arrive to a local optimum – the entire process might be repeated several times with a different seed, outliers might be excluded from the seed set, or the seed might be obtained from another method

# Clustering criterion functions

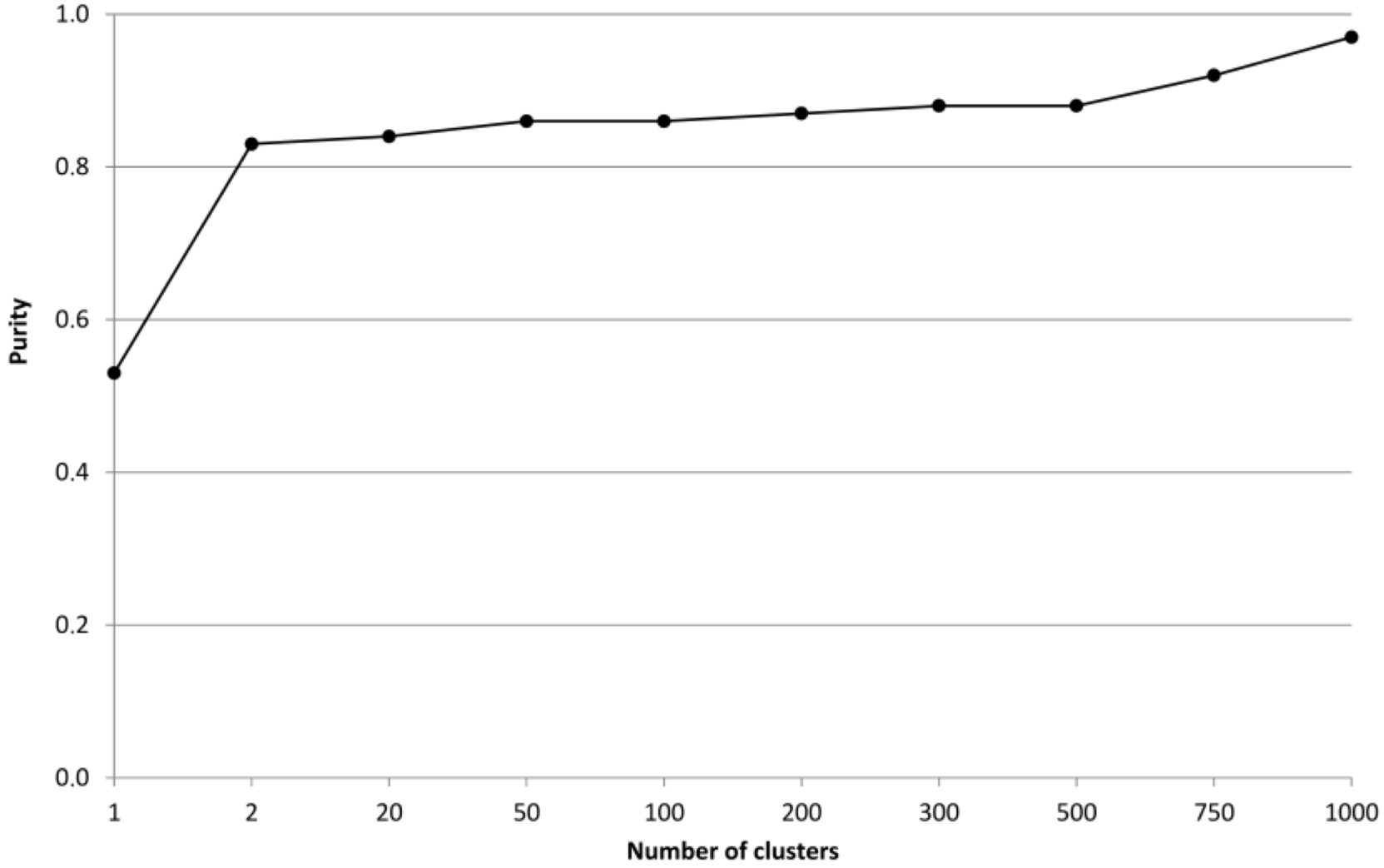
- functions representing the quality of flat clustering solutions
- during assigning the documents into the clusters, a particular clustering criterion (objective) function defined over the entire clustering solution is optimized (typically using a greedy strategy)

# Criterion functions

- internal – maximizing the similarity of documents in individual clusters while not considering the documents in different clusters
- external – focuses on optimization of dissimilarity of individual clusters
- hybrid – combine both internal and external criteria
- graph-based criterion functions – minimizing the edge-cut in a graph

# Evaluating the clustering process

- a correct result (labels) is not known unlike in a supervised learning problem
- the perfectness of the output is usually expected to be much lower that desired because only a human has a clear objective and can use some additional, external information
- a user typically wants a reasonable amount of clusters with sufficient quality
- approaches use
  - internal evaluation measures
  - external evaluation measures
  - expert opinion




Quality of clustering solutions with different numbers of clusters measured by the Purity criterion

# Internal evaluation measures

- derived from the data itself
- usually based on the criteria of
  - compactness – how much the objects are in a cluster related to each other (low pairwise or center-based distances in the cluster)
  - separation – how a cluster is separated from other clusters (distances between cluster centers, pairwise distances between objects from different clusters, or measures based on density)

# External evaluation measures

- class labels (ground truth) for the data to be clustered are available a priori
- entropy: $E_j = -\sum_i p_{ij} \log p_{ij}$ ($p_{ij}$ – probability that a member of the cluster *j* belongs to the class *i*)
- purity: $P_j = \frac{1}{n_j} * \max_{i=1..l} n_{ij}$ ($n_j$ is the number of instances in the cluster *j*, $n_{ij}$ is the number of instances of the class *i* in the cluster *j*, and *l* is the number of classes)

# Evaluation based on expert opinion

- suitable under many circumstances
- may reveal new insight into the data
- generally very expensive and demanding
- the results that are subjectively influenced are also not very well comparable

# Evaluation based on expert opinion

- visual examination
  - not possible when the number of dimensions > 3
  - objects might be projected from a multidimensional space to 2D or 3D (groups of objects that are separated in 2D or 3D space are also separated in a space with many dimensions
  - difficult to use for data with a very high number of dimensions

# Evaluation based on expert opinion

- examining the objects in individual clusters
  - usually not possible to consider not all of the documents but examine only some of them:
    - an average document – lies most closely to the remaining documents in the cluster (located near the centroid)
    - the most typical – located near the border of the cluster, far from the other clusters (the most different from all of the documents in the collection = most specific)
    - the least typical element – close the border of the cluster, closely to the remaining clusters; it is most similar to the documents in the other clusters

# Evaluation based on expert opinion with machine learning support

- an advanced automated method that would help in evaluating individual clusters is useful
- relevant attributes that sufficiently well characterize the documents in the clusters might be filtered
- a feature selection method using clusters as classes is an option

# Feature selection

# The purpose

- select (feature selection) or derive (feature extraction) a suitable subset of features from the original set
- decreasing the size of the vocabulary
  - important for, e.g., classifiers that are expensive to train
- eliminating noise
  - to prevent overfitting

# Approaches to FS

- filter methods
  - assign a metric to each feature
  - top *k* features are selected
  - attributes are treated separately
  - redundancy of attributes or dependency between attributes is not considered
  - independent on a task, based only on information in the data
  - popular in text mining because of low computational costs

# Approaches to FS

- wrapper methods
  - tailored to a particular task and algorithm
  - sets of features are evaluated in a search procedure
  - forward selection – starts with an empty set, adds attributes as long that the performance improves
  - backward elimination – starts with the full set, eliminates attributes as long that the performance improves
  - a greedy approach is usually applied because of computational complexity (exponential number of subsets)

# Approaches to FS

- embedded methods
  - feature selection is an integral part of an algorithm
  - for example, a decision tree learner selects the best attributes in a training process

# Basic FS algorithm

```
SelectFeatures (D, c, k)
1  ExtractVocabulary -> V
2  [] -> L
3  for each t in V do
4      ComputeFeatureUtility(D, t, c) -> A(t, c)
5      Append(L, A(t, c))
6  return FeaturesWithLargestValues(L, k)
```

D …a set of documents
c … a given class
k … the number of best features

http://nlp.stanford.edu/IR-book/html/htmledition/feature-selection-1.html

# Score-computing functions

- often work with some probabilities (e.g., the probability that a document *x* contains term *t*, the joint probability that a document *x* contains term *t* and also belongs to category $c_i$)

# Score-computing functions

- these probabilities can be estimated from the following statistical information:
  - $A_i$ = the number of the documents that contain the term *t* and also belong to category $c_i$
  - $B_i$ = the number of the documents that contain the term *t* but do not belong to category $c_i$
  - $C_i$ = the number of the documents that do not contain the term *t* but belong to category $c_i$, i.e., $N_i - A_i$
  - $D_i$ = the number of the documents that neither contain the term *t* nor belong to category $c_i$, i.e., $N_{all} - N_i - B_i$
  - $N_i$ = the total number of the documents that belong to category $c_i$
  - $N_{all}$ = the total number of all documents from the training data
  - $c_i$ – a class (i = *1..m*)

# $\chi^2$

- in statistics, $\chi^2$ test is applied to categorical data to evaluate whether two events are independent on each other
- independent events: $P(AB)=P(A)P(B)$, or $P(A|B)=P(A)$ or $P(B|A)=P(B)$
- in FS, the two events are occurrence of a class and a term

# $\chi^2$

$$X^2(D,T,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_t})^2}{E_{e_t e_c}}$$

$e_t$ = 1 … the document contains *t*, $e_t$ = 0 … the document does not contain t

$e_c$ = 1 … the document is in class *c*, $e_c$ = 0 … the document is not in class *c*

*N* … observed frequency

*E* … expected frequency given that *t* and *c* are independent (e.g., $E_{11}$ = *N*P(t)*P(c)*)

# $\chi^2$

- the above formula might be rewritten as

$$CHI = \frac{N_{all}(A_i D_i - C_i B_i)^2}{(A_i + C_i)(B_i + D_i)(A_i + B_i)(C_i + D_i)}$$

- high values of $\chi^2$ indicate that the hypothesis of independence of *t* and *c* is incorrect
- a weakness – not reliable for low frequency terms

# Word embeddings

# Basic ideas

- the bag-of-words model does not capture relations between words
- it would be possible to add other information to the existing words, e.g., *the word is A, it's POS tag is T, the previous word is B, and the following word is C* – too many features
- similar words should have similar properties → more than one dimension is needed to represent each word or feature
- each word is mapped to a continuous multidimensional space that has typically a few hundred dimensions, similar words are expected to be close to each other

# Basic ideas

- vector operations bring interesting results
  - vector("king") - vector("man") + vector("woman") ≈ vector("queen") which captures the male/female relationship
  - vector("paris") - vector("france") + vector("poland") ≈ vector("warsaw") – the capital-of relation
  - vector("cars") - vector("car") + vector("apple") ≈ vector("apples") – pluralisation relation

# Basic ideas

- word vectors can also significantly help in calculating true document similarities:
- e.g., *The queen visited the capital of US.* and *Elizabeth came to Washington in the USA.* share no words
- they, however, describe the same situation
  - word vectors of *Elisabeth* and *queen* and *USA* and *US* will be very close

# Context

- finding suitable values of the vector elements is based on the hypothesis stating that words in similar contexts have similar meanings
- a context = a set of words in somehow defined environment of the word (the same document, sentence, or a piece of a text of some length)
- very often, a few word to the left and right are considered (smaller window – more syntactic relations, bigger window – more semantic relations)

# Computing word embeddings

- supervised methods
  - require annotated data for a specific task (e.g., POS tagging)
  - embeddings are trained towards the given goal and can capture information relevant for the task
- unsupervised methods
  - do not require annotated data (they only compute embeddings)
  - the task is to predict a word given its context or deciding, whether a word can belong to a context given examples of real and randomly created word-context pairs
  - embeddings capture general syntactic and semantic relationships and can be applied in a wide variety of tasks

# word2vec

- a family of methods proposed by T. Mikolov that strongly attracted the NLP community to neural language models
- the prediction has two forms:
  - the word based on its context = Continuous bag-of-words (CBOW)
  - the context for a word = skip-gram

# fastText

- most of the techniques for learning word embeddings ignore sub-word information (prefixes, suffixes) which is very important for morphologically rich languages
- fastText (which is derived from word2vec) treats each word as a bag of character n-grams where the vectors are associated at the n-gram level
- the vector for a word is calculated as the sum of n-gram vectors
- this enables creating vectors for words that are not in the training data

# Word vectors aggregation

- to represent larger pieces of text (sentences, paragraphs, documents), word vectors need to be somehow aggregated (fixed length vectors are often needed)
- simple aggregation approaches – sum, average, or a weighted average (using tf-idf weights) of the word vectors
- more complex methods
  - Paragraph Vector
    - a larger piece of text, here a paragraph, should be useful in predicting other words too
    - a paragraph vector behaves like another word in the prediction task
  - a recursive neural network architecture to aggregate words vectors requiring parsing was proposed by Socher et al. (2013)
  - aggregation of word embedding to binary hash codes through Fisher kernel and hashing methods was proposed by Zhang et al. (2014)

# Workshop evaluation

https://www.fintech-ho2020.eu/free/app/evaluation-suptech-prague-4

Please fill in the evaluation form:

- Role: National Supervisor
- Scale evaluations
- Comments

MUNI