

Data preprocessing



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Tania Cerquitelli
Politecnico di Torino



Outline

- Data types and properties
- Data preparation
- Data preparation for document data
- Similarity and dissimilarity
- Correlation

Data types and properties

D^B_{MG}



Data Base and Data Mining Group of Politecnico di Torino



What is Data?

- Collection of ***data objects*** and their ***attributes***
- An ***attribute*** is a property or characteristic of an object
 - Examples: eye color of a person, temperature, etc.
 - Attribute is also known as variable, field, characteristic, dimension, or feature
- A collection of attributes describes an ***object***
 - Object is also known as record, point, case, sample, entity, or instance

Objects

Attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Attribute Values

- ***Attribute values*** are numbers or symbols assigned to an attribute for a particular object
- Distinction between attributes and attribute values
 - Same attribute can be mapped to different attribute values
 - Example: height can be measured in feet or meters
 - Different attributes can be mapped to the same set of values
 - Example: Attribute values for ID and age are integers
 - But properties of attribute values can be different



Attribute types

- There are different types of attributes
 - Nominal
 - Examples: ID numbers, eye color, zip codes
 - Ordinal
 - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
 - Interval
 - Examples: calendar dates
 - Ratio
 - Examples: temperature in Kelvin, length, time, counts



Properties of Attribute Values

- The type of an attribute depends on which of the following properties it possesses:
 - Distinctness: $= \neq$
 - Order: $< >$
 - Addition: $+ -$
 - Multiplication: $* /$

- Nominal attribute: distinctness
- Ordinal attribute: distinctness & order
- Interval attribute: distinctness, order & addition
- Ratio attribute: all 4 properties



Discrete and Continuous Attributes

■ Discrete Attribute

- Has only a finite or countably infinite set of values
- Examples: zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables.
- Note: **binary attributes** are a special case of discrete attributes

■ Continuous Attribute

- Has real numbers as attribute values
- Examples: temperature, height, or weight.
- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.



More Complicated Examples

- ID numbers
 - Nominal, ordinal, or interval?
- Number of cylinders in an automobile engine
 - Nominal, ordinal, or ratio?



Key Messages for Attribute Types

- The types of operations you choose should be “meaningful” for the type of data you have
 - Distinctness, order, meaningful intervals, and meaningful ratios are only four properties of data
- The data type you see – often numbers or strings – may not capture all the properties or may suggest properties that are not there
- Analysis may depend on these other properties of the data
 - Many statistical analyses depend only on the distribution
- Many times what is meaningful is measured by statistical significance
- But in the end, what is meaningful is measured by the domain



Data set types

- Record
 - Tables
 - Transaction Data
- Graph
 - *Relationships* among Objects (webpages)
 - Objects *are* graphs (molecules)
- Ordered
 - Spatial Data
 - Temporal Data
 - Sequential Data



Tabular Data

- A collection of records
 - Each record is characterized by a fixed set of attributes

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Transaction Data

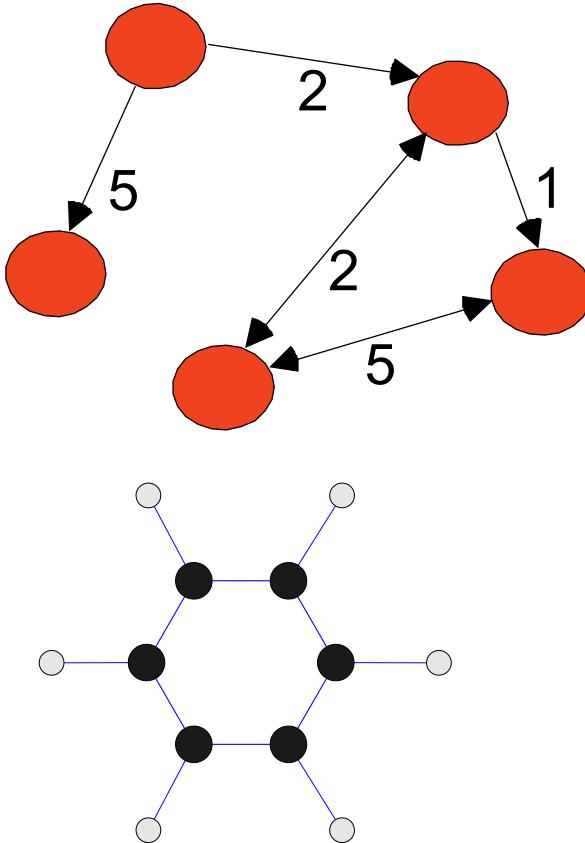
- A special type of record data, where
 - each record (transaction) involves a set of items.
 - For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk



Graph Data

- Examples: Generic graph, a molecule, and webpages



Benzene Molecule: C₆H₆

D
B
M

Useful Links:

- [Bibliography](#)
- Other Useful Web sites
 - [ACM SIGKDD](#)
 - [KDnuggets](#)
 - [The Data Mine](#)

Knowledge Discovery and Data Mining Bibliography

(Gets updated frequently, so visit often!)

- [Books](#)
- [General Data Mining](#)

Book References in Data Mining and Knowledge Discovery

Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Iyer, "Advances in Knowledge Discovery and Data Mining", AAAI Press/the MIT Press, 1996.

J. Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993. Michael Berry and Gordon Linoff, "Data Mining Techniques (For Marketing, Sales, and Customer Support)", John Wiley & Sons, 1997.

General Data Mining

Usama Fayyad, "Mining Databases: Towards Algorithms for Knowledge Discovery", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol. 21, no. 1, March 1998.

Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro, "Systems for knowledge Discovery in databases", IEEE Transactions on Knowledge and Data Engineering, 5(6):903-913, December 1993.

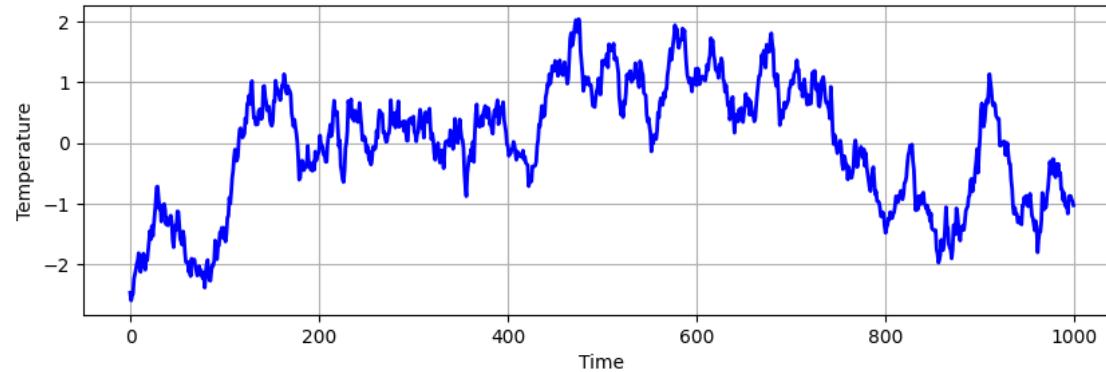


Ordered Data

- Examples:
 - Time series
 - Sequences of events
 - (e.g., transactions)

Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A,B) (t2:C,D) (t5:A,E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)





Ordered Data

- Genomic sequence data

```
GGTTCCGCCTTCAGCCCCGCGCC  
CGCAGGGCCCAGCCCCGCCGCCGTC  
GAGAAGGGCCCGCCTGGCGGGCG  
GGGGGAGGCAGGGCCGCCGAGC  
CCAACCGAGTCCGACCAAGGTGCC  
CCCTCTGCTCGGCCTAGACCTGA  
GCTCATTAGGCAGCAGGGACAG  
GCCAAGTAGAACACGCGAAGCGC  
TGGGCTGCCTGCTGCGACCAGGG
```

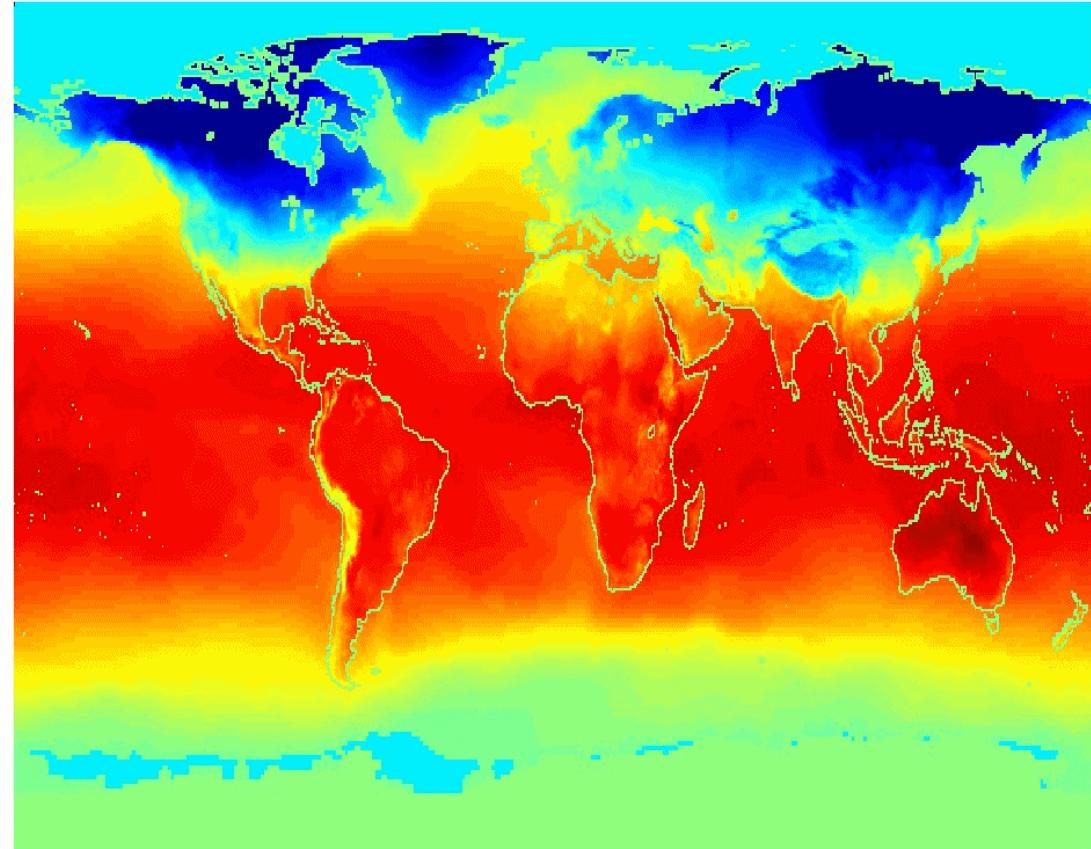


Ordered Data

- Spatio-Temporal Data

Average Monthly Temperature of land and ocean

Jan





Ordered data (text)

- Natural language («text») is a sequence of words, often **semi-structured** or **unstructured**:
 - Plain text can be organized in sentences, paragraphs, sections, documents
- Additional metedata/semantics may be available
 - **Web pages** are enriched with tags
 - **Documents in digital libraries** are enriched with metadata
- There are ways of representing text as «records»
 - (e.g., via TF-IDF – but some information is lost!)



Data Quality

- Poor data quality negatively affects many data processing efforts

“The most important point is that poor data quality is an unfolding disaster. Poor data quality costs the typical company at least ten percent (10%) of revenue; twenty percent (20%) is probably a better estimate.”

Thomas C. Redman, DM Review, August 2004

- Data mining example: a classification model for detecting people who are loan risks is built using poor data
 - Some credit-worthy candidates are denied loans
 - More loans are given to individuals that default



Data Quality

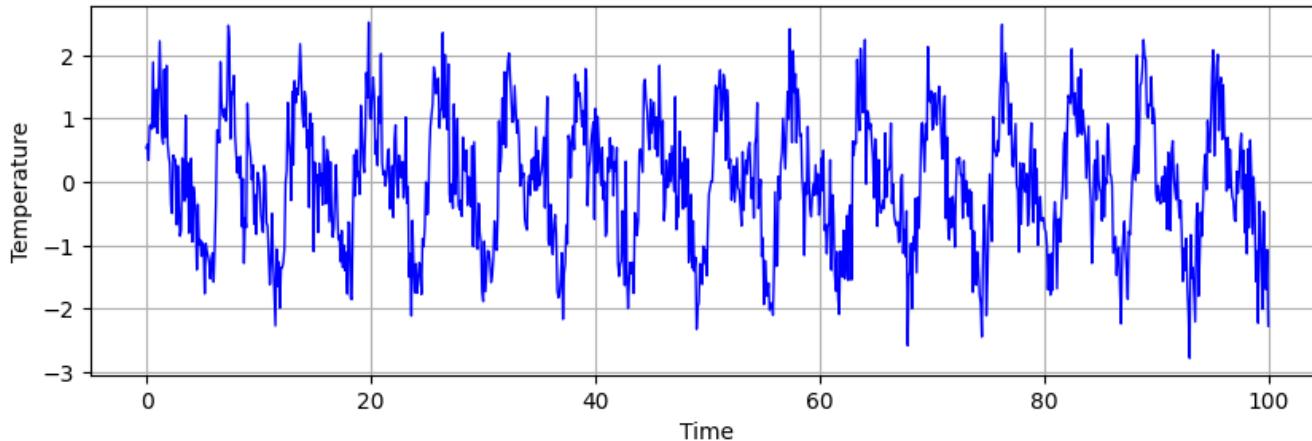
- What kinds of data quality problems?
- How can we detect problems with the data?
- What can we do about these problems?

- Examples of data quality problems
 - Noise and outliers
 - Missing values
 - Duplicate data
 - Wrong data



Noise

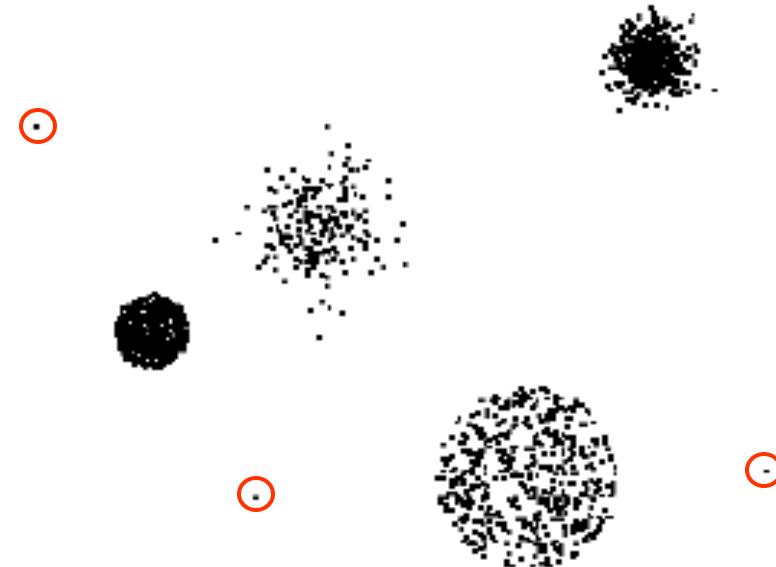
- Noise refers to modification of original values
- Sources of noise:
 - Data collection (measurements, human, ...)
 - Data processing (inconsistencies, missing values, ...)
 - Intrinsic (stochasticity of processes)





Outliers

- **Outliers** are data objects with characteristics that are considerably different than most of the other data objects in the data set
 - **Case 1:** Outliers are noise that interferes with data analysis
 - **Case 2:** Outliers are the goal of our analysis
 - Credit card fraud
 - Intrusion detection





Missing Values

- Reasons for missing values
 - Information is not collected
(e.g., people decline to give their age and weight)
 - Attributes may not be applicable to all cases
(e.g., annual income is not applicable to children)



Missing Values

- Handling missing values
 - Eliminate data objects or variables
 - Estimate missing values
 - Example: time series of temperature
 - Ignore the missing value during analysis
 - Fill with fixed values
 - Average value of the attribute for other points
 - 0, -1, ...
 - Impute the missing values
 - Use predictive model to estimate a value



Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
 - Major issue when merging data from heterogeneous sources
- Examples
 - Different words/abbreviations for the same concept (e.g., Street, St.)
- Data cleaning
 - Process of dealing with duplicate data issues

Data preparation

D^B_{MG}



Data Base and Data Mining Group of Politecnico di Torino



Data Preprocessing

- Aggregation
- Sampling
- Dimensionality Reduction
- Feature subset selection
- Feature creation
- Discretization and Binarization
- Attribute Transformation



Aggregation

- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
 - Data reduction
 - Reduce the number of attributes or objects
 - Change of scale
 - Cities aggregated into regions, states, countries, etc
 - More “stable” data
 - Aggregated data tends to have less variability



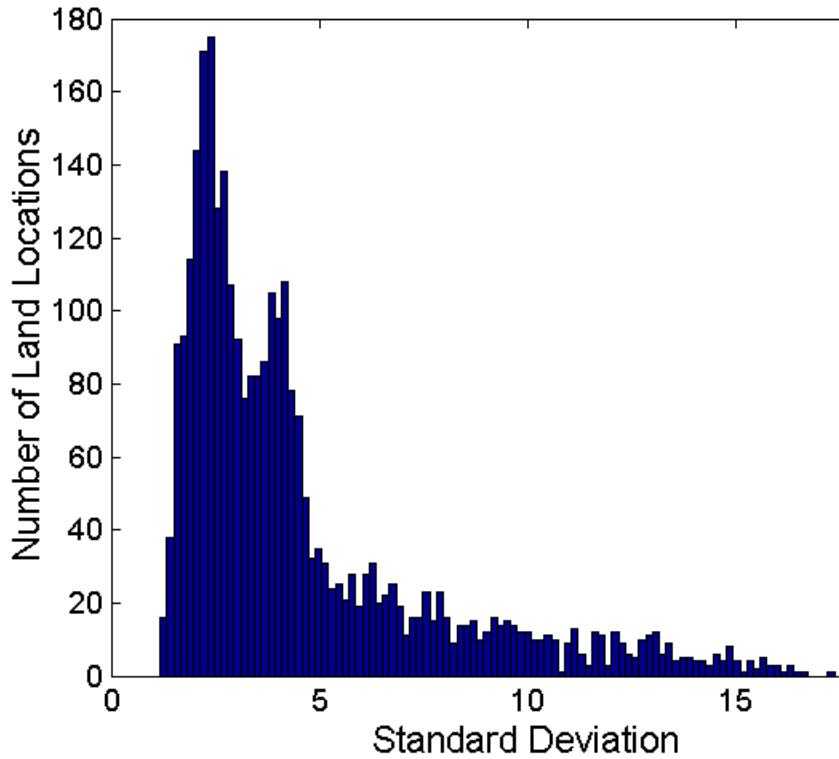
Example: Precipitation in Australia

- The next slide shows precipitation in Australia from the period 1982 to 1993
 - A histogram for the standard deviation of average monthly precipitation for 3,030 0.5° by 0.5° grid cells in Australia
 - A histogram for the standard deviation of the average yearly precipitation for the same locations.
- The average yearly precipitation has less variability than the average monthly precipitation.
- All precipitation measurements (and their standard deviations) are in centimeters.

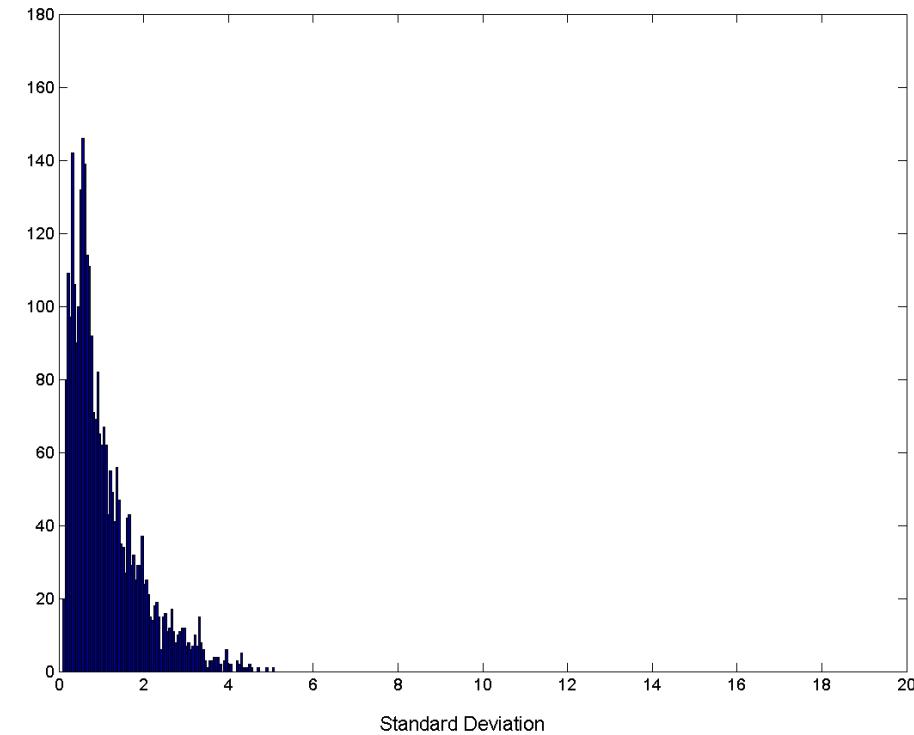


Aggregation

Variation of Precipitation in Australia



Standard Deviation of Average
Monthly Precipitation



Standard Deviation of Average
Yearly Precipitation



Data reduction

- Generates a reduced representation of the dataset
- This representation is smaller in volume, but it can provide similar analytical results
 - sampling
 - reduces the cardinality of the set
 - feature selection
 - reduces the number of attributes
 - discretization
 - reduces the cardinality of the attribute domain



Sampling

- Sampling is the main technique employed for data selection.
 - It is often used for both the preliminary investigation of the data and the final data analysis.
- Statisticians sample because **obtaining** the entire set of data of interest is too expensive or time consuming.
- Sampling is used in data mining because **processing** the entire set of data of interest is too expensive or time consuming.

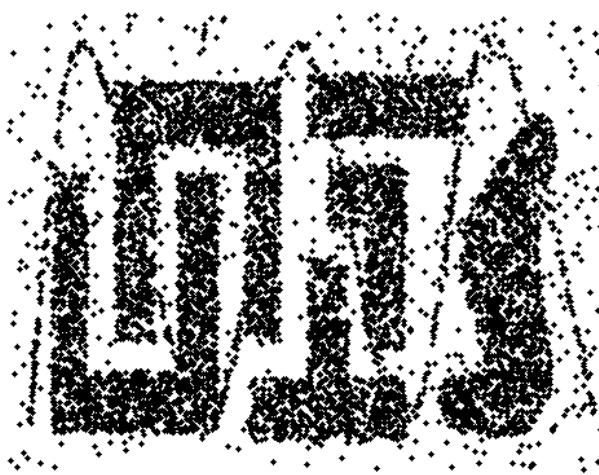


Sampling ...

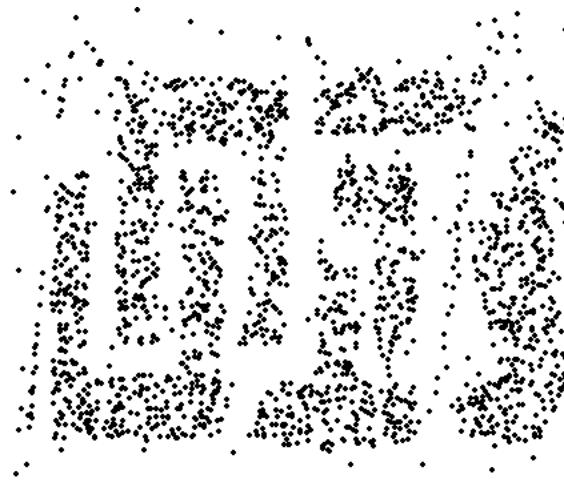
- The key principle for effective sampling is the following:
 - using a sample will work almost as well as using the entire data set, if the sample is representative
 - A sample is representative if it has approximately the same property (of interest) as the original set of data



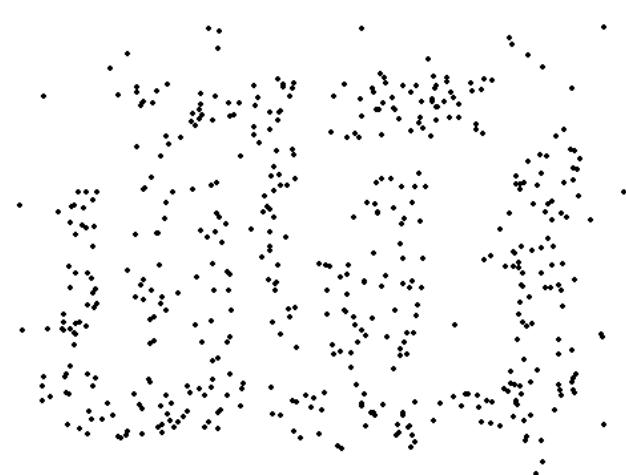
Sample Size: examples



8000 points



2000 Points



500 Points



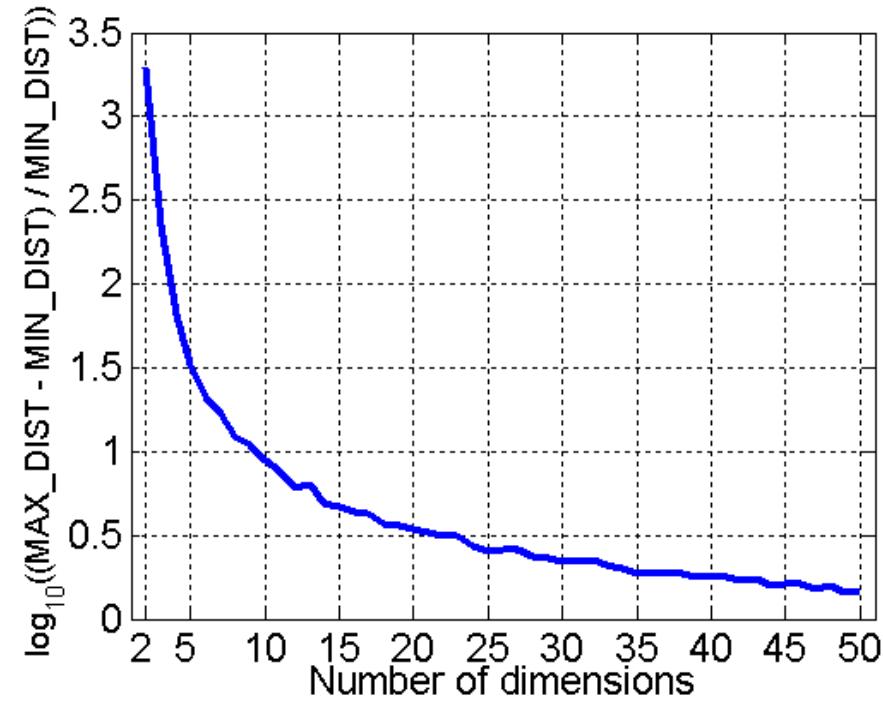
Types of Sampling

- Simple Random Sampling
 - There is an equal probability of selecting any particular item
 - Sampling without replacement
 - As each item is selected, it is removed from the population
 - Sampling with replacement
 - Objects are not removed from the population as they are selected for the sample.
 - In sampling with replacement, the same object can be picked up more than once
- Stratified sampling
 - Split the data into several partitions; then draw random samples from each partition



Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies
- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful



- Randomly generate 500 points
- Compute difference between max and min distance between any pair of points



Dimensionality Reduction

■ Purpose

- Avoid curse of dimensionality
- Reduce amount of time and memory required by data mining algorithms
- Allow data to be more easily visualized
- May help to eliminate irrelevant features or reduce noise

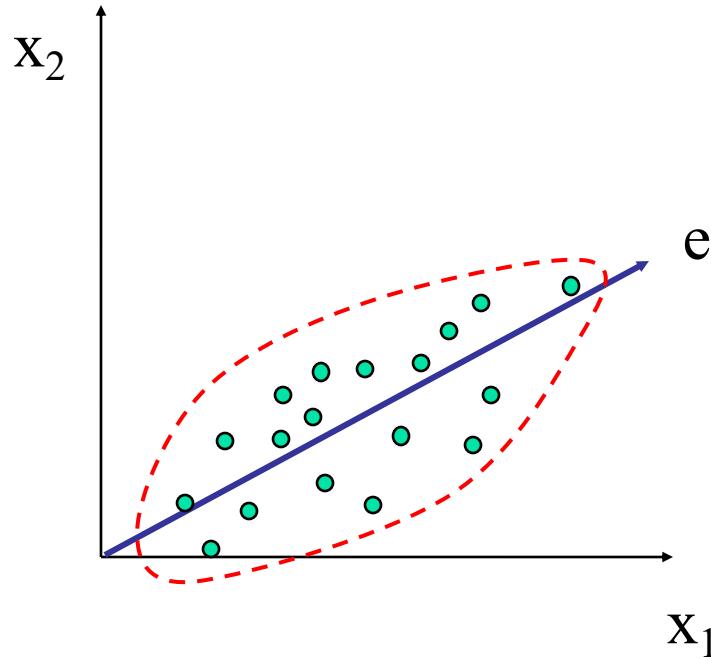
■ Techniques

- Principal Component Analysis (PCA)
- Singular Value Decomposition
- Others: supervised and non-linear techniques



Dimensionality Reduction: PCA

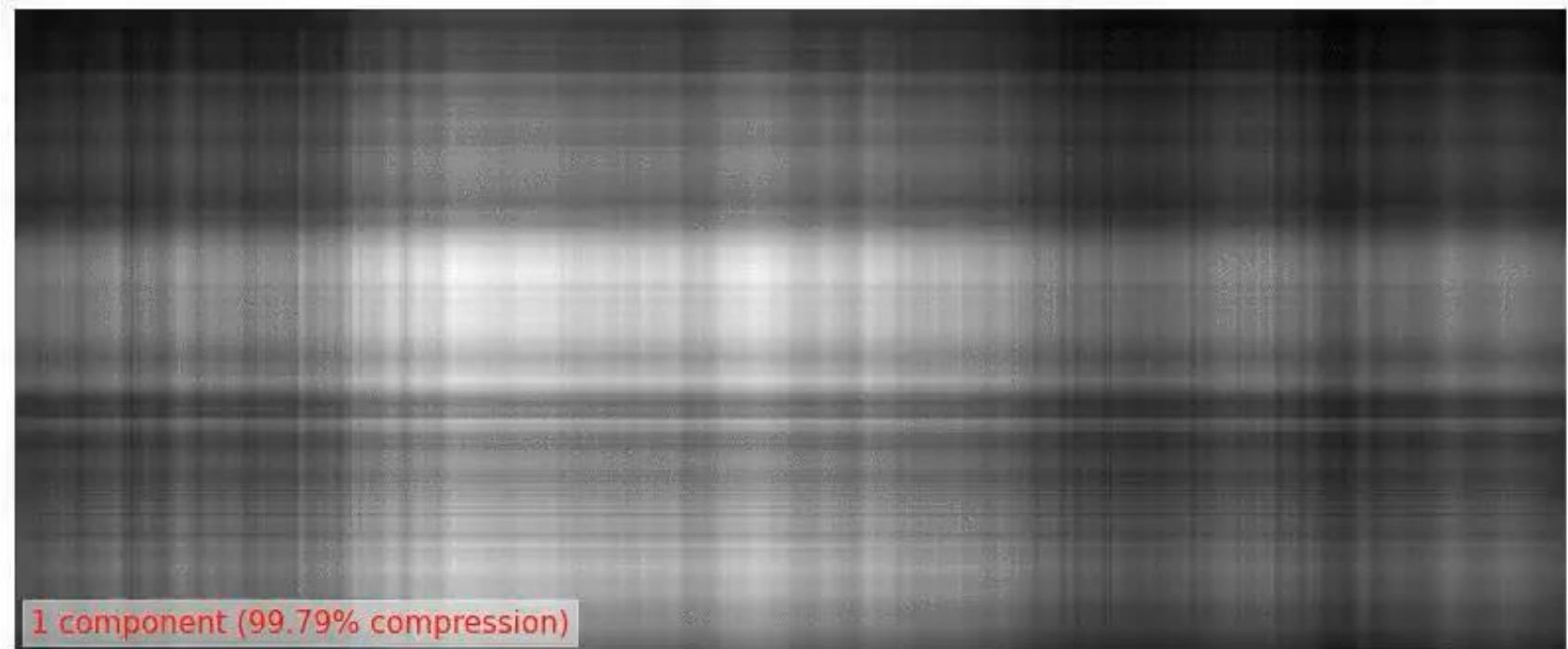
- Goal is to find a projection that captures the largest amount of variation in data





Dimensionality Reduction: SVD

```
n_components = 10
U, S, Vd = np.linalg.svd(im)
U[:, :n_components] @ np.diag(S[:n_components]) @ Vd[:n_components]
```





Feature Subset Selection

- Another way to reduce dimensionality of data
- Redundant features
 - duplicate much or all of the information contained in one or more other attributes
 - Example: purchase price of a product and the amount of sales tax paid
- Irrelevant features
 - contain no information that is useful for the data mining task at hand
 - Example: students' ID is irrelevant to the task of predicting students' GPA



Feature Subset Selection

- Techniques
 - Brute-force approach
 - Try all possible feature subsets as input to data mining algorithm
 - Embedded approaches
 - Feature selection occurs naturally as part of the data mining algorithm
 - Filter approaches
 - Features are selected before data mining algorithm is run
 - Wrapper approaches
 - Use the data mining algorithm as a black-box to find best subset of attributes



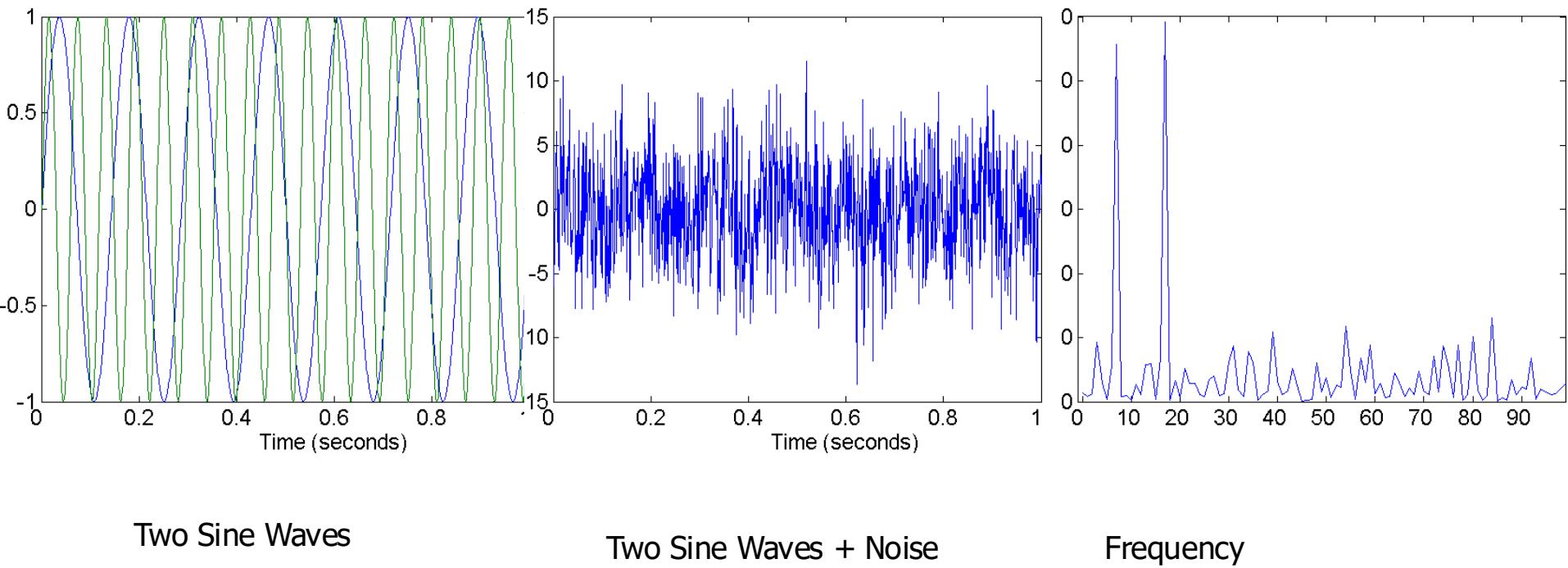
Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes
- Three general methodologies
 - Feature Extraction
 - domain-specific
 - Mapping Data to New Space
 - Feature Construction
 - combining features



Mapping Data to a New Space

- Fourier transform
- Wavelet transform





Discretization

- **Discretization** is the process of converting a continuous attribute into an ordinal attribute
 - A potentially infinite number of values are mapped into a small number of categories
 - Discretization is commonly used in classification
 - Many classification algorithms work best if both the independent and dependent variables have only a few values



Iris Sample Data Set

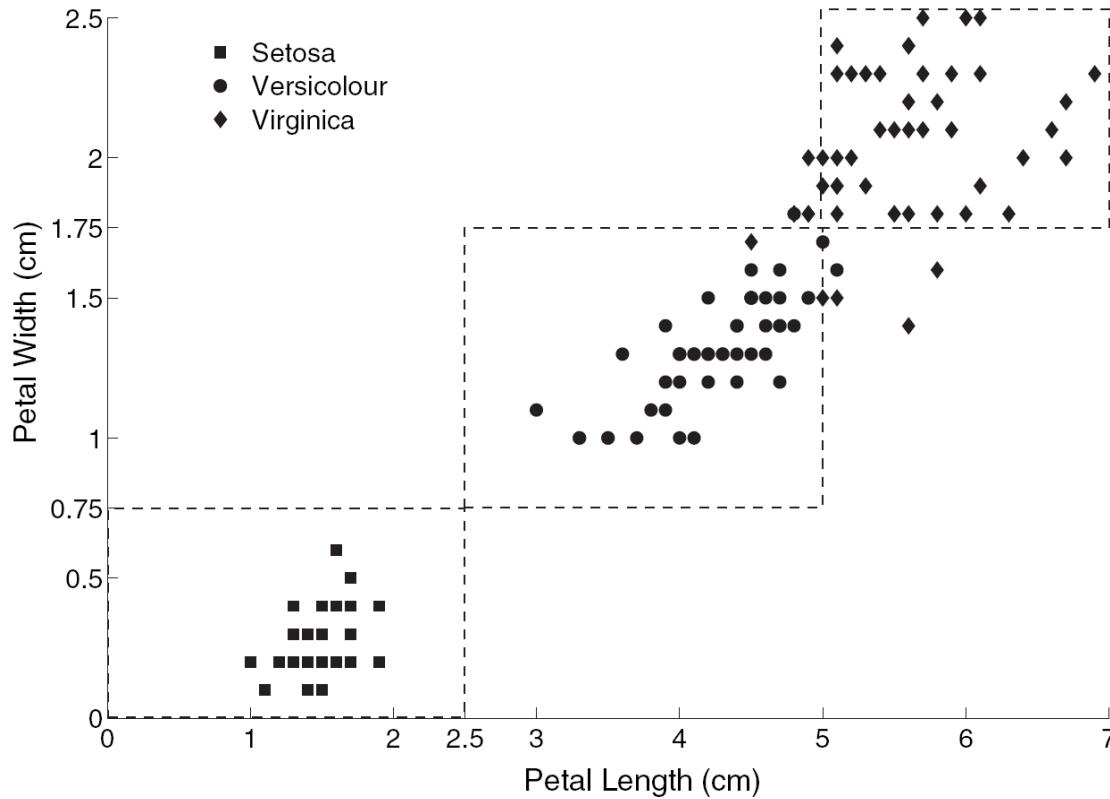
- Iris Plant data set
 - Can be obtained from the UCI Machine Learning Repository
<http://www.ics.uci.edu/~mlearn/MLRepository.html>
 - From the statistician Douglas Fisher
 - Three flower types (classes)
 - Setosa
 - Versicolour
 - Virginica
 - Four (non-class) attributes
 - Sepal width and length
 - Petal width and length



Virginica. Robert H. Mohlenbrock. USDA NRCS. 1995. Northeast wetland flora: Field office guide to plant species. Northeast National Technical Center, Chester, PA. Courtesy of USDA NRCS Wetland Science Institute.



Discretization: Iris Example



Petal width low or petal length low implies Setosa.

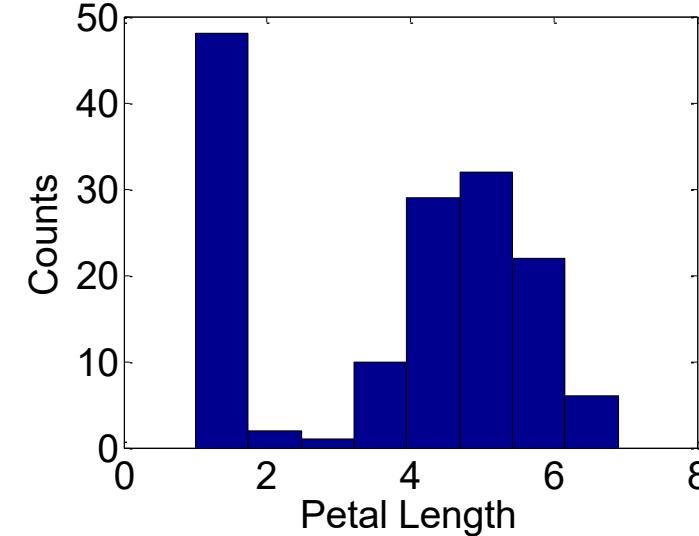
Petal width medium or petal length medium implies Versicolour.

Petal width high or petal length high implies Virginica.



Discretization: Iris Example ...

- How can we tell what the best discretization is?
 - Unsupervised discretization: find breaks in the data values
 - Example:
Petal Length



- Supervised discretization: Use class labels to find breaks

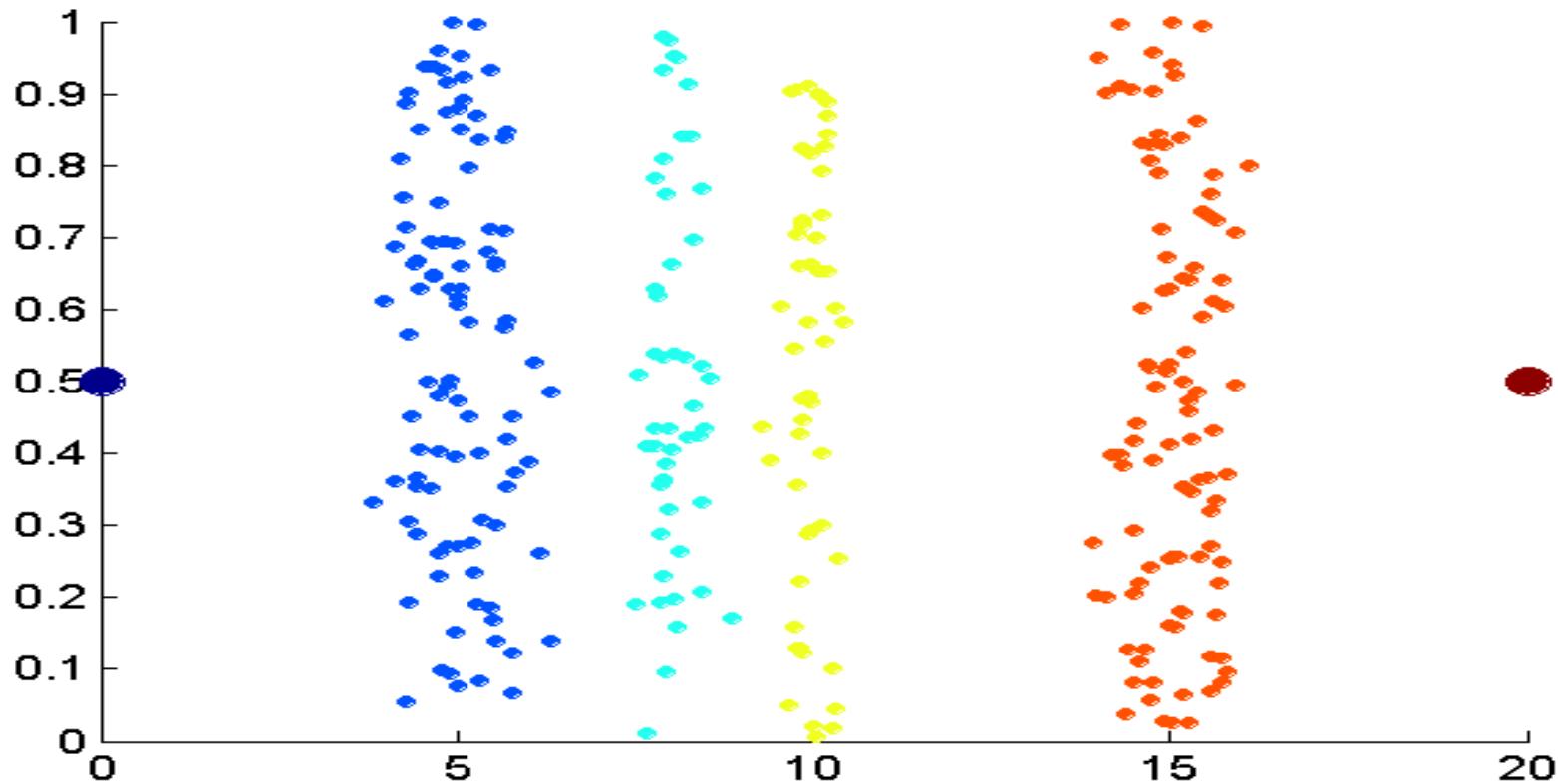


Discretization

- Examples of unsupervised discretization techniques
 - N intervals with the same width $W = (v_{\max} - v_{\min})/N$
 - Easy to implement
 - It can be badly affected by outliers and sparse data
 - Incremental approach
 - N intervals with (approximately) the same cardinality
 - It better fits sparse data and outliers
 - Non incremental approach
 - clustering
 - It fits well sparse data and outliers
 - analysis of data distribution
 - e.g., 4 intervals, one for each quartile



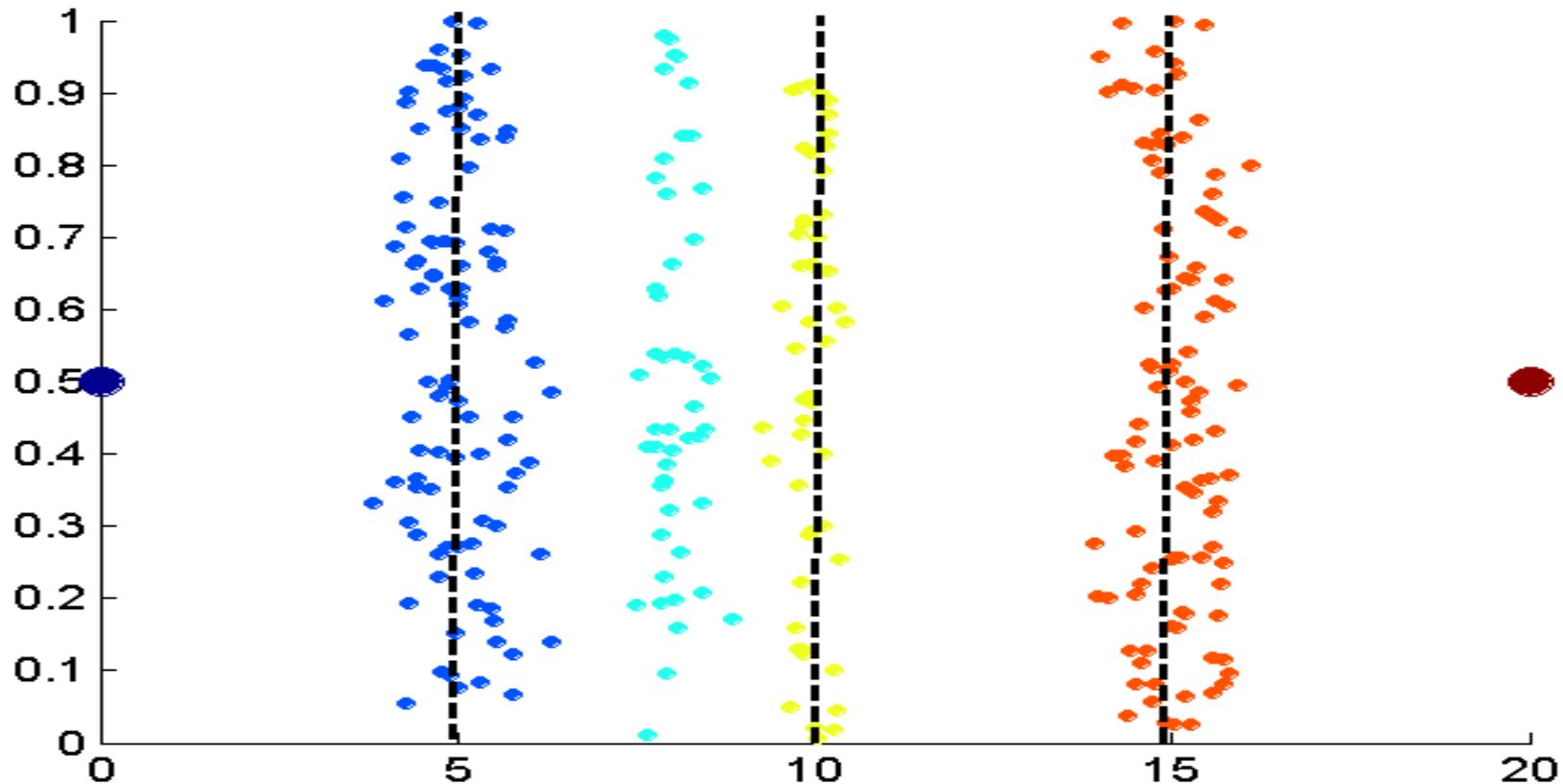
Example: unsupervised discretization technique



Data consists of four groups of points and two outliers. Data is one-dimensional, but a random y component is added to reduce overlap.



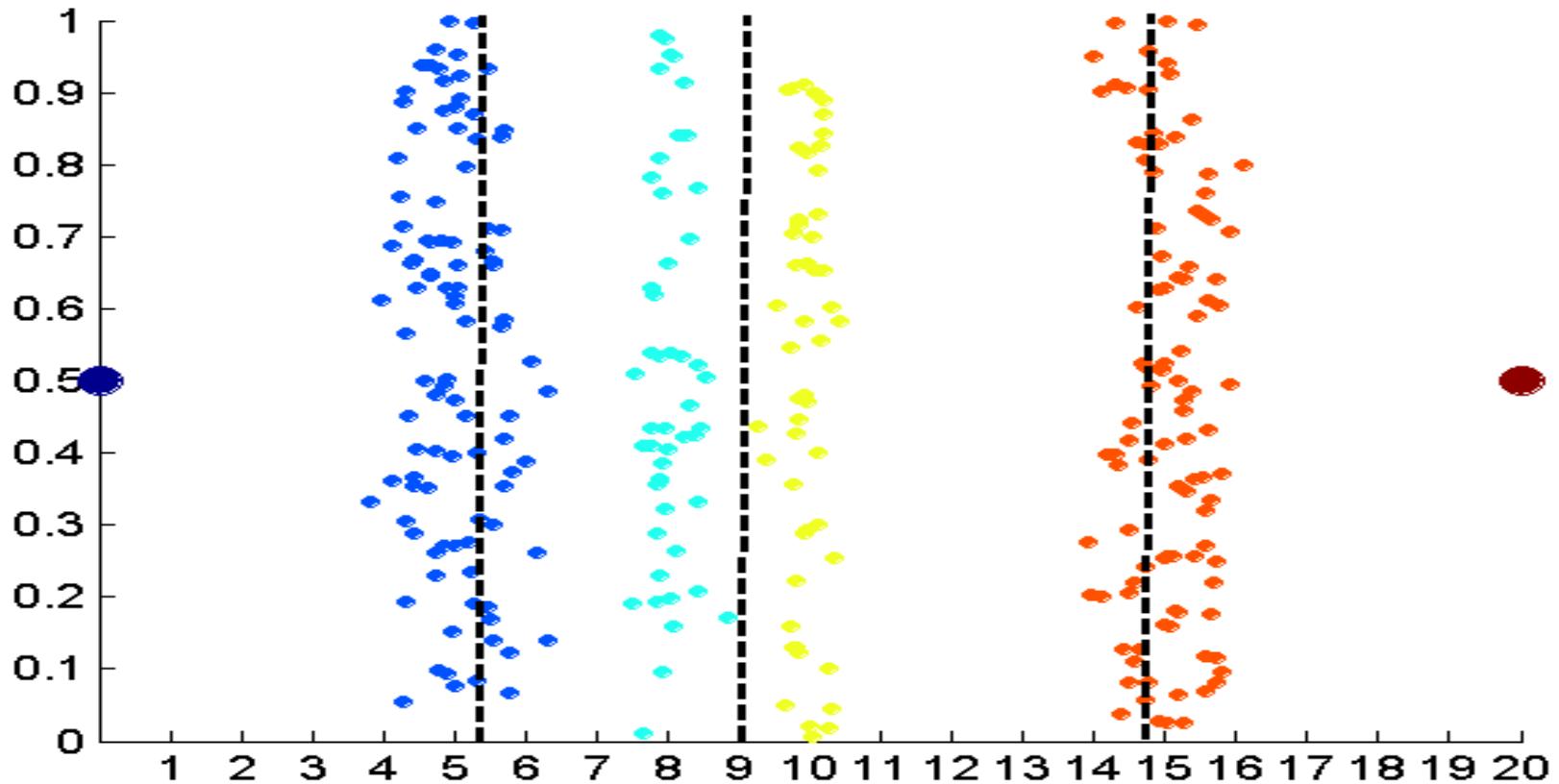
Example: unsupervised discretization technique



Equal interval width approach used to obtain 4 values.



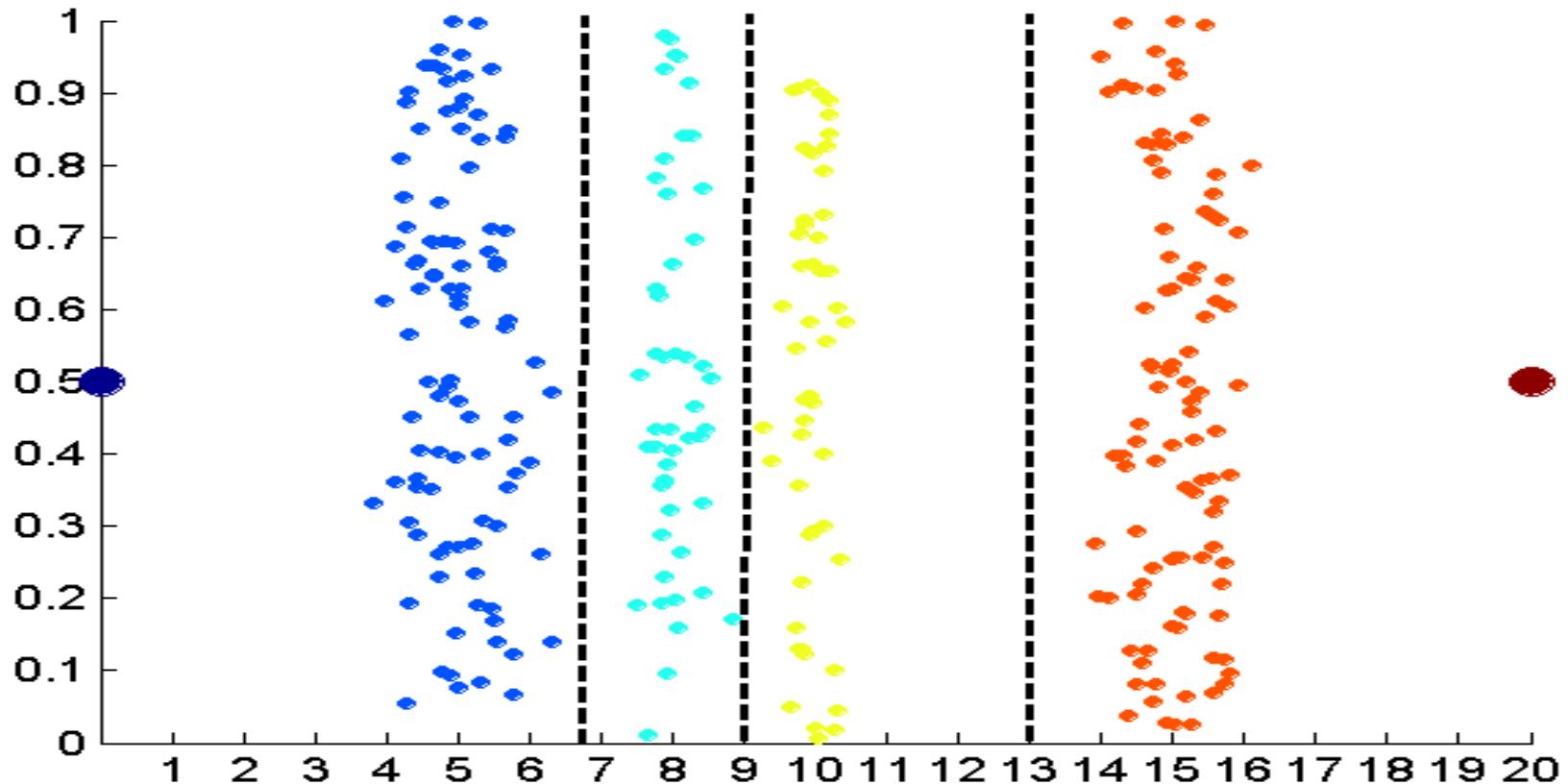
Example: unsupervised discretization technique



Equal frequency approach used to obtain 4 values.



Example: unsupervised discretization technique



K-means approach to obtain 4 values.



Binarization

- Binarization maps an attribute into one or more binary variables
- **Continuous** attribute: first map the attribute to a categorical one
 - Example: height measured as {low, medium, high}
- **Categorical** attribute
 - Mapping to a set of binary attributes
 - Example: Low, medium, high as 1 0 0, 0 1 0, 0 0 1
 - **One-hot encoding**
 - Only 1 bit takes value 1
 - It represents the specific value taken by the attribute



Attribute Transformation

- An **attribute transform** is a function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
 - Simple functions: x^k , $\log(x)$, e^x , $|x|$
- **Normalization**
 - Refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range
 - Take out unwanted, common signal, e.g., seasonality
- In statistics, **standardization** refers to subtracting off the means and dividing by the standard deviation



Normalization

- It is a type of data transformation
 - The values of an attribute are scaled so as to fall within a small specified range, typically [-1,+1] or [0,+1]
- Techniques
 - min-max normalization

$$v' = \frac{v - \min_A}{\max_A - \min_A} (new_max_A - new_min_A) + new_min_A$$

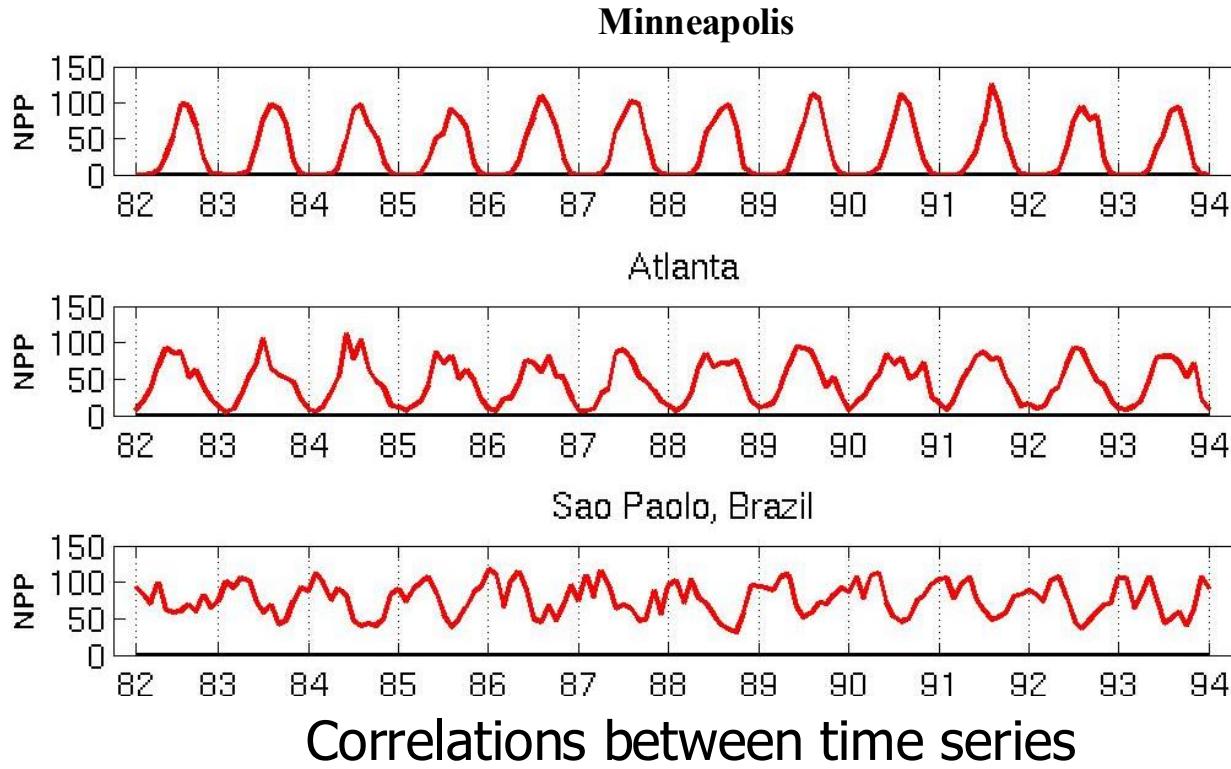
- z-score normalization
- decimal scaling

$$v' = \frac{v - \text{mean}_A}{\text{stand_dev}_A}$$

$$v' = \frac{v}{10^j} \quad j \text{ is the smallest integer such that } \max(|v'|) < 1$$



Example: Sample Time Series of Plant Growth

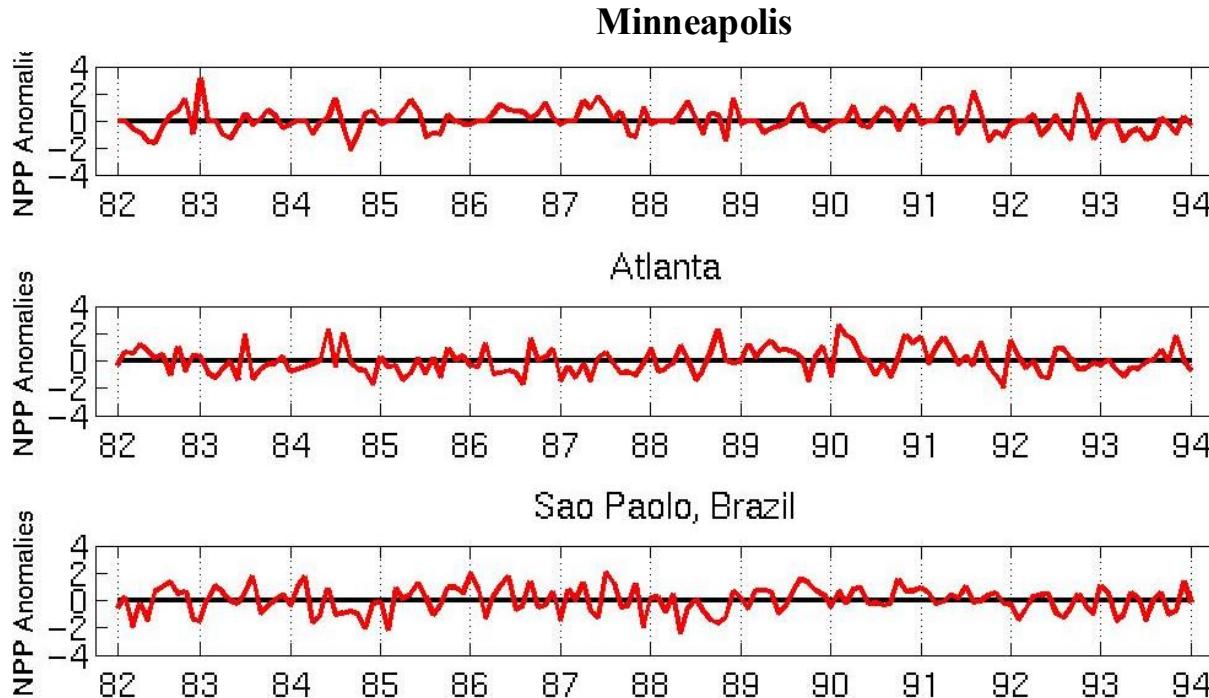


	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.7591	-0.7581
Atlanta	0.7591	1.0000	-0.5739
Sao Paolo	-0.7581	-0.5739	1.0000

Net Primary Production (NPP) is a measure of plant growth used by ecosystem scientists.



Example: Sample Time Series of Plant Growth



Normalized using monthly Z Score:
Subtract off monthly mean and divide by monthly standard deviation

Correlations between time series

	Minneapolis	Atlanta	Sao Paolo
Minneapolis	1.0000	0.0492	0.0906
Atlanta	0.0492	1.0000	-0.0154
Sao Paolo	0.0906	-0.0154	1.0000

From: Tan, Steinbach, Karpatne, Kumar, Introduction to Data Mining 2nd Ed., McGraw Hill 2018

Data preparation for document data

D^B_{MG}



Data Base and Data Mining Group of Politecnico di Torino



Document representation

- A document might be modeled in different ways
 - The choice heavily affects the quality of the mining result
- The most common representation models a document as **a set of features**
 - Each feature might represent a set of characters, a word, a term, a concept



Document processing

- It is the activity to generate a structured data representation of document data
- It includes five sequential steps
 - Document splitting
 - Tokenisation
 - Case normalisation
 - Stemming
 - Stopword removal



Document splitting

- Based on the data analytics goal, documents can be split into
 - sentences, paragraphs, or analyzed in their entire content
- Short documents are typically not split
 - e.g., emails or social posts
- Long documents can be
 - broken up into sections or paragraphs
 - analyzed as a whole



Tokenization

- It is the process of breaking text into sentences or text into tokens (i.e., words)
 - Identify sentence boundaries based on punctuation, capitalization
 - Separate words in sentences
 - Language-dependent



Case normalization

- This step converts each token to completely upper-case or lower-case characters
 - Capitalisation helps human readers differentiate, for example, between nouns and proper nouns and can be useful for automated algorithms as well
 - However, an upper-case word at the beginning of the sentence should be treated no differently than the same word in lower case appearing elsewhere in a document



Stemming

- Reduce a word to its root form (i.e., the **stem**)
 - It includes the identification and removal of prefixes, suffixes, and pluralisation
- It operates on a single word without knowledge of the context
 - It cannot discriminate between words which have different meanings depending on the part of speech
- Stemmers are
 - Easy to implement
 - Available for most spoken languages
 - Run significantly faster than lemmatization and POS tagging algorithms



Stopword elimination

- “Stop words” refers to the most common words in a language
 - E.g., prepositions, articles, conjunctions in English
- Stop words are filtered out before or after processing of textual data
 - They are likely to have little semantic meaning



Stopword elimination

- There is no single universal list of stop words used by all natural language processing tools
- Any group of words can be chosen as the stop words for a given purpose
 - different search engines use different stop word lists
 - Some of them remove lexical words, such as "want", from a query in order to improve performance
- Some tools specifically avoid removing these stop words to support phrase search

Weighted document representation

D^B_MG



Data Base and Data Mining Group of Politecnico di Torino



Text representation: feature vectors

- Most data mining algorithms are unable to directly process textual data in their original form
 - documents are transformed into a more manageable representation
- Documents are represented by **feature vectors**
- A feature is simply an entity without internal structure
 - A dimension of the feature space
- A document is represented as a vector in this space
 - a collection of features and their weights



Example

- Each document becomes a term vector
 - each term is a component (attribute) of the vector
 - the value of each component is the number of times the corresponding term occurs in the document

	team	coach	play	ball	score	game	winner	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0



Bag-of-word representation

- All words in a document are considered as separate features
 - the dimension of the feature space is equal to the number of different words in the entire document collection
- The feature vector of a document consists of a set of weights, one for each distinct word
- The methods for giving weights to the features may vary



Weighting schemes

- Binary
 - One, if the corresponding word is present in the document
 - Zero, otherwise
 - Occurrences of all words have the same importance
- Simple document frequency
 - The number of times in which the corresponding word occurs in the document
 - Most frequent words are not always representative of the document content



Weighting schemes

- More complex weighting schemes are possible to take into account the frequency of the word
 - in the document
 - in the section/paragraph
 - in the category (for indexed documents)
 - in the collection of documents



Weighting schemes

- Term frequency inverse document frequency (tf-idf)
 - Tf-idf of term t in document d of collection D (consisting of m documents)
$$\text{tf-idf}(t) = \text{freq}(t, d) * \log(m/\text{freq}(t, D))$$
 - Terms occurring frequently in a single document but rarely in the whole collection are preferred
- Suitable for
 - A single document consisting of many sections or subsections
 - A collection of *heterogeneous* documents



Tf-idf matrix example

major	malform	materi	matric	matrix	mean	measur	mechan	medicin	medium	medlin	method	methodolog	micro	microarch...	migrat	mo	model	molecular	morbid	moreov	mortal
0	0	0.153	0.051	0.021	0	0	0	0	0	0.051	0.069	0.072	0	0.020	0	0.034	0.072	0	0.072	0.063	
0.032	0.032	0.048	0.032	0.020	0.032	0.032	0.032	0.064	0.032	0.032	0.048	0.043	0.023	0.032	0.018	0.032	0.022	0.023	0.095	0.023	0.033
0	0	0	0	0.016	0	0.077	0.077	0	0	0	0.039	0.026	0	0.077	0.007	0.077	0	0	0	0	0.016
0.085	0.171	0	0	0	0	0	0	0	0.171	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.153	0.051	0.021	0	0	0	0	0	0.051	0.069	0.072	0	0.020	0	0.034	0.072	0	0.072	0.063	
0	0	0	0.052	0	0.105	0	0	0.052	0	0.052	0	0.035	0	0	0.020	0	0.035	0	0	0	0.022
0.093	0	0	0	0.039	0	0	0	0.093	0	0.093	0	0	0	0	0.018	0	0	0	0	0	0
0.077	0	0.154	0	0.032	0	0	0	0.077	0	0.077	0	0	0	0	0.030	0	0.052	0	0	0	0.032



- Most common words (e.g., “model”) have low values
- Peculiar words (e.g., “medlin”, “micro”, “methodolog”) have high values

Similarity and dissimilarity

D^B_MG



Data Base and Data Mining Group of Politecnico di Torino



Similarity and Dissimilarity

■ Similarity

- Numerical measure of how alike two data objects are
- Is higher when objects are more alike
- Often falls in the range [0,1]

■ Dissimilarity

- Numerical measure of how different are two data objects
- Lower when objects are more alike
- Minimum dissimilarity is often 0
- Upper limit varies

■ Proximity refers to a similarity or dissimilarity



Similarity/Dissimilarity for Simple Attributes

The following table shows the similarity and dissimilarity between two objects, x and y , with respect to a single, simple attribute.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = x - y /(n - 1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = x - y $	$s = -d, s = \frac{1}{1+d}, s = e^{-d},$ $s = 1 - \frac{d - \min_d}{\max_d - \min_d}$



Euclidean Distance

- Euclidean Distance

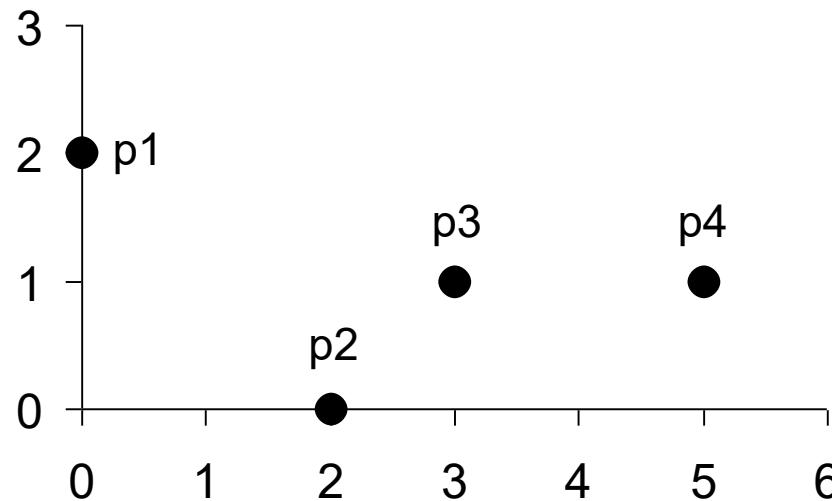
$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

where n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) of data objects \mathbf{x} and \mathbf{y} .

- Standardization is necessary, if scales differ.



Euclidean Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix



Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

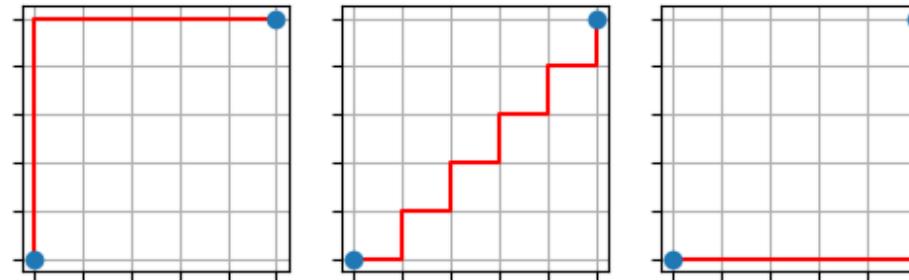
$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}$$

Where r is a parameter, n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) of data objects \mathbf{x} and \mathbf{y} .



Minkowski Distance: Examples

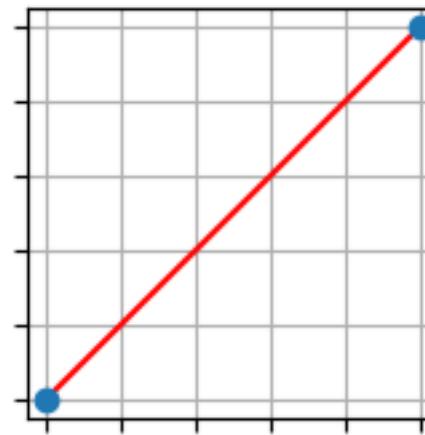
- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance
 - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors





Minkowski Distance: Examples

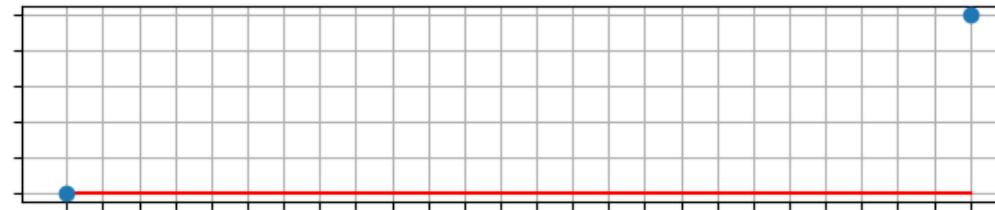
- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance
 - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$. Euclidean distance





Minkowski Distance: Examples

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance
 - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. “supremum” (L_{\max} norm, L_∞ norm) distance
 - This is the maximum difference between any component of the vectors





Minkowski Distance: Examples

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance
 - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. “supremum” (L_{\max} norm, L_∞ norm) distance
 - This is the maximum difference between any component of the vectors
- Do not confuse r with n , i.e., all these distances are defined for all numbers of dimensions.



Minkowski Distance

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix



Common Properties of a Distance

- Distances, such as the Euclidean distance, have some well-known properties.
 1. $d(x, y) \geq 0$ for all x and y and $d(x, y) = 0$ only if $x = y$. (Positive definiteness)
 2. $d(x, y) = d(y, x)$ for all x and y . (Symmetry)
 3. $d(x, z) \leq d(x, y) + d(y, z)$ for all points x, y , and z . (Triangle Inequality)where $d(x, y)$ is the distance (dissimilarity) between points (data objects) x and y .
- A distance that satisfies these properties is a **metric**



Common Properties of a Similarity

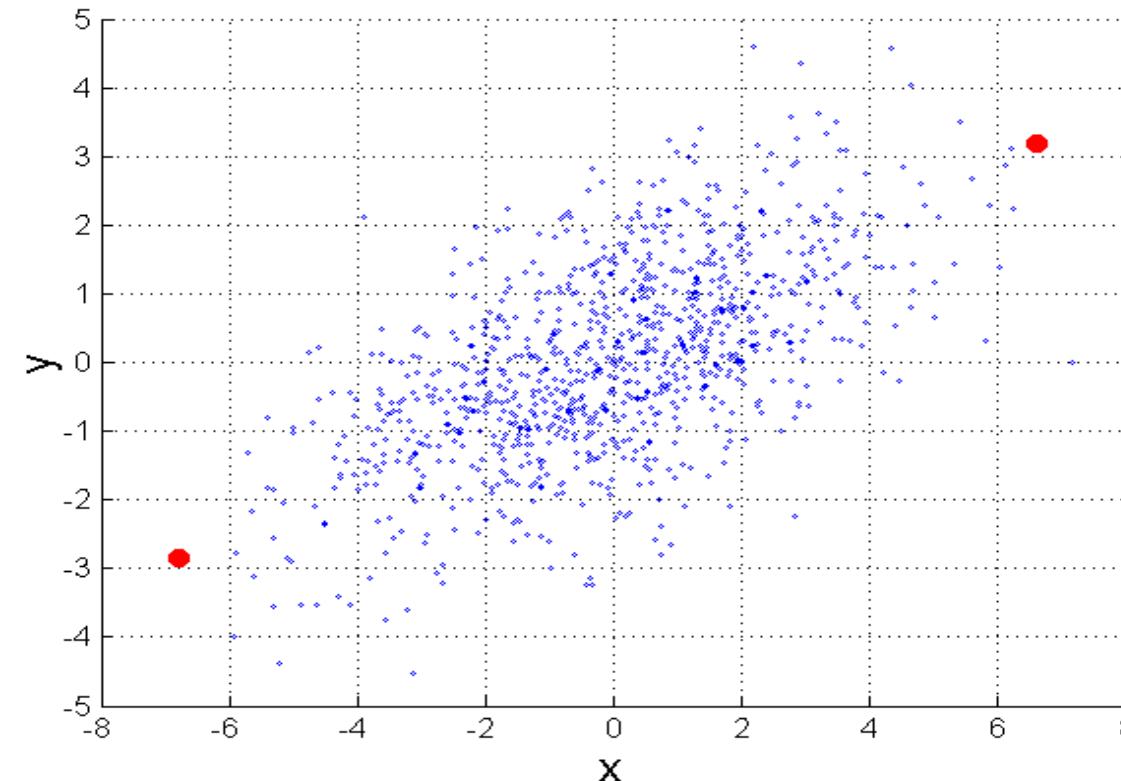
- Similarities also have some well known properties
 1. $s(x, y) = 1$ (or maximum similarity) only if $x = y$
 2. $s(x, y) = s(y, x)$ for all x and y (symmetry)

where $s(x, y)$ is the similarity between points (data objects) x and y



Mahalanobis Distance

$$\text{mahalanobis}(x, y) = (x - y)^T \Sigma^{-1} (x - y)$$

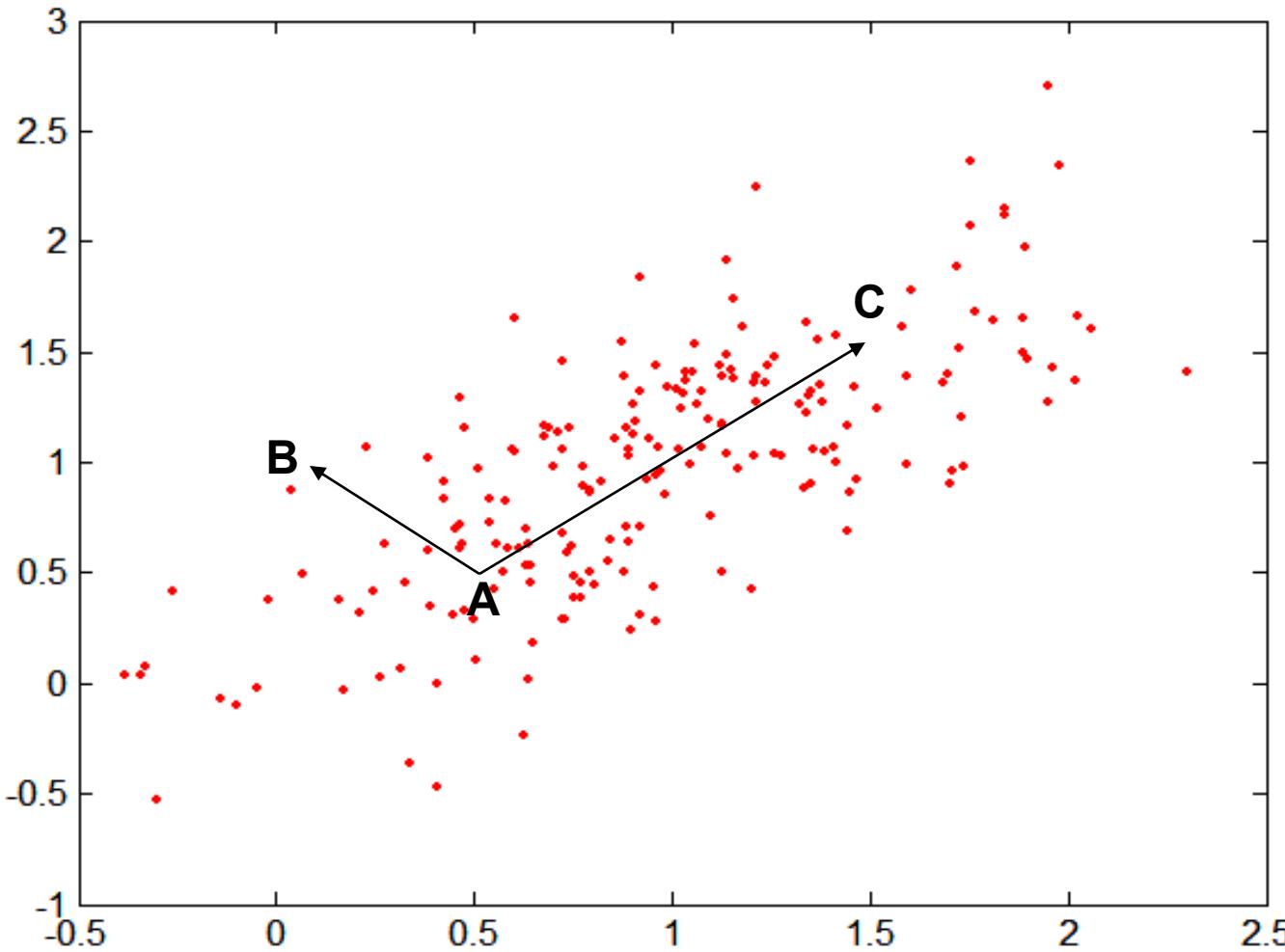


Σ is the covariance matrix

For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.



Mahalanobis Distance



Covariance Matrix:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

$\text{Mahal}(A, B) = 5$

$\text{Mahal}(A, C) = 4$



Similarity Between Binary Vectors

- Common situation is that objects p and q have only binary attributes
- Compute similarities using the following quantities

M_{01} = the number of attributes where p was 0 and q was 1

M_{10} = the number of attributes where p was 1 and q was 0

M_{00} = the number of attributes where p was 0 and q was 0

M_{11} = the number of attributes where p was 1 and q was 1

- Simple Matching and Jaccard Coefficients

SMC = number of matches / number of attributes

$$= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$$

J = number of 11 matches / number of not-both-zero attributes values

$$= (M_{11}) / (M_{01} + M_{10} + M_{11})$$



SMC versus Jaccard: Example

$p = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$

$q = 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1$

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)

$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)

$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)

$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$$\text{SMC} = (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$$

$$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$$



Cosine Similarity

- If d_1 and d_2 are two document vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|,$$

where \bullet indicates vector dot product and $\|d\|$ is the norm of vector d

- Example:

$$d_1 = \begin{bmatrix} 3 & 2 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

$$d_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \end{bmatrix}$$

$$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$$

$$\|d_1\| = (\sqrt{3^2+2^2+0^2+5^2+0^2+0^2+0^2+2^2+0^2+0^2})^{0.5} = (\sqrt{42})^{0.5} = 6.481$$

$$\|d_2\| = (\sqrt{1^2+0^2+0^2+0^2+0^2+0^2+0^2+1^2+0^2+2^2})^{0.5} = (\sqrt{6})^{0.5} = 2.245$$

$$\cos(d_1, d_2) = .3150$$



General Approach for Combining Similarities

- Sometimes attributes are of many different types, but an overall similarity is needed.
- For the k^{th} attribute, compute a similarity, $s_k(\mathbf{x}, \mathbf{y})$, in the range $[0, 1]$.
 - Define an indicator variable, δ_k , for the k^{th} attribute as follows:
$$\delta_k = 0 \text{ if the } k^{\text{th}} \text{ attribute is an asymmetric attribute and both objects have a value of 0, or if one of the objects has a missing value for the } k^{\text{th}} \text{ attribute}$$
$$\delta_k = 1 \text{ otherwise}$$
 - Compute $\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^n \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^n \delta_k}$



Using Weights to Combine Similarities

- May not want to treat all attributes the same.
 - Use non-negative weights ω_k

$$\text{■ } \text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^n \omega_k \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^n \omega_k \delta_k}$$

- Can also define a weighted form of distance

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n w_k |x_k - y_k|^r \right)^{1/r}$$

Correlation

D^B_{MG}



Data Base and Data Mining Group of Politecnico di Torino



Data correlation

- Measure of the linear relationship between two data objects
 - having binary or continuous variables
- Useful during the data exploration phase
 - To be better aware of data properties
- Analysis of feature correlation
 - Correlated features should be removed
 - simplifying the next analytics steps
 - improving the performance of the data-driven algorithms



Pearson's correlation

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard_deviation}(\mathbf{x}) * \text{standard_deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x s_y},$$

where we are using the following standard statistical notation and definitions

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n - 1} \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y})$$

$$\text{standard_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n - 1} \sum_{k=1}^n (x_k - \bar{x})^2}$$

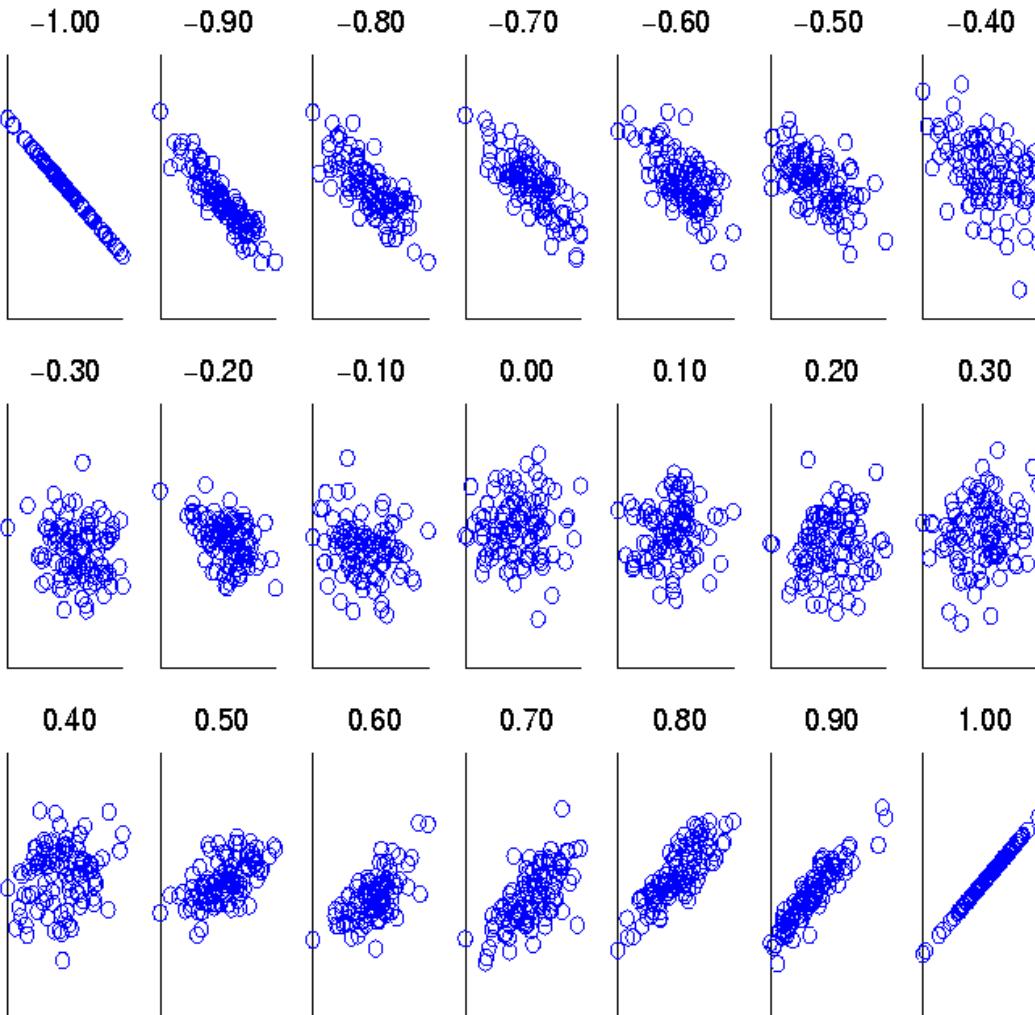
$$\text{standard_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n - 1} \sum_{k=1}^n (y_k - \bar{y})^2}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k \text{ is the mean of } \mathbf{x}$$

$$\bar{y} = \frac{1}{n} \sum_{k=1}^n y_k \text{ is the mean of } \mathbf{y}$$



Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

Perfect linear correlation when value is 1 or -1



Drawback of Correlation

- $\mathbf{x} = (-3, -2, -1, 0, 1, 2, 3)$

- $\mathbf{y} = (9, 4, 1, 0, 1, 4, 9)$

$$y_i = x_i^2$$

- $\text{mean}(\mathbf{x}) = 0, \text{mean}(\mathbf{y}) = 4$
- $\text{std}(\mathbf{x}) = 2.16, \text{std}(\mathbf{y}) = 3.74$

$$\begin{aligned}\text{corr} &= (-3)(5)+(-2)(0)+(-1)(-3)+(0)(-4)+(1)(-3)+(2)(0)+3(5) / (6 * 2.16 * 3.74) \\ &= 0\end{aligned}$$

Classification fundamentals



Data Base and Data Mining Group of Politecnico di Torino

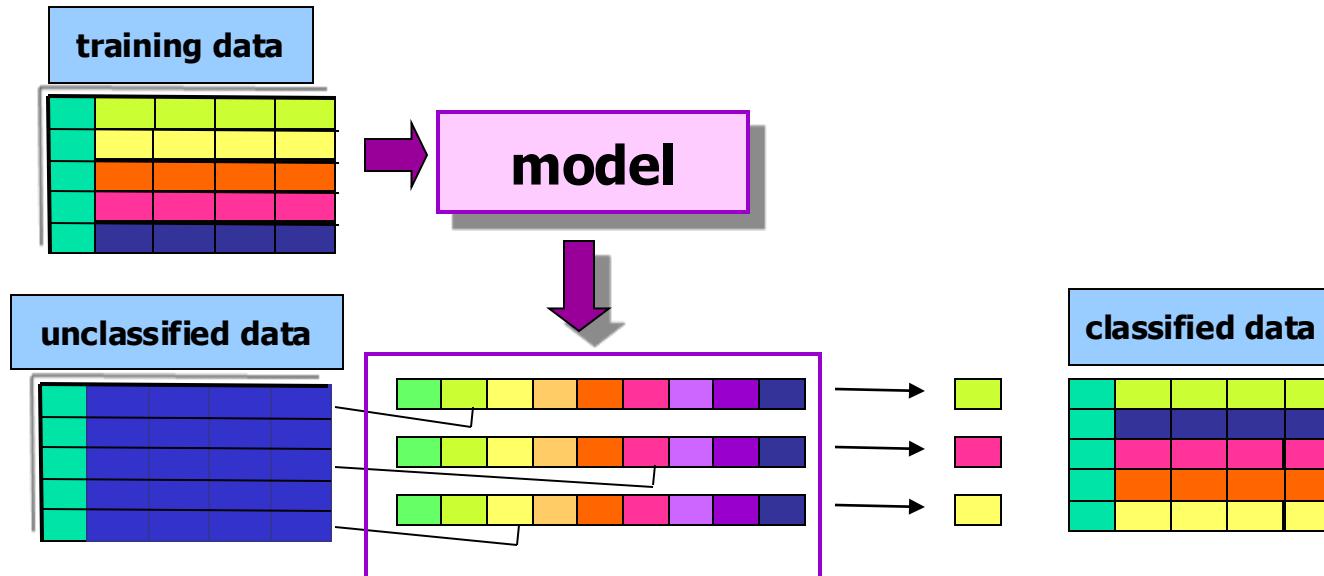
Elena Baralis
Politecnico di Torino



Classification

■ Objectives

- prediction of a class label
- definition of an interpretable model of a given phenomenon

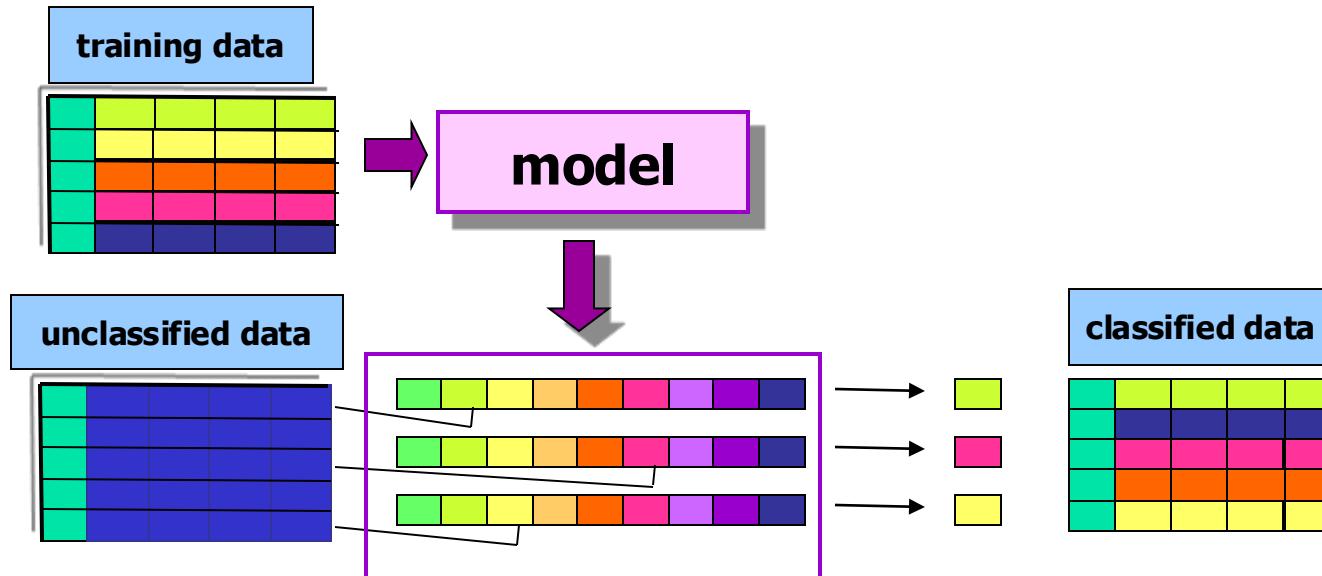




Classification

■ Applications

- detection of customer propensity to leave a company (churn or attrition)
- fraud detection
- classification of different pathology types
- ...





Classification: definition

- Given
 - a collection of class labels
 - a collection of data objects labelled with a class label
- Find a descriptive profile of each class, which will allow the assignment of unlabeled objects to the appropriate class



Definitions

- Training set
 - Collection of labeled data objects used to learn the classification model
- Test set
 - Collection of labeled data objects used to validate the classification model



Classification techniques

- Decision trees
- Classification rules
- Association rules
- Neural Networks
- Naïve Bayes and Bayesian Networks
- k-Nearest Neighbours (k-NN)
- Support Vector Machines (SVM)
- ...



Evaluation of classification techniques

- Accuracy
 - quality of the prediction
- Interpretability
 - model interpretability
 - model compactness
- Incrementality
 - model update in presence of newly labelled record
- Efficiency
 - model building time
 - classification time
- Scalability
 - training set size
 - attribute number
- Robustness
 - noise, missing data

Decision trees



Data Base and Data Mining Group of Politecnico di Torino

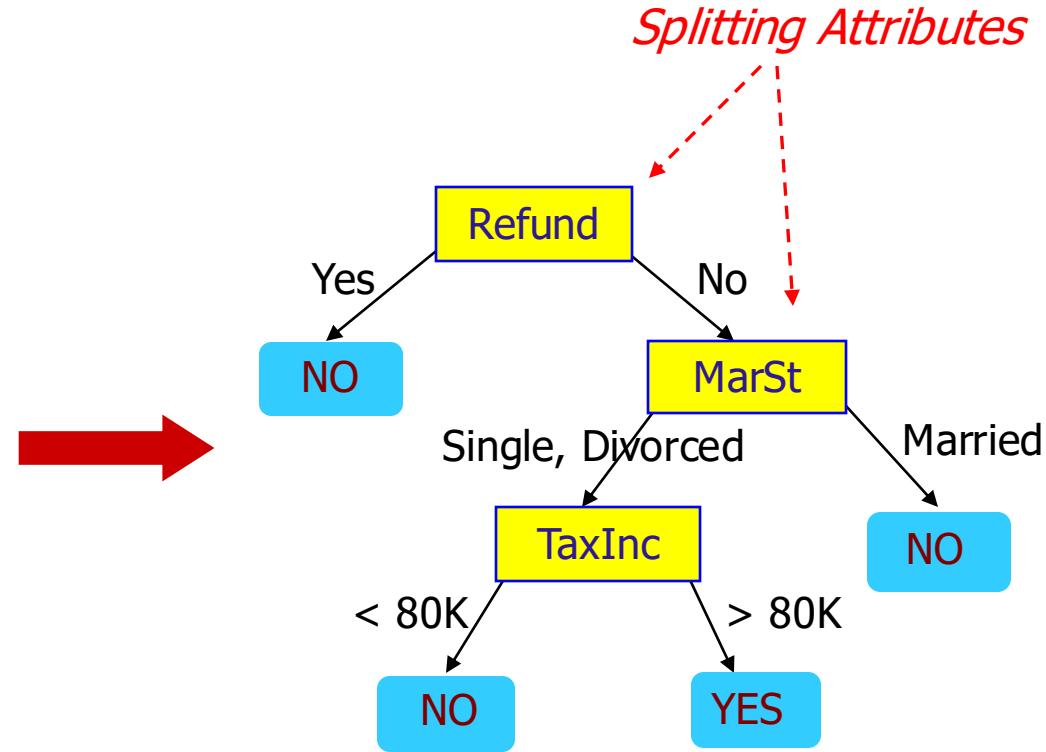
Elena Baralis
Politecnico di Torino



Example of decision tree

Categorical
categorical
continuous
class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



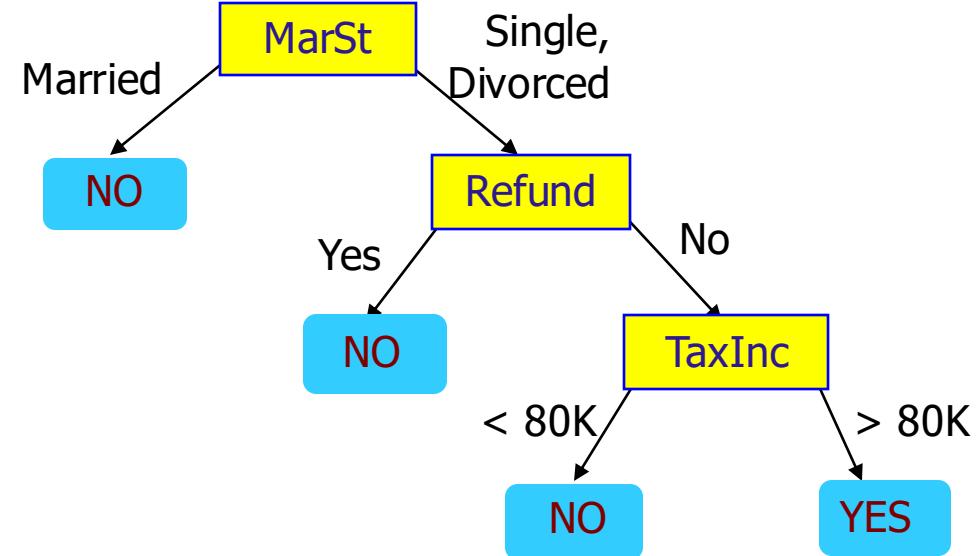
Training Data

Model: Decision Tree



Another example of decision tree

Tid	Refund	Marital Status	Taxable Income	Cheat	categorical	categorical	continuous	class
1	Yes	Single	125K	No				
2	No	Married	100K	No				
3	No	Single	70K	No				
4	Yes	Married	120K	No				
5	No	Divorced	95K	Yes				
6	No	Married	60K	No				
7	Yes	Divorced	220K	No				
8	No	Single	85K	Yes				
9	No	Married	75K	No				
10	No	Single	90K	Yes				

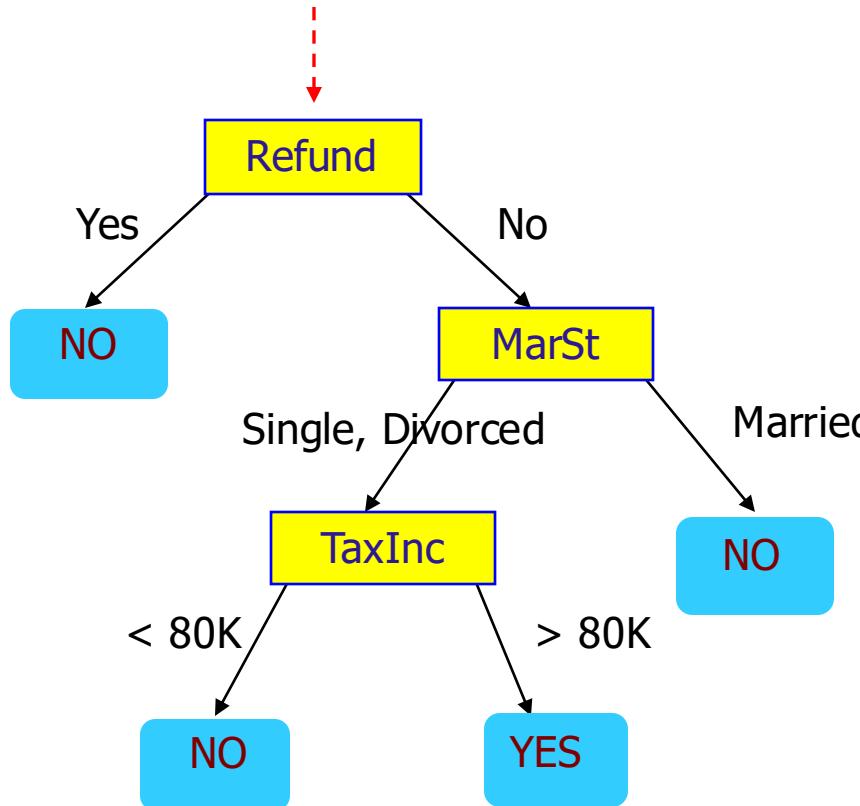


There could be more than one tree that fits the same data!



Apply Model to Test Data

Start from the root of tree.



Test Data

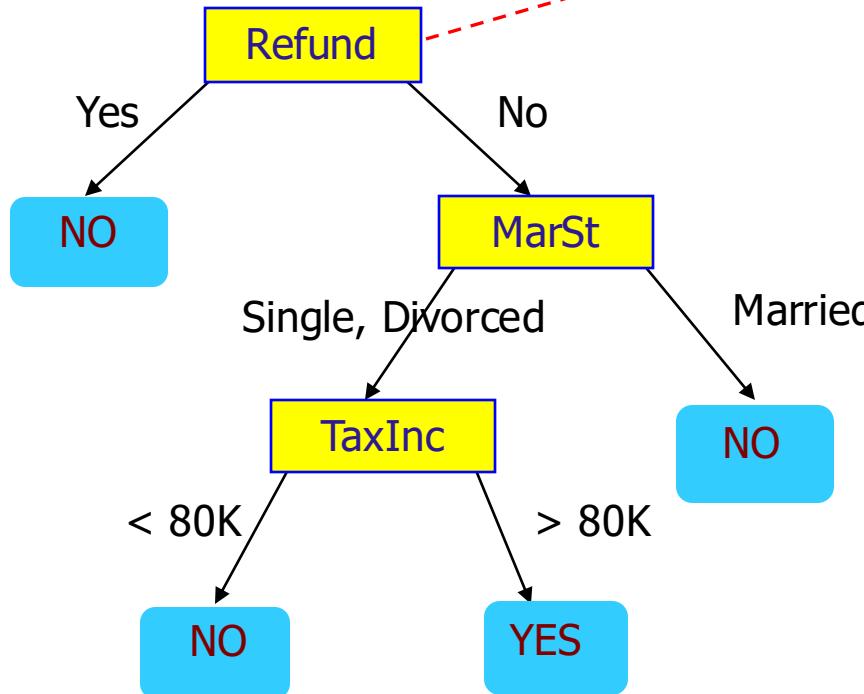
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

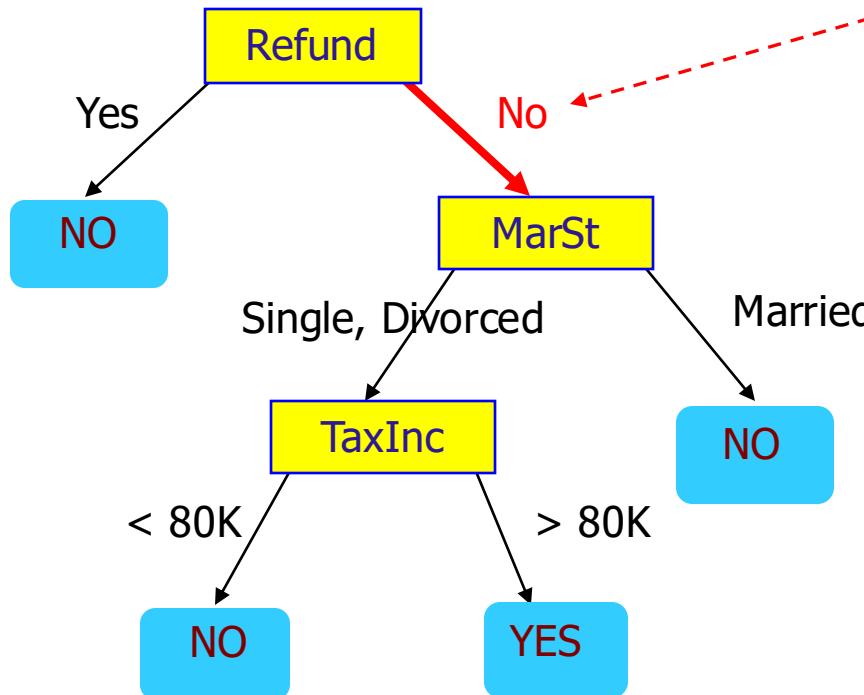




Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

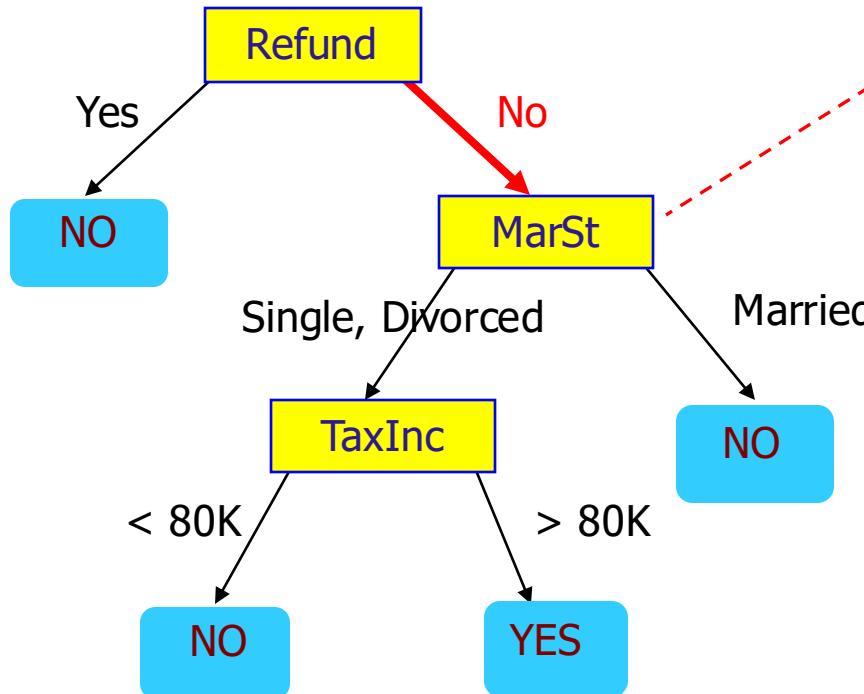




Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

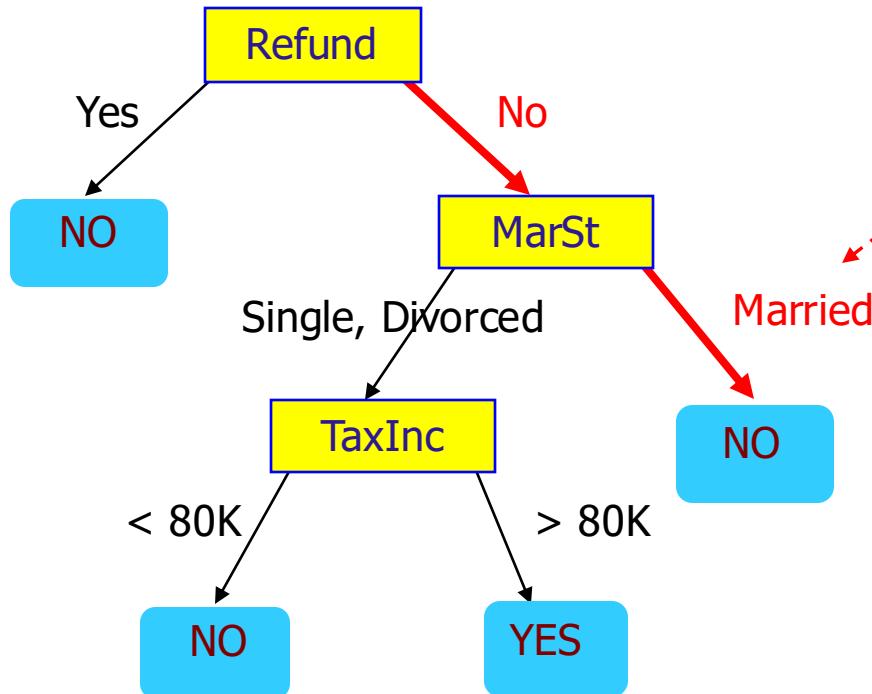




Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



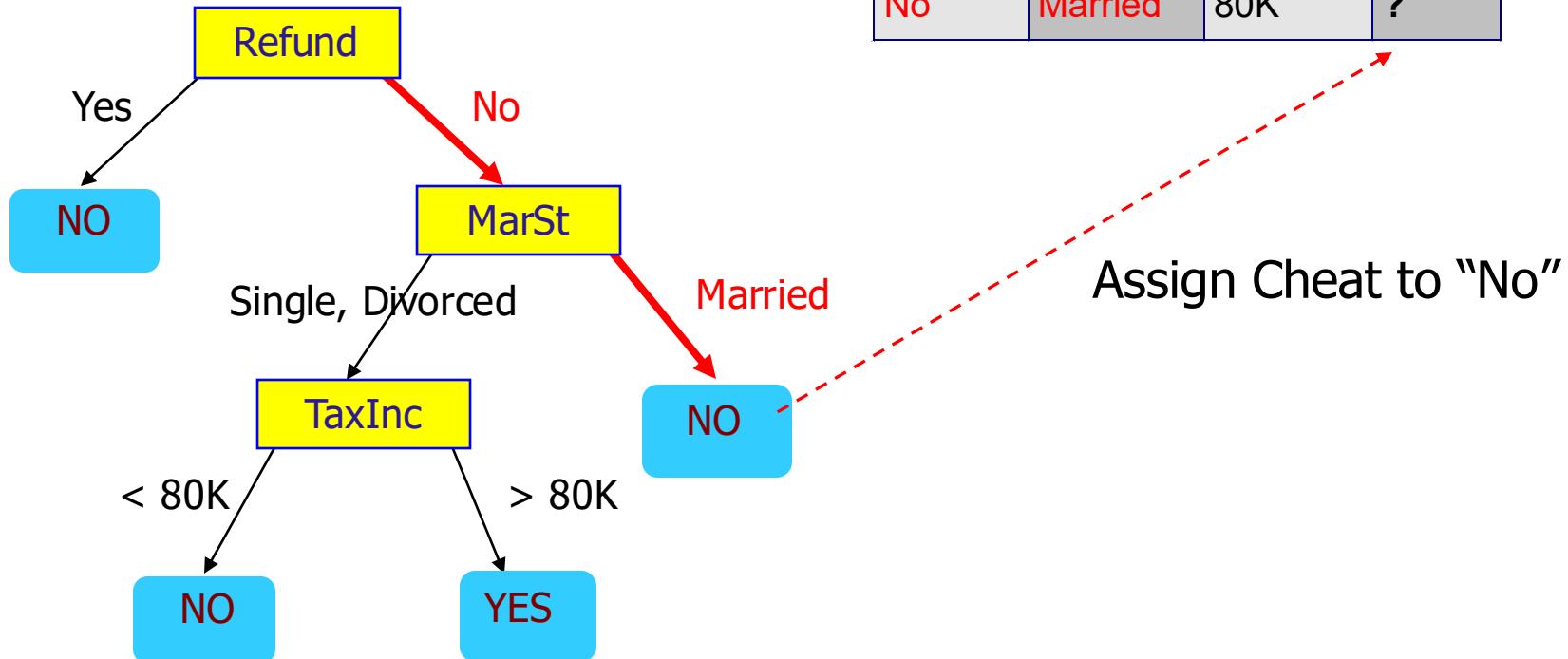
From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?





Decision tree induction

- Many algorithms to build a decision tree
 - Hunt's Algorithm (one of the earliest)
 - CART
 - ID3, C4.5, C5.0
 - SLIQ, SPRINT



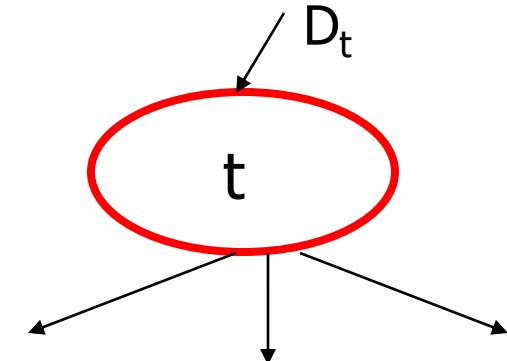
General structure of Hunt's algorithm

Basic steps

- If D_t contains records that belong to more than one class
 - select the “best” attribute A on which to split D_t and label node t as A
 - split D_t into smaller subsets and recursively apply the procedure to each subset
- If D_t contains records that belong to the same class y_t
 - then t is a leaf node labeled as y_t
- If D_t is an empty set
 - then t is a leaf node labeled as the default (majority) class, y_d

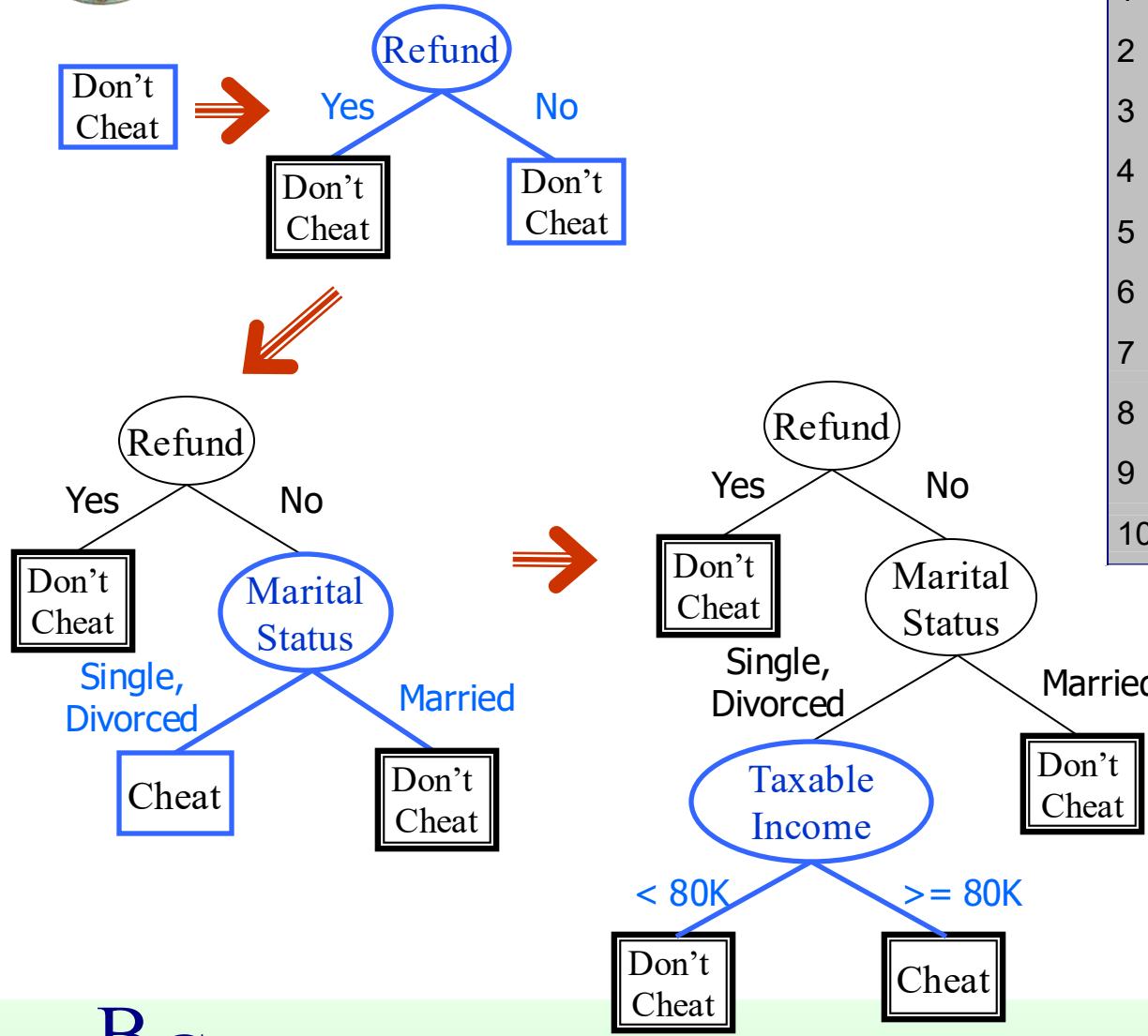
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

D_t , set of training records that reach a node t





Hunt's algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Decision tree induction

- Adopts a greedy strategy
 - “Best” attribute for the split is selected locally at each step
 - not a global optimum
- Issues
 - Structure of test condition
 - Binary split versus multiway split
 - Selection of the best attribute for the split
 - Stopping condition for the algorithm



Structure of test condition

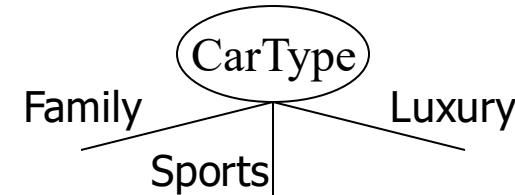
- Depends on attribute type
 - nominal
 - ordinal
 - continuous
- Depends on number of outgoing edges
 - 2-way split
 - multi-way split



Splitting on nominal attributes

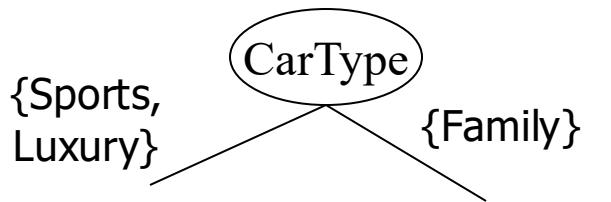
■ Multi-way split

- use as many partitions as distinct values

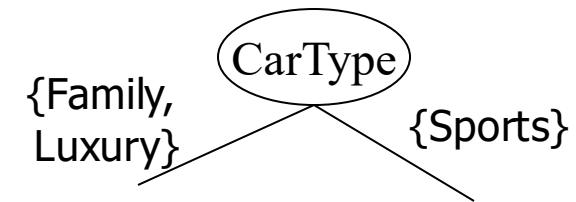


■ Binary split

- Divides values into two subsets
- Need to find optimal partitioning



OR





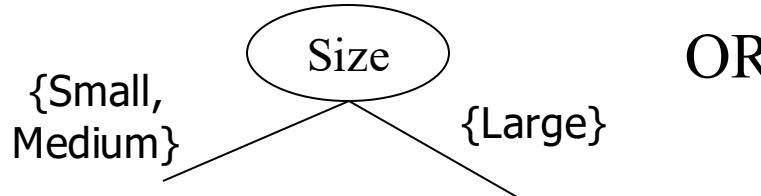
Splitting on ordinal attributes

■ Multi-way split

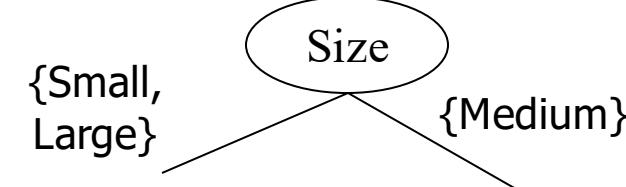
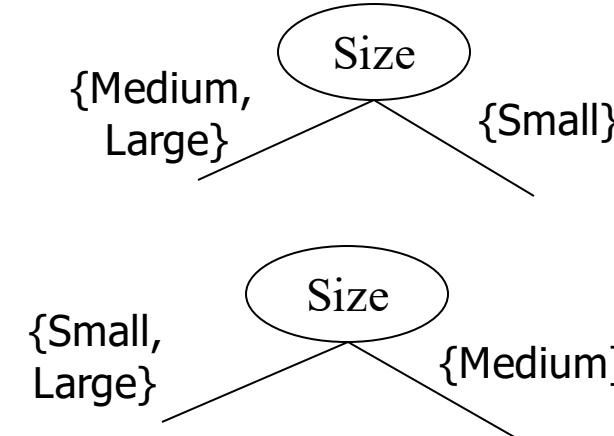
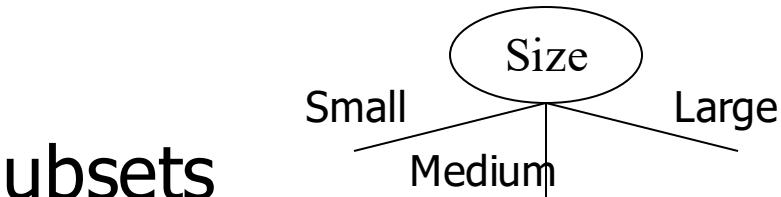
- use as many partitions as distinct values

■ Binary split

- Divides values into two subsets
- Need to find optimal partitioning



OR



What about this split?



Common approaches

- Nominal attributes
 - Encoded with 1-hot encoding
 - Each feature can be split on presence/absence
- Ordinal attributes
 - We can encode ordinal attributes with integers, to produce only “reasonable splits”
 - E.g., Small=0, Medium=1, Large=2
 - Splits only consider {Small} vs {Medium, Large}, {Small, Medium} vs {Large}



Splitting on continuous attributes

■ Different techniques

- **Discretization** to form an ordinal categorical attribute

- Static – discretize once at the beginning
 - Dynamic – discretize during tree induction

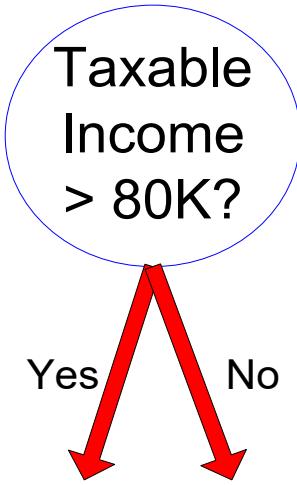
Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering

- **Binary decision** ($A < v$) or ($A \geq v$)

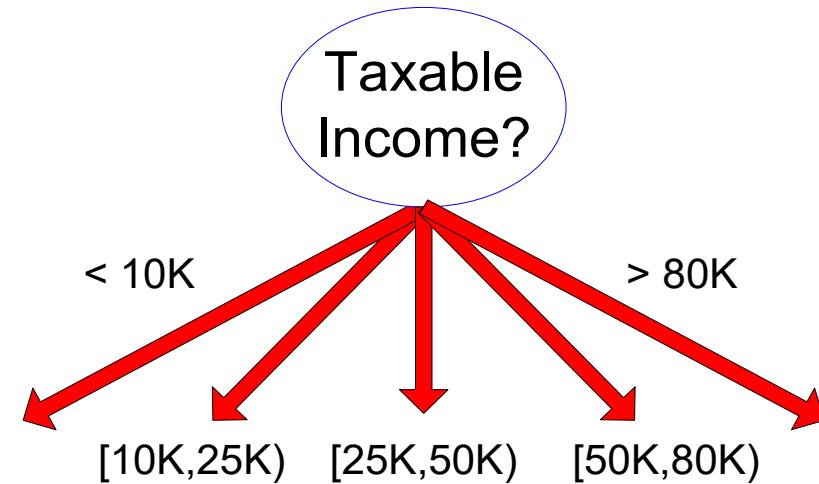
- consider all possible splits and find the best cut
 - more computationally intensive



Splitting on continuous attributes



(i) Binary split



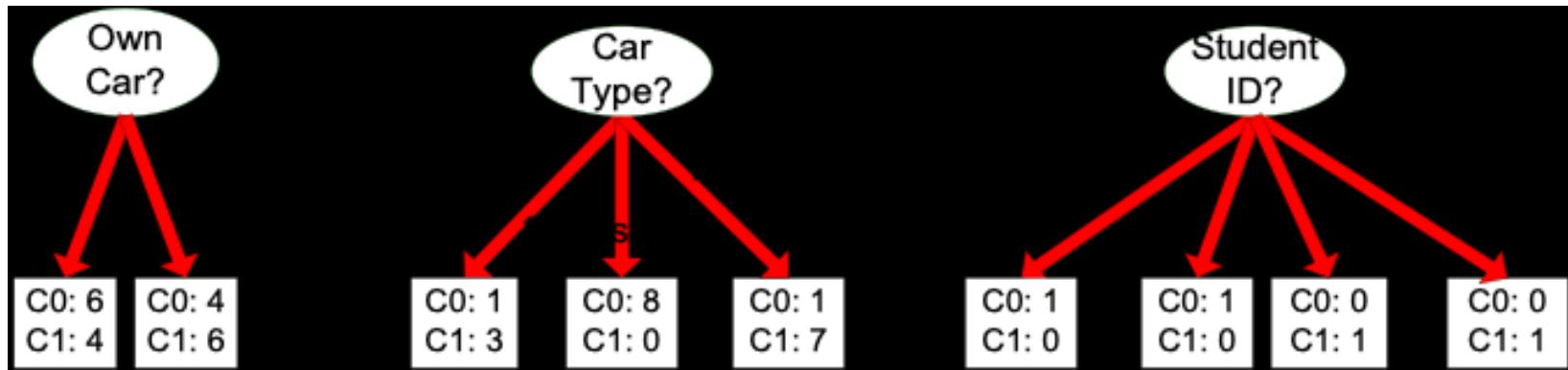
(ii) Multi-way split



Selection of the best attribute

Before splitting:

10 records of class 0,
10 records of class 1



Which attribute (test condition) is the best?



Selection of the best attribute

- Attributes with *homogeneous* class distribution are preferred
- Need a measure of node impurity

C0: 5
C1: 5

Non-homogeneous,
high degree of impurity

C0: 9
C1: 1

Homogeneous, low
degree of impurity



Measures of node impurity

- Many different measures available
 - Gini index
 - Entropy
 - Misclassification error
- Different algorithms rely on different measures

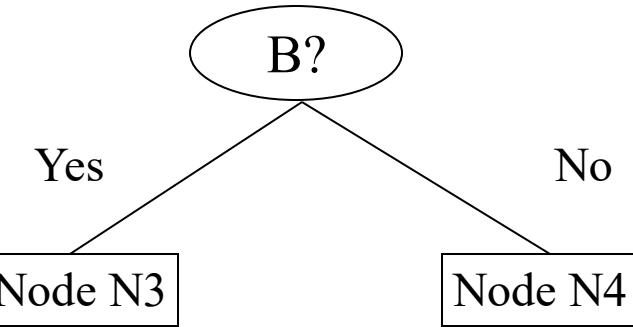
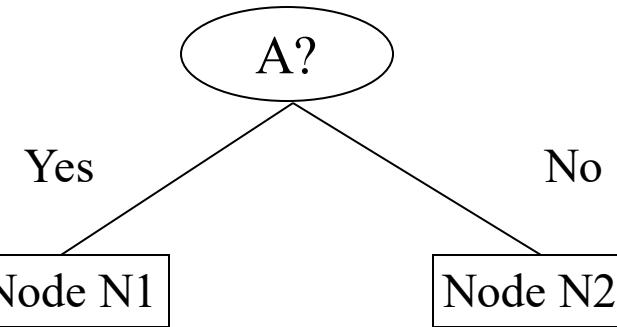


How to find the best attribute

Before Splitting:

C0	N00
C1	N01

→ M0



C0	N10
C1	N11

C0	N20
C1	N21

C0	N30
C1	N31

C0	N40
C1	N41



$$\text{Gain} = M0 - M12 \text{ vs } M0 - M34$$



GINI impurity measure

- Gini Index for a given node t

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

$p(j | t)$ is the relative frequency of class j at node t

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying higher impurity degree
- Minimum (0.0) when all records belong to one class, implying lower impurity degree

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	



Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$



Splitting based on GINI

- Used in CART, SLIQ, SPRINT
- When a node p is split into k partitions (children), the quality of the split is computed as

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where

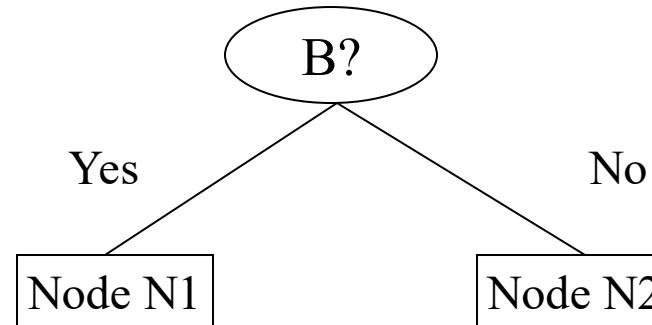
n_i = number of records at child i

n = number of records at node p



Computing GINI index: Boolean attribute

- Splits into two partitions
 - larger and purer partitions are sought for



	Parent
C1	6
C2	6
Gini = 0.500	

$\text{Gini}(N_1)$

$$\begin{aligned} &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.408 \end{aligned}$$

$\text{Gini}(N_2)$

$$\begin{aligned} &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.32 \end{aligned}$$

	N1	N2
C1	5	1
C2	2	4
Gini=?		

$\text{Gini}(\text{split on } B)$

$$\begin{aligned} &= 7/12 * 0.408 + \\ &\quad 5/12 * 0.32 \\ &= 0.371 \end{aligned}$$



Computing GINI index: Categorical attribute

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

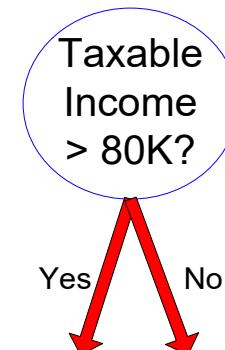
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	



Computing GINI index: Continuous attribute

- Binary decision on one splitting value
 - Number of possible splitting values
= Number of distinct values
- Each splitting value v has a count matrix
 - class counts in the two partitions
 - $A < v$
 - $A \geq v$

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





Computing GINI index: Continuous attribute

- For each attribute
 - Sort the attribute on values
 - Linearly scan these values, each time updating the count matrix and computing gini index
 - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No	No		
Sorted Values Split Positions	Taxable Income												
	60	70	75	85	90	95	100	120	125	172	220		
	55	65	72	80	87	92	97	110	122	172	230		
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	
	Yes	0	3	0	3	0	3	0	3	0	3	0	
	No	0	7	1	6	2	5	3	4	3	4	7	0
	Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420	



Entropy impurity measure (INFO)

- Entropy at a given node t

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

$p(j | t)$ is the relative frequency of class j at node t

- Maximum ($\log n_c$) when records are equally distributed among all classes, implying higher impurity degree
- Minimum (0.0) when all records belong to one class, implying lower impurity degree
- Entropy based computations are similar to GINI index computations



Examples for computing entropy

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Entropy} = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$



Splitting Based on INFO

- Information Gain

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
 n_i is number of records in partition i

- Measures reduction in entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits yielding a large number of partitions, each small but pure



Splitting Based on INFO

- Gain Ratio

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^k \frac{n_i}{n} \log \frac{n_i}{n}$$

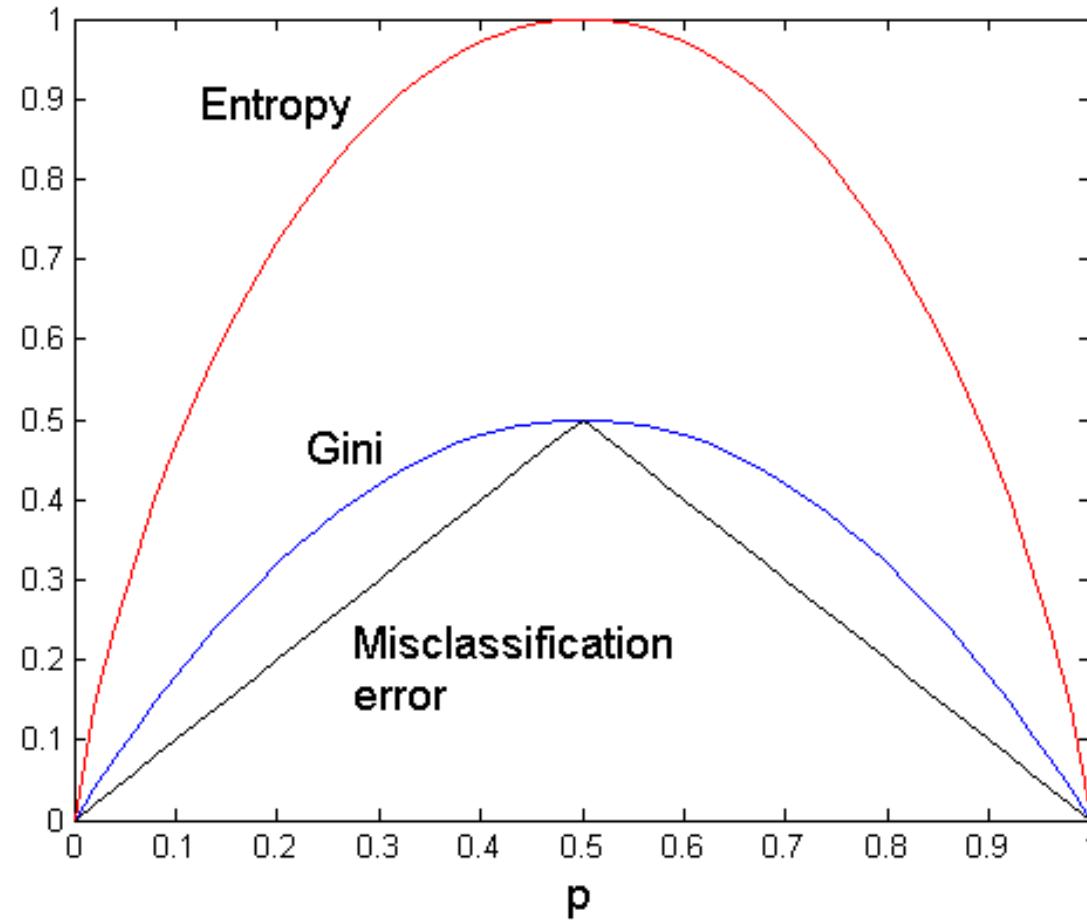
Parent Node, p is split into k partitions
 n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain



Comparison among splitting criteria

For a 2-class problem



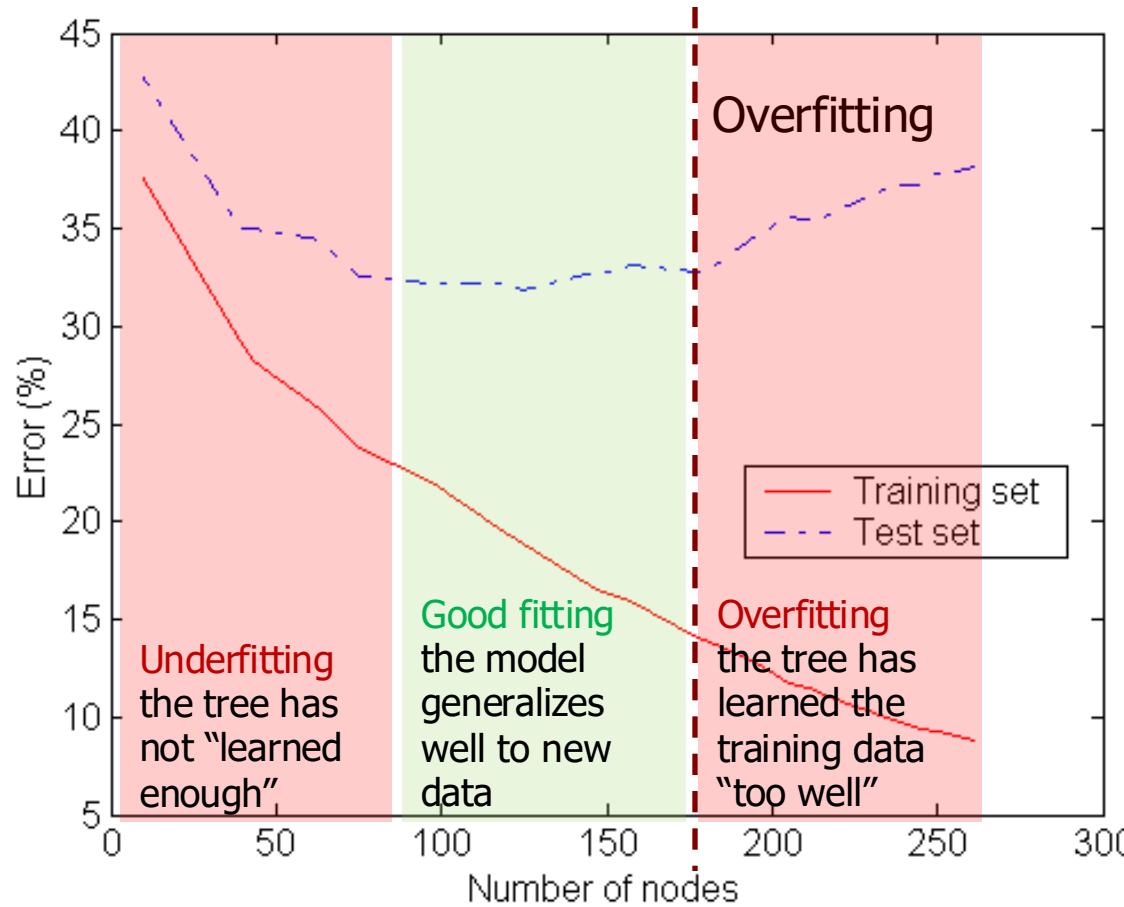


Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination
 - Pre-pruning
 - Post-pruning



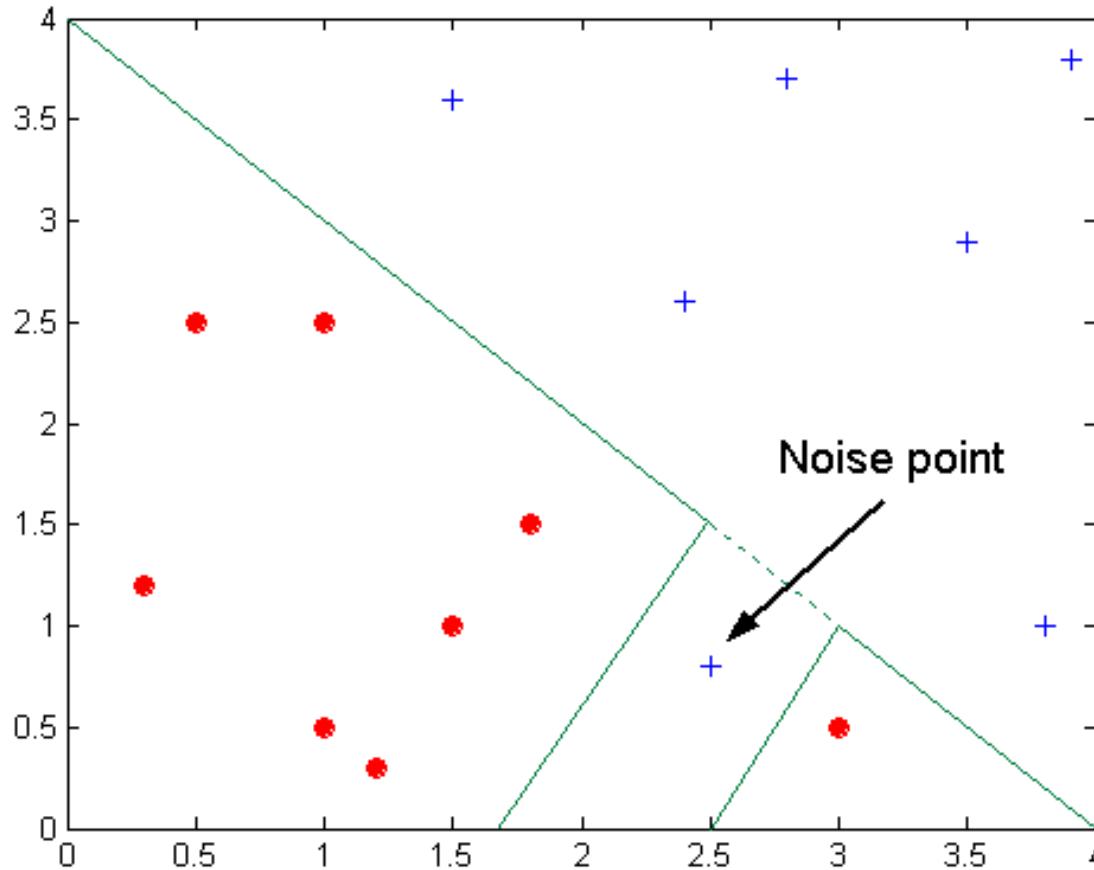
Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large



Overfitting due to Noise



Decision boundary is distorted by noise point



How to address overfitting

- Pre-Pruning (Early Stopping Rule)
 - Stop the algorithm before it becomes a fully-grown tree
 - Typical stopping conditions for a node
 - Stop if all instances belong to the same class
 - Stop if all the attribute values are the same
 - More restrictive conditions
 - Stop if number of instances is less than some user-specified threshold
 - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
 - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain)



How to address overfitting

■ Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the sub-tree



Data fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

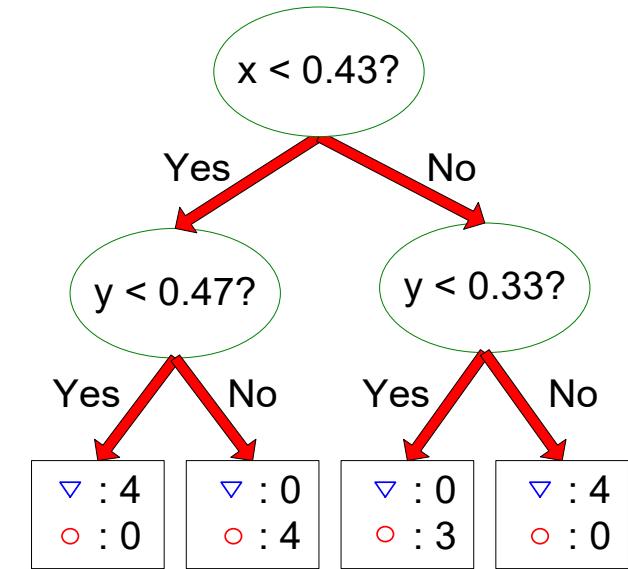
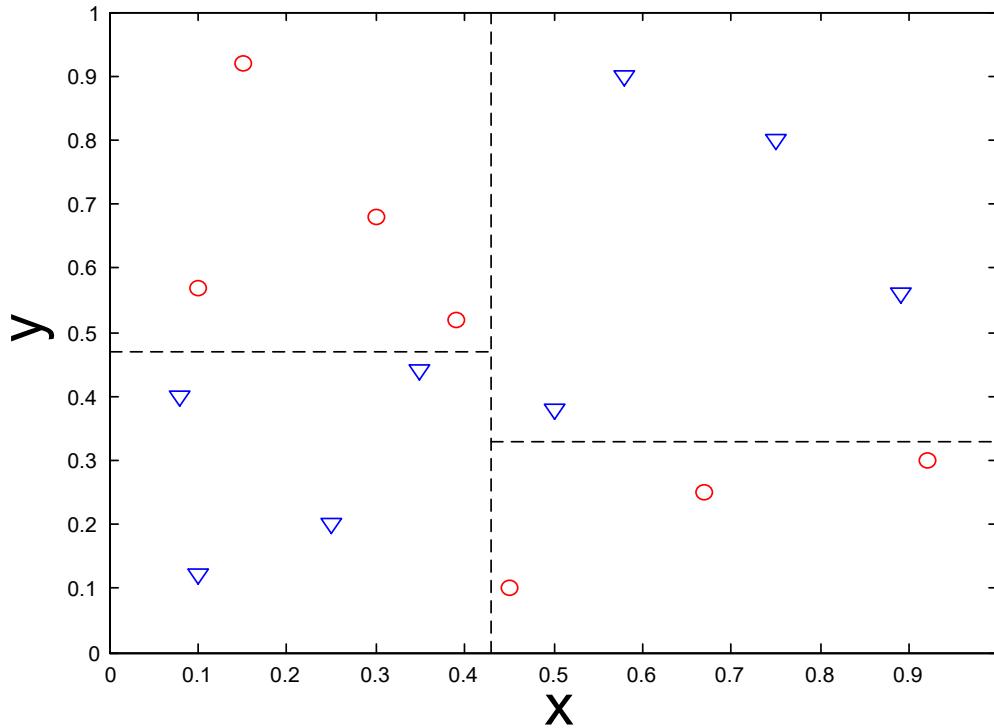


Handling missing attribute values

- Missing values affect decision tree construction in three different ways
 - Affect how impurity measures are computed
 - Affect how to distribute instances with missing value to child nodes
 - Affect how a test instance with missing value is classified



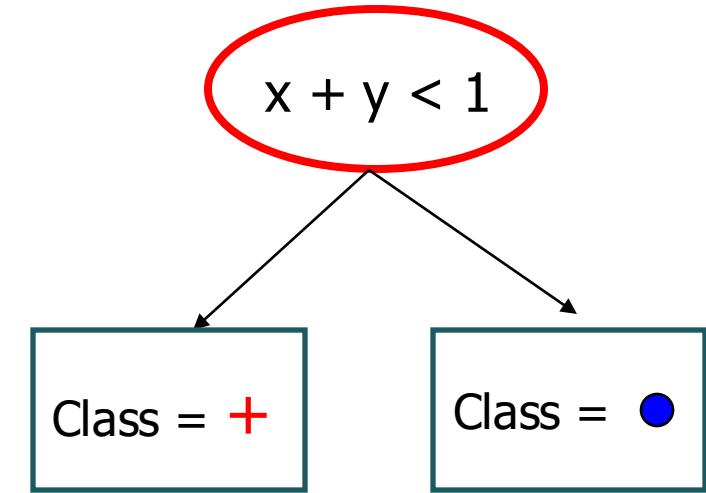
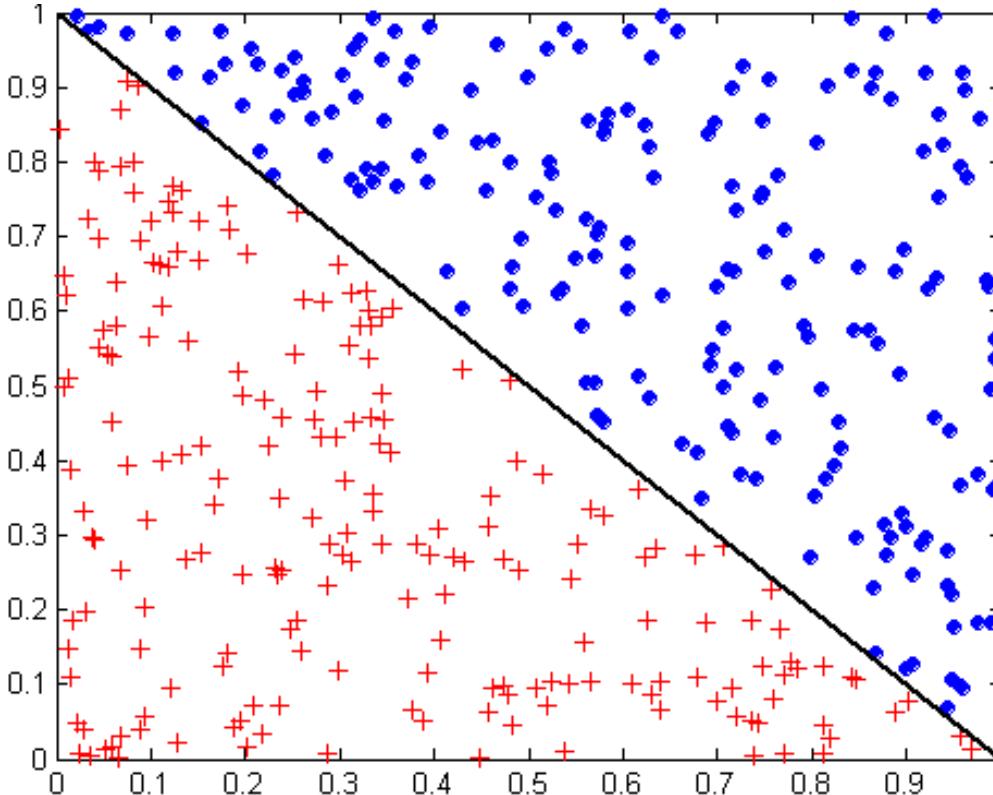
Decision boundary



- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time



Oblique decision trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive



Evaluation of decision trees

- Accuracy
 - For simple datasets, comparable to other classification techniques
- Interpretability
 - Model is interpretable for small trees
 - Single predictions are interpretable
- Incrementality
 - Not incremental
- Efficiency
 - Fast model building
 - Very fast classification
- Scalability
 - Scalable both in training set size and attribute number
- Robustness
 - Difficult management of missing data

Random Forest



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis
Politecnico di Torino



Random Forest

- Ensemble learning technique
 - multiple base models are combined
 - to improve accuracy and stability
 - to avoid overfitting
- Random forest = set of decision trees
 - a number of decision trees are built at training time
 - the class is assigned by majority voting



Random Forest

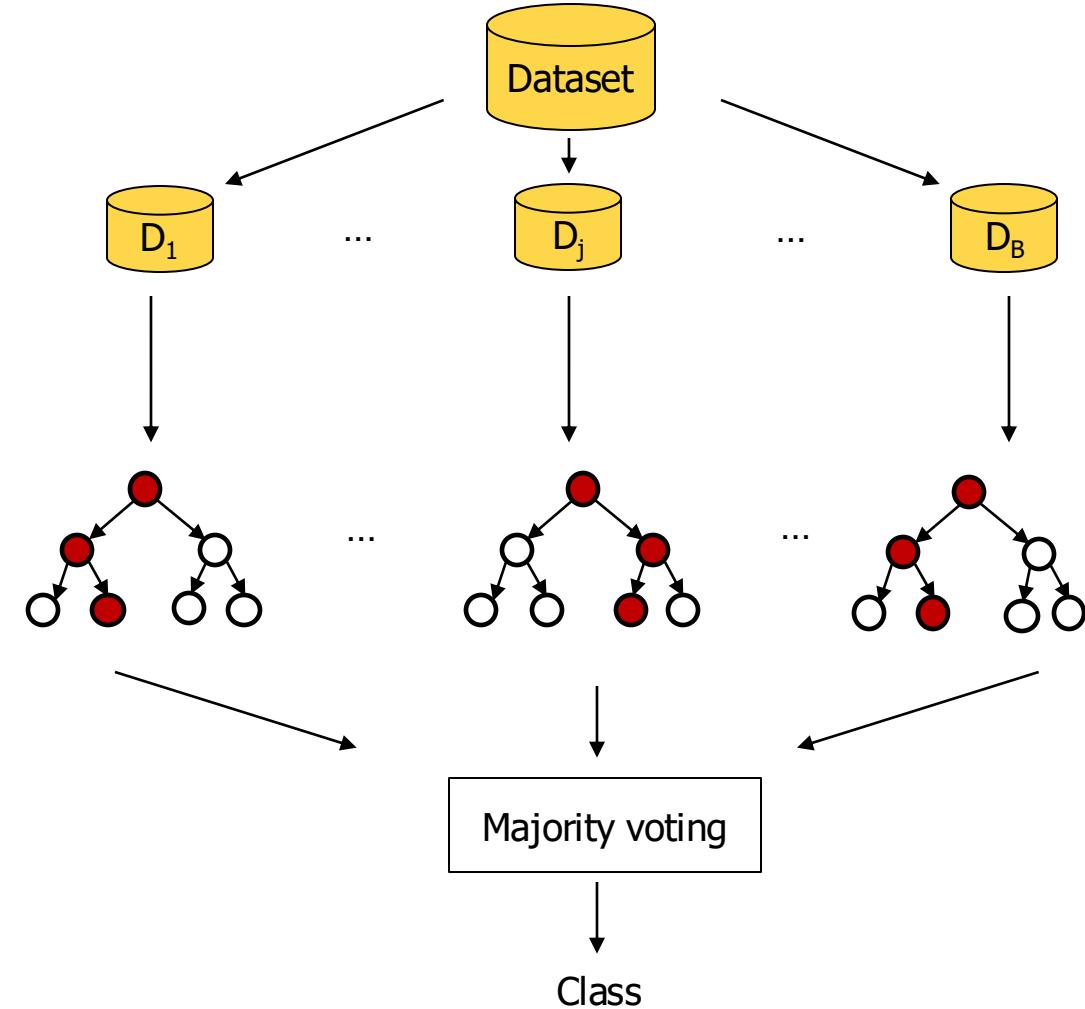
Original Training data

Random subsets

Multiple decision trees

For each subset, a tree is learned on a *random* set of features

Aggregating classifiers





Bootstrap aggregation

- Given a training set D of n instances, it selects B times a *random* sample with replacement from D and trains trees on these dataset samples
 - For $b = 1, \dots, B$
 - Sample with replacement n' training examples, $n' \leq n$
 - A dataset subset D_b is generated
 - Train a classification tree on D_b



Feature bagging

- Selects, *for each candidate split* in the learning process, a *random* subset of the features
 - Being p the number of features, \sqrt{p} features are typically selected
- Trees are decorrelated
 - Feature subsets are sampled randomly, hence different features can be selected as best attributes for the split



Random Forest – Algorithm Recap

- Given a training set D of n instances with p features
- For $b = 1, \dots, B$
 - Sample randomly with replacement n' training examples. A subset D_b is generated
 - Train a classification tree on D_b
 - During the tree construction, for each candidate split
 - $m \ll p$ random features are selected (typically $m \approx \sqrt{p}$)
 - the best split is computed among these m features
- Class is assigned by majority voting among the B predictions



Evaluation of random forests

- Accuracy
 - Higher than decision trees
- Interpretability
 - Model and prediction are not interpretable
 - A prediction may be given by hundreds of trees
 - Provide global feature importance
 - an estimate of which features are important in the classification
- Incrementality
 - Not incremental
- Efficiency
 - Fast model building
 - Very fast classification
- Scalability
 - Scalable both in training set size and attribute number
- Robustness
 - Robust to noise and outliers

K-Nearest Neighbor



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis
Politecnico di Torino



Instance-Based Classifiers

Set of Stored Cases

Atr1	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

Atr1	AtrN



Instance Based Classifiers

■ Examples

- Rote-learner

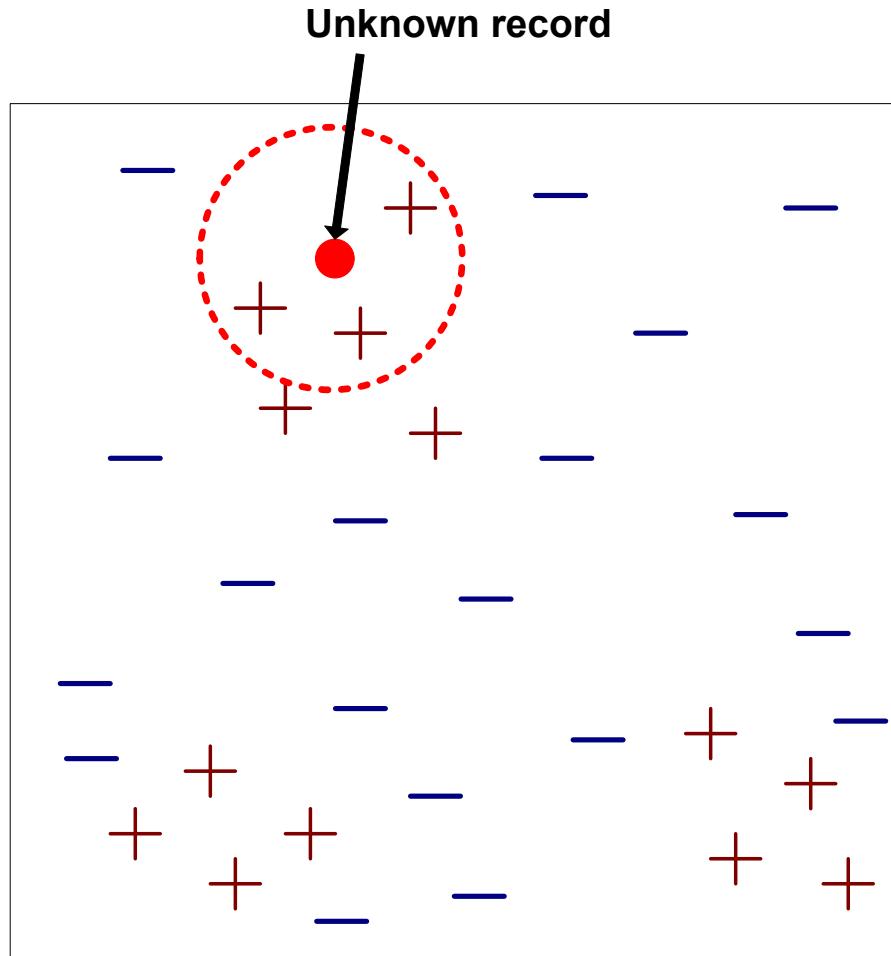
- Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

- Nearest neighbor

- Uses k “closest” points (nearest neighbors) for performing classification



Nearest-Neighbor Classifiers

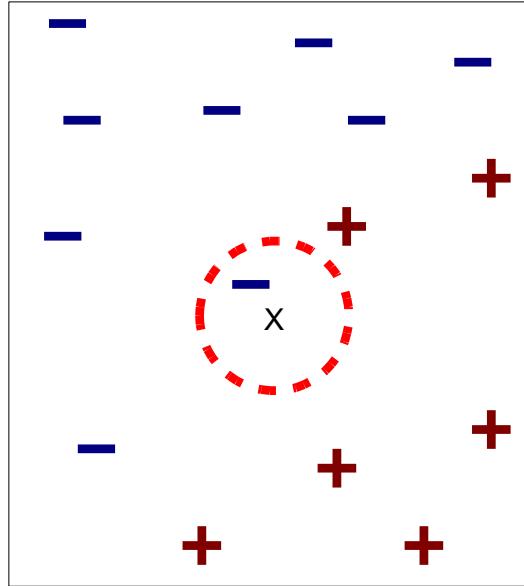


- Requires
 - The set of stored records
 - Distance Metric to compute distance between records
 - The value of k , the number of nearest neighbors to retrieve

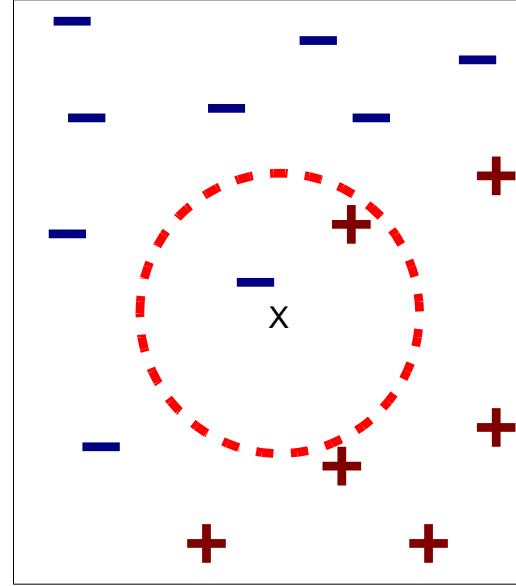
- To classify an unknown record
 - Compute distance to other training records
 - Identify k nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



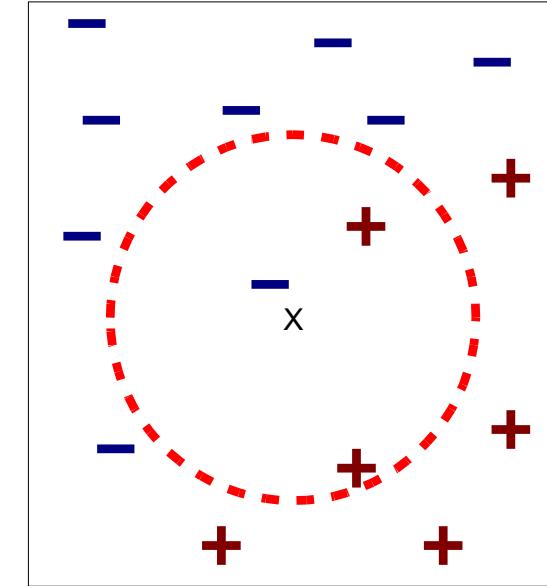
Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor



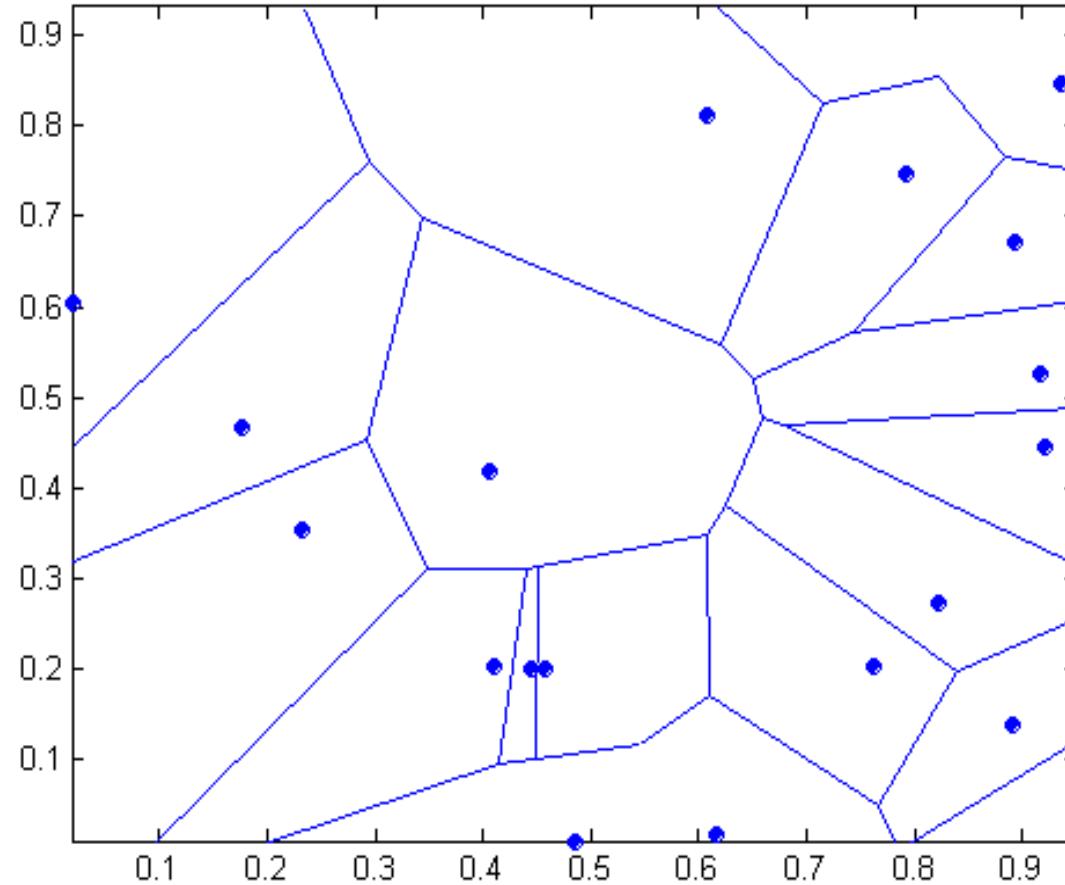
(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x



1 nearest-neighbor

Voronoi Diagram





Nearest Neighbor Classification

- Compute distance between two points
 - Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

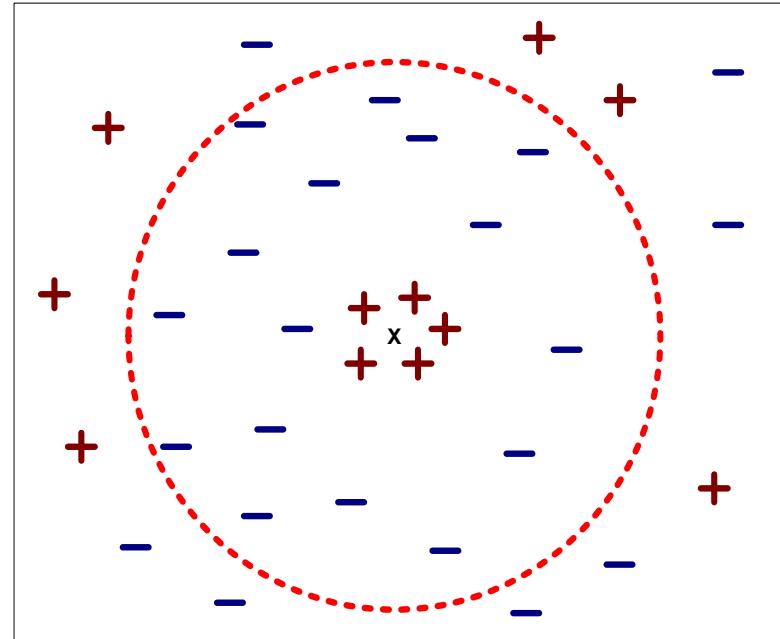
- Determine the class from nearest neighbor list
 - take the majority vote of class labels among the k-nearest neighbors
 - Weigh the vote according to distance
 - weight factor, $w = 1/d^2$



Nearest Neighbor Classification

■ Choosing the value of k:

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes





Nearest Neighbor Classification

■ Scaling issues

- Attribute domain should be normalized to prevent distance measures from being dominated by one of the attributes
- Example: height [1.5m to 2.0m] vs. income [\$10K to \$1M]

■ Problem with distance measures

- High dimensional data
 - curse of dimensionality



Evaluation of KNN

- Accuracy
 - Comparable to other classification techniques for simple datasets
- Interpretability
 - Model is not interpretable
 - Single predictions can be "described" by neighbors
- Incrementality
 - Incremental
 - Training set *must* be available
- Efficiency
 - (Almost) no model building
 - Slower classification, requires computing distances
- Scalability
 - Weakly scalable in training set size
 - Curse of dimensionality for increasing attribute number
- Robustness
 - Depends on distance computation

Bayesian Classification



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis
Politecnico di Torino



Bayes theorem

- Let C and X be random variables

$$P(C,X) = P(C|X) P(X)$$

$$P(C,X) = P(X|C) P(C)$$

- Hence

$$P(C|X) P(X) = P(X|C) P(C)$$

- and also

$$P(C|X) = P(X|C) P(C) / P(X)$$



Bayesian classification

- Let the class attribute and all data attributes be random variables
 - $C = \text{any class label}$
 - $X = \langle x_1, \dots, x_k \rangle$ record to be classified
- Bayesian classification
 - compute $P(C|X)$ for all classes
 - probability that record X belongs to C
 - assign X to the class with *maximal* $P(C|X)$
- Applying Bayes theorem
$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$
 - $P(X)$ constant for all C , disregarded for maximum computation
 - $P(C)$ a priori probability of C

$$P(C) = N_c/N$$



Bayesian classification

- How to estimate $P(X|C)$, i.e. $P(x_1, \dots, x_k|C)$?
- Naïve hypothesis

$$P(x_1, \dots, x_k|C) = P(x_1|C) P(x_2|C) \dots P(x_k|C)$$

- *statistical independence* of attributes x_1, \dots, x_k
- not always true
 - model quality may be affected
- Computing $P(x_k|C)$
 - for discrete attributes
$$P(x_k|C) = |x_{kC}| / N_C$$
 - where $|x_{kC}|$ is number of instances having value x_k for attribute k and belonging to class C
 - for continuous attributes, use probability distribution
- Bayesian networks
 - allow specifying a subset of dependencies among attributes



Bayesian classification: Example

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Bayesian classification: Example

outlook		
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$	$P(p) = 9/14$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$	$P(n) = 5/14$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$	
temperature		
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$	
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$	
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$	
humidity		
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$	
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$	
windy		
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$	
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$	



Bayesian classification: Example

- Data to be labeled

$$X = \langle \text{rain}, \text{hot}, \text{high}, \text{false} \rangle$$

- For class p

$$P(X|p) \cdot P(p) =$$

$$= P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p)$$

$$= 3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$$

- For class n

$$P(X|n) \cdot P(n) =$$

$$= P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n)$$

$$= 2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = \textcolor{red}{0.018286}$$



Evaluation of Naïve Bayes Classifiers

- Accuracy
 - Similar or lower than decision trees
 - Naïve hypothesis simplifies model
- Interpretability
 - Model and prediction are not interpretable
 - The weights of contributions in a single prediction may be used to explain
- Incrementality
 - Fully incremental
 - Does *not* require availability of training data
- Efficiency
 - Fast model building
 - Very fast classification
- Scalability
 - Scalable both in training set size and attribute number
- Robustness
 - Affected by attribute correlation

Support Vector Machines

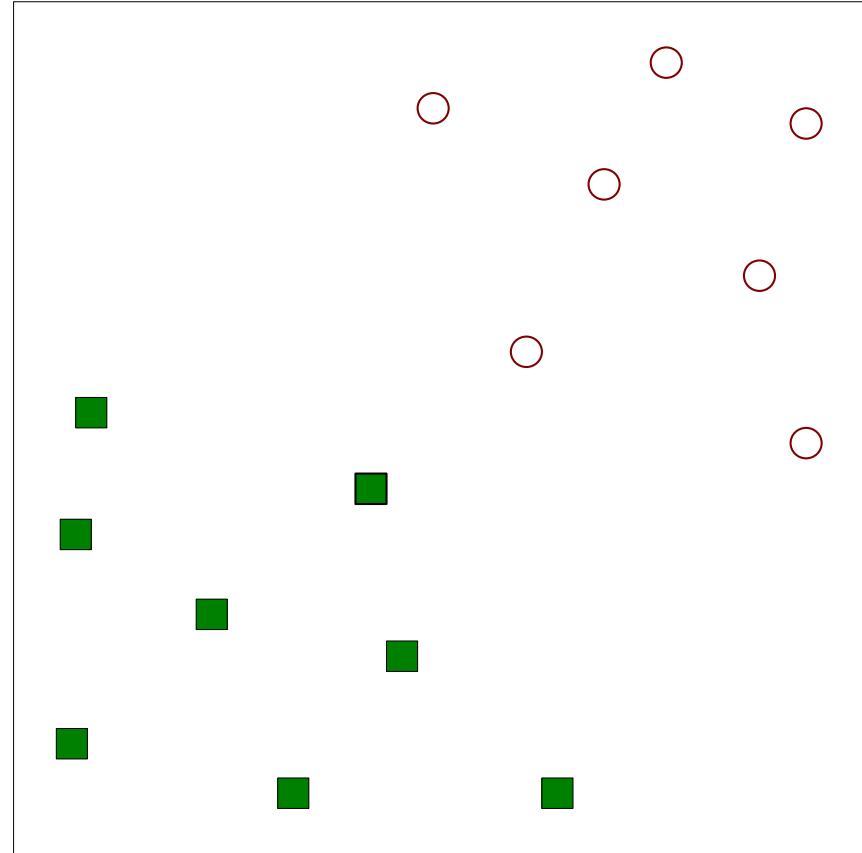


Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis
Politecnico di Torino



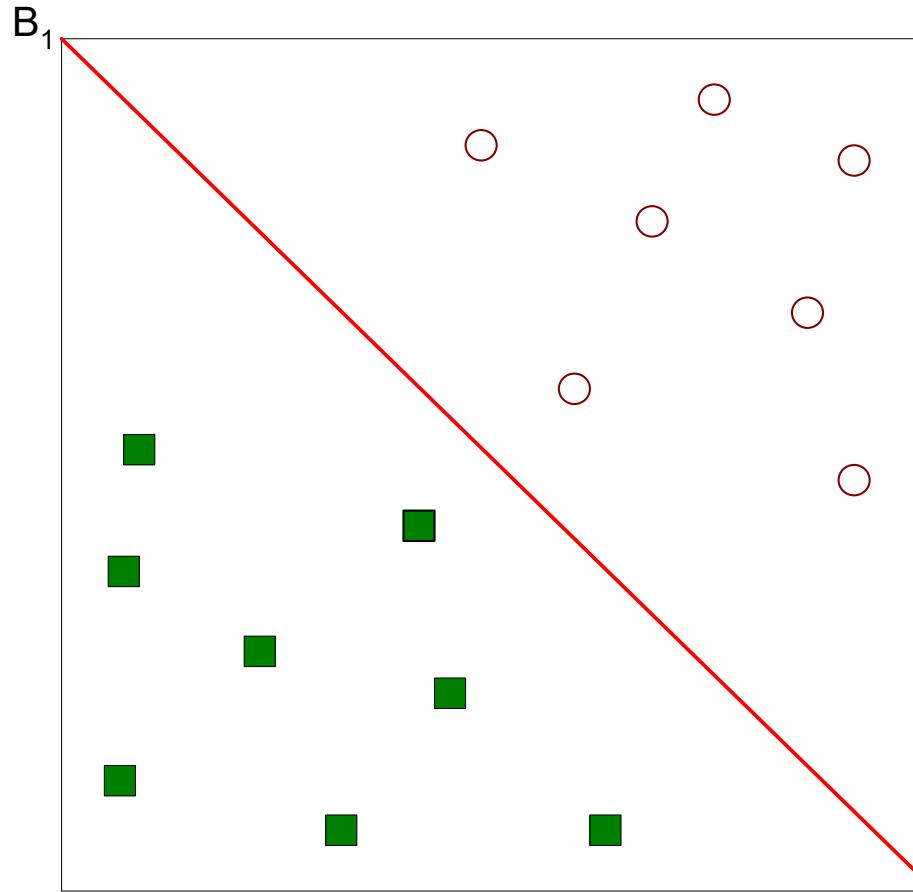
Support Vector Machines



- Find a linear hyperplane (decision boundary) that will separate the data



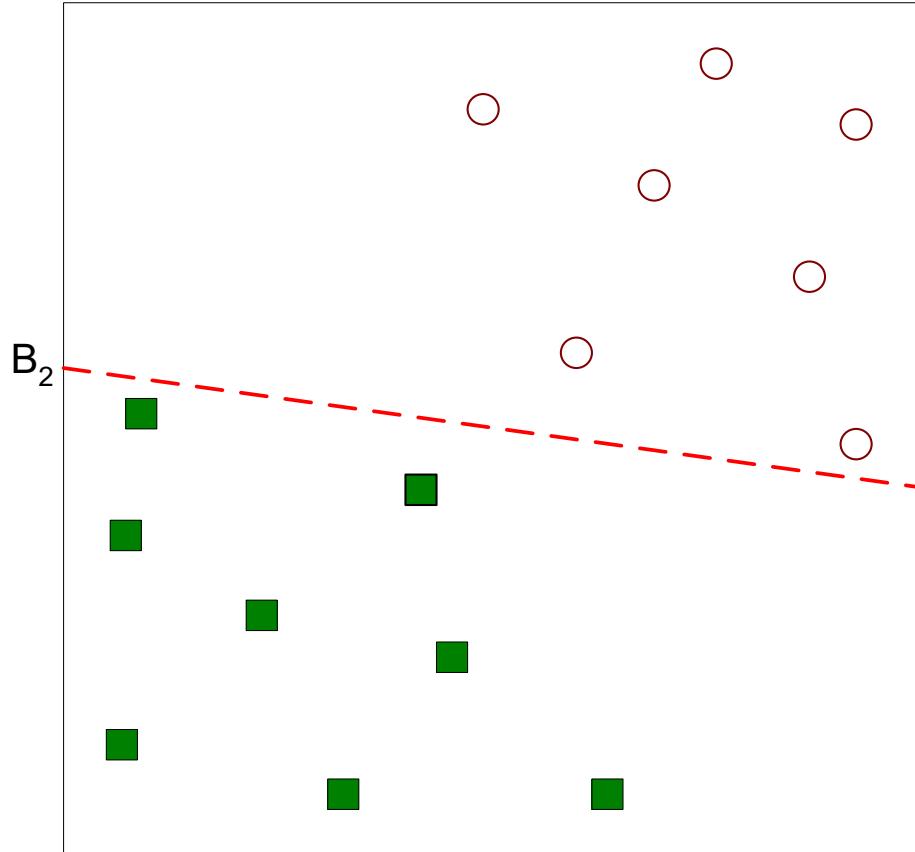
Support Vector Machines



- One Possible Solution



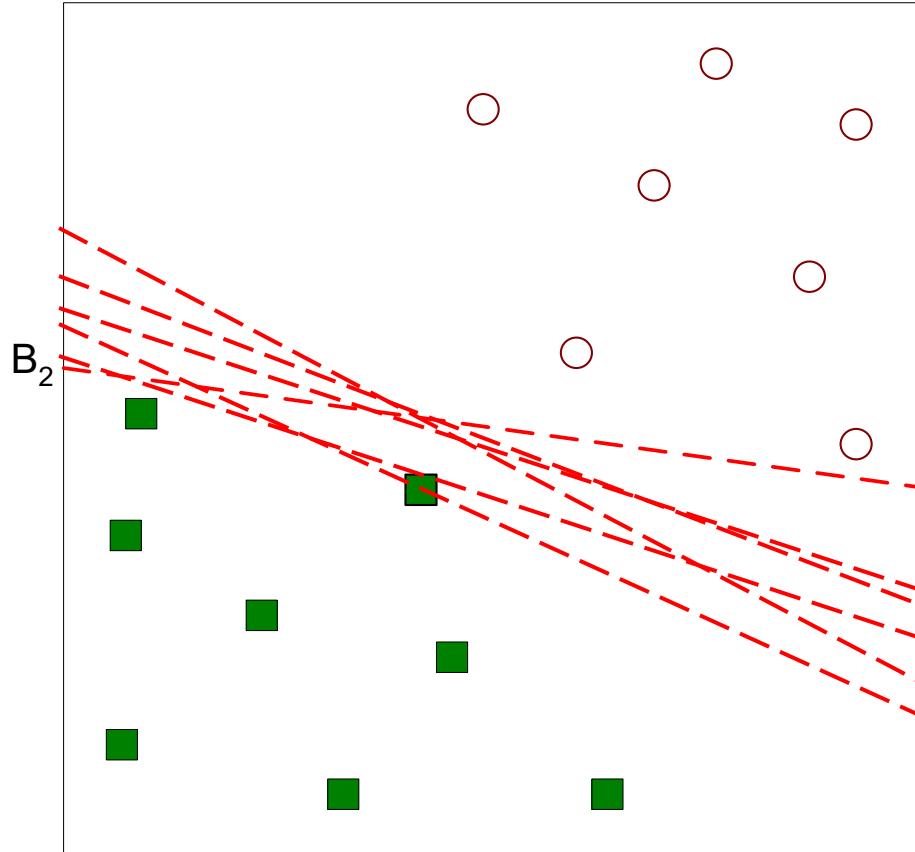
Support Vector Machines



- Another possible solution



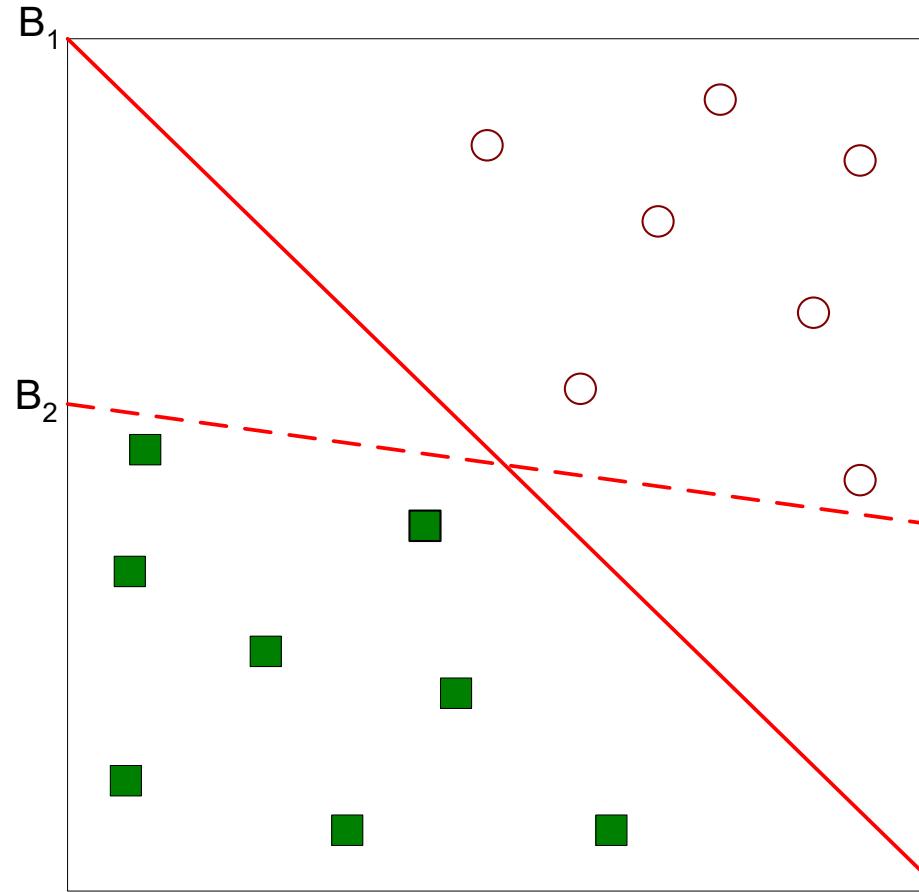
Support Vector Machines



- Other possible solutions



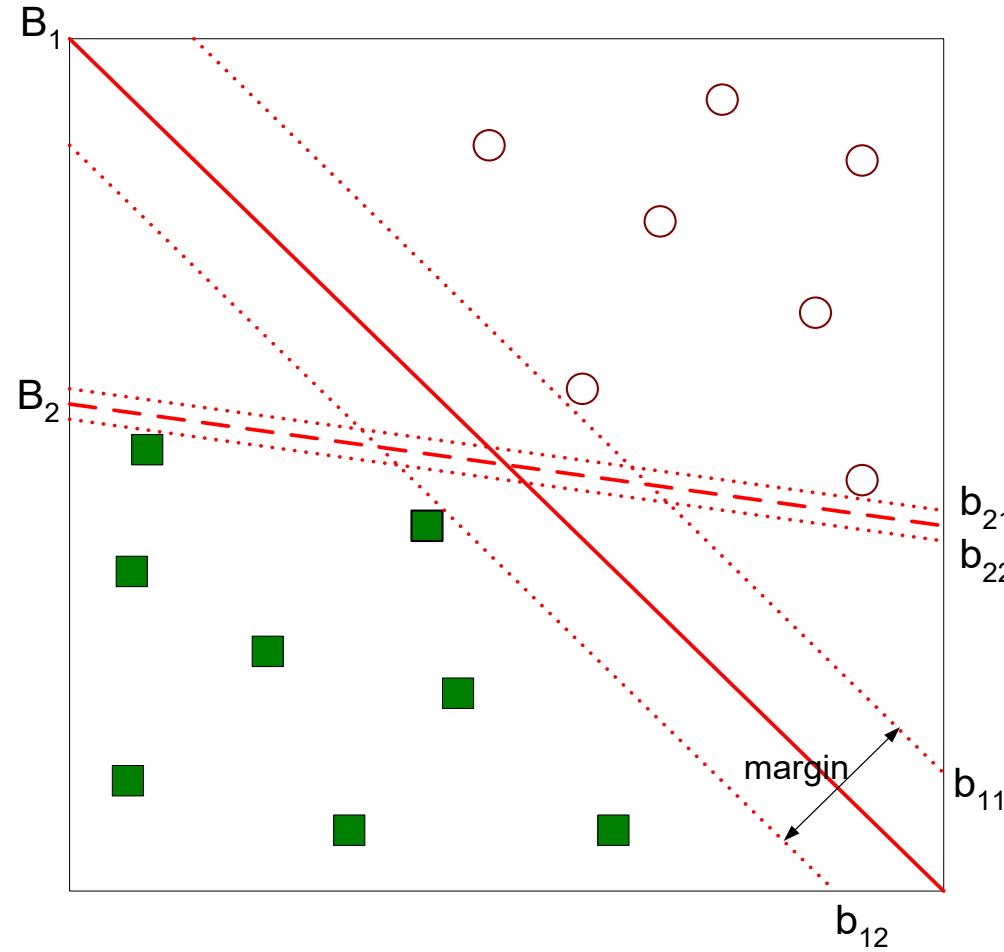
Support Vector Machines



- Which one is better? B1 or B2?
- How do you define better?



Support Vector Machines

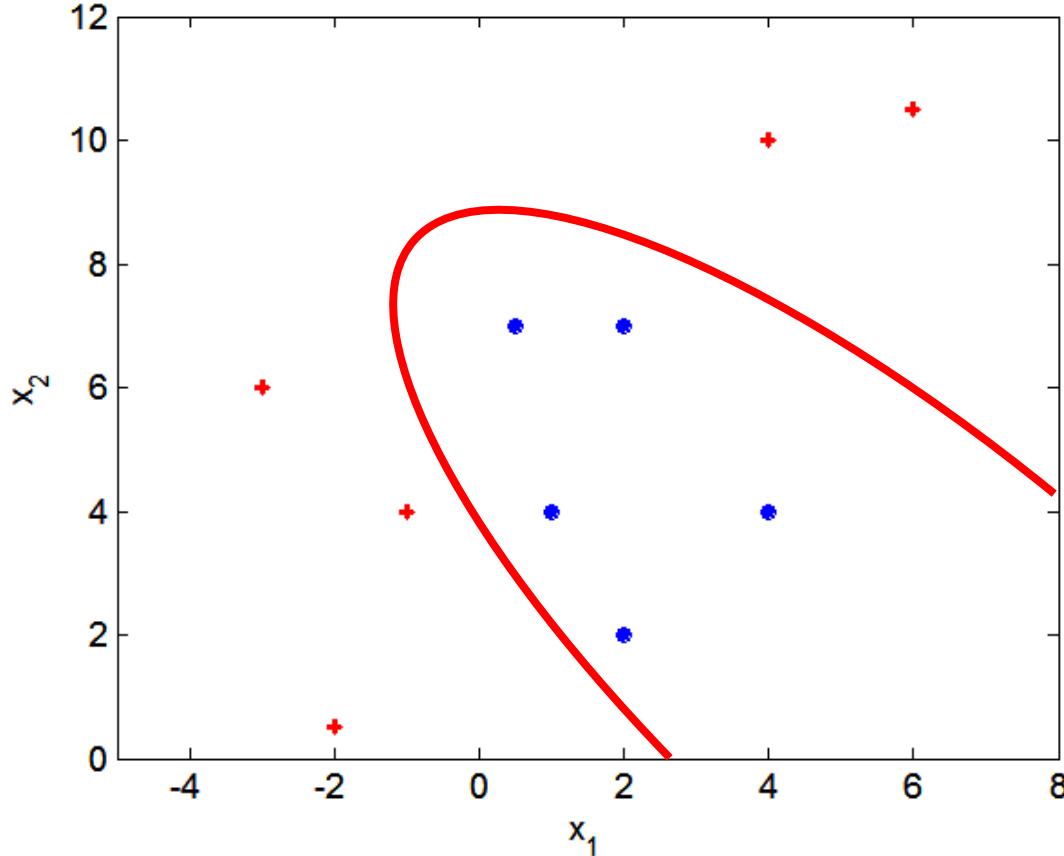


- Find hyperplane **maximizes** the margin => B1 is better than B2



Nonlinear Support Vector Machines

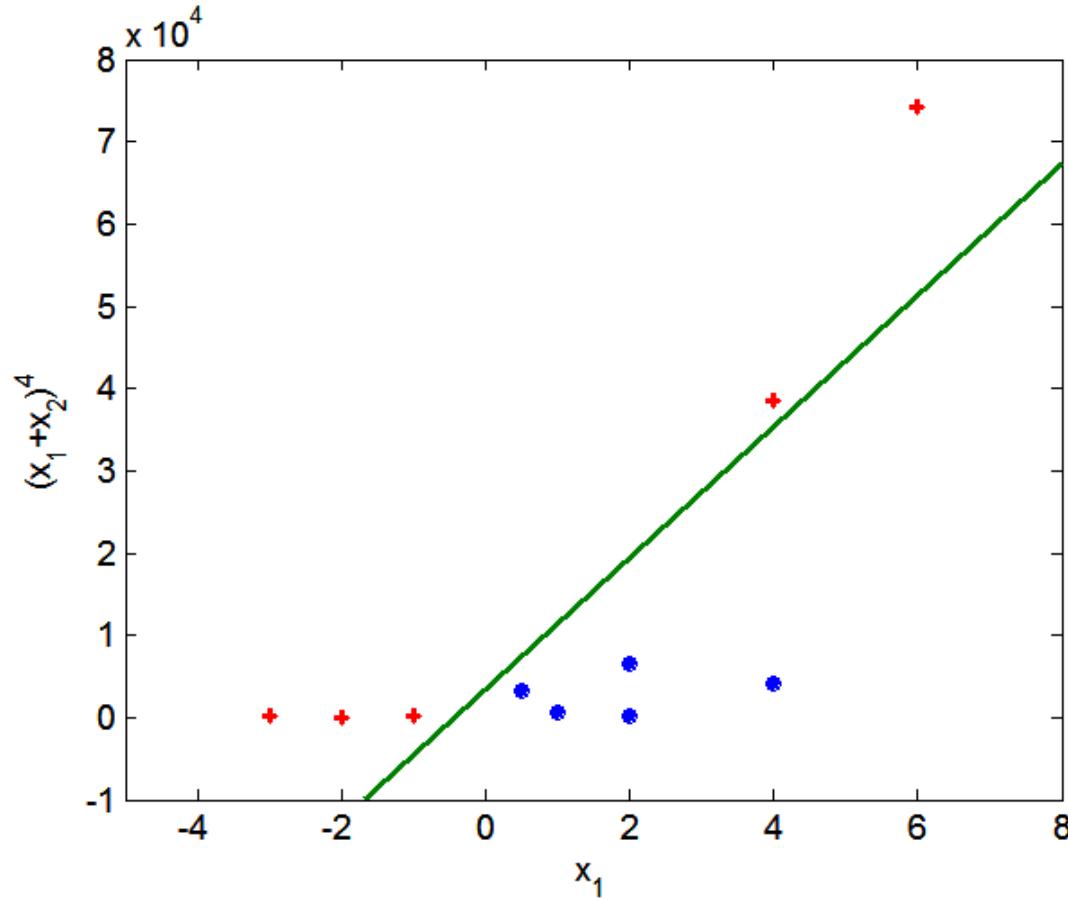
- What if decision boundary is not linear?





Nonlinear Support Vector Machines

- Transform data into higher dimensional space





Evaluation of Support Vector Machines

- Accuracy
 - Among best performers
- Interpretability
 - Model and prediction are not interpretable
 - Black box model
- Incrementality
 - Not incremental
- Efficiency
 - Model building requires significant parameter tuning
 - Very fast classification
- Scalability
 - Scalable for # of attributes (linear), not in training set size (superlinear)
- Robustness
 - Robust to noise and outliers

Model evaluation



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Flavio Giobergia
Politecnico di Torino



Model evaluation

- Methods for performance evaluation
 - Partitioning techniques for training and test sets
- Metrics for performance evaluation
 - Accuracy, other measures
- Techniques for model comparison
 - ROC curve



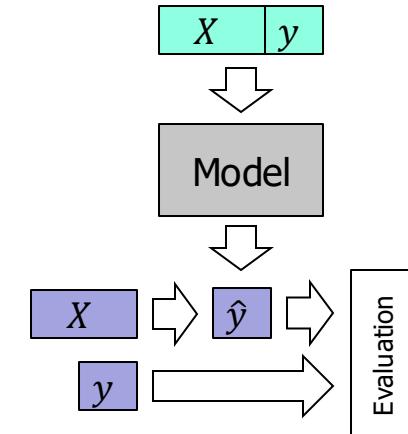
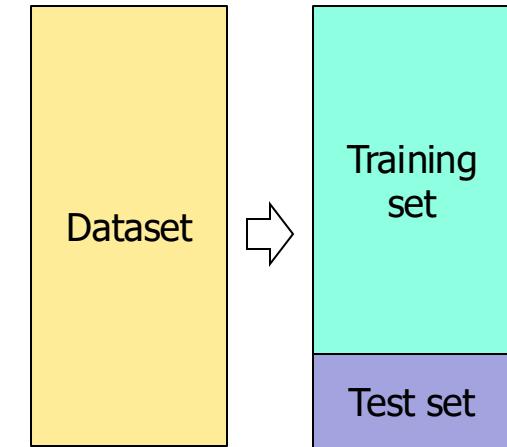
Partitioning

- To assess the quality of a classifier, we need to *test* the model on *unseen* data
- Testing the model on training data does not provide a meaningful result
 - Models generally work very well on training data, less so on test data!
- We set aside some data for the *testing*, and use the rest of the *training*
 - Depending on the cases, we use either *hold out sets*, or *k-fold cross-validation*



Hold-out

- Fixed partitioning
 - For instance, 80% of data for **training**, 20% for **testing**
 - Other proportions may be appropriate, depending on the dataset size
- Appropriate for large datasets
 - may be repeated several times
 - repeated holdout



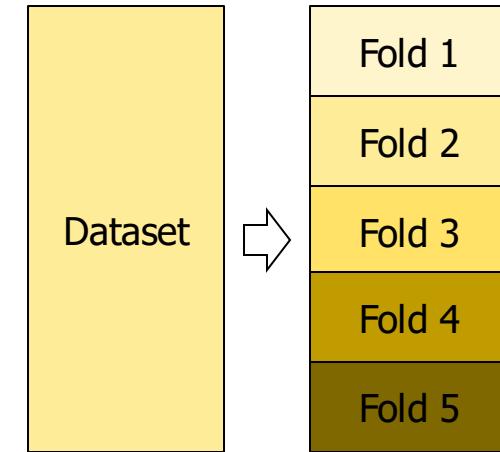


Cross validation

■ Cross validation

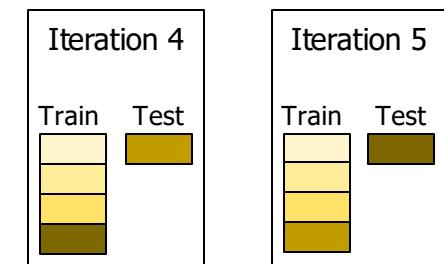
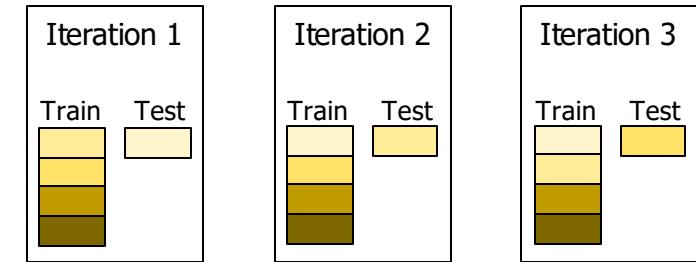
- partition data into k disjoint subsets (i.e., folds)
- k -fold: train on $k-1$ partitions, test on the remaining one
 - repeat for all folds
- reliable accuracy estimation, not appropriate for very large datasets

k -fold, $k = 5$



■ Leave-one-out

- cross validation for $k=n$
- only appropriate for very small datasets





Model performance estimation

- Model training step
 - Building a new model
- Model validation step
 - Hyperparameter tuning
 - Algorithm selection
- Model test step
 - Estimation of model performance



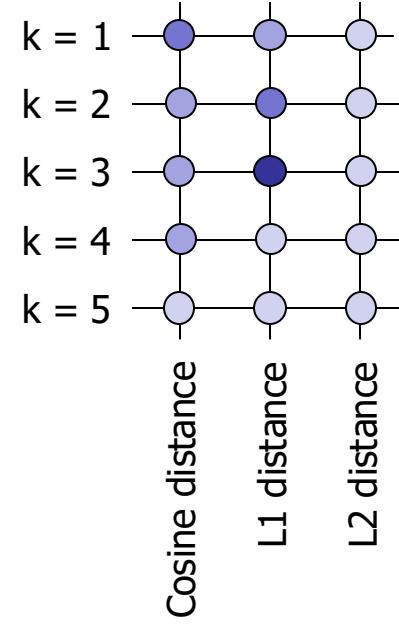
Model validation

- Each ML algorithm has different “hyperparameters”
 - E.g., *max depth* for decision trees, K for KNN
- We need to find a “good choice” of hyperparams
- Different strategies exist, such as:
 - Manual search
 - Grid search
 - Randomized search
 - Bayesian optimization
- A “validation set” is used to pick the best configuration



Grid search

- We define a set of relevant hyperparameters
 - Each with a set of candidate values
- Compute the Cartesian product and train models for all possible configurations
- Computationally expensive
 - But parallelizable!
- Guarantees to find the “best solution” among the candidates



Example, for KNN.
Hyperparams = {K, distance}
Candidate values:

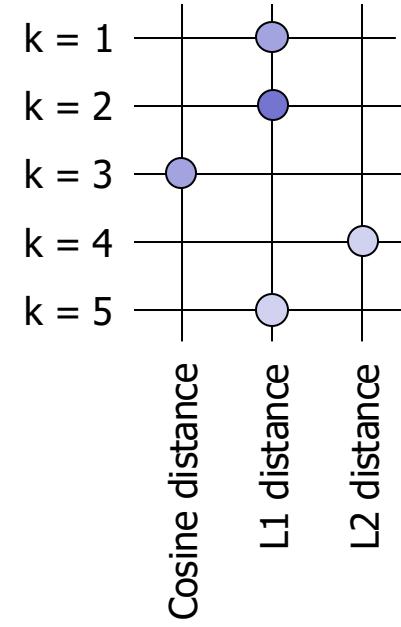
- k = {1, 2, 3, 4, 5}
- distance = {cosine, L1, L2}

configurations = $5 * 3 = 15$



Randomized search

- We define a set of relevant hyperparameters
 - Each with a set of candidate values
- A “budget” is defined
 - How many models should be tested
- A random subset of all configurations is sampled and tested
- Computationally inexpensive
 - And parallelizable!
- May not find “best” solution
- Doesn’t learn from past outcomes



Example, for KNN.
Hyperparams = {K, distance}
Candidate values:

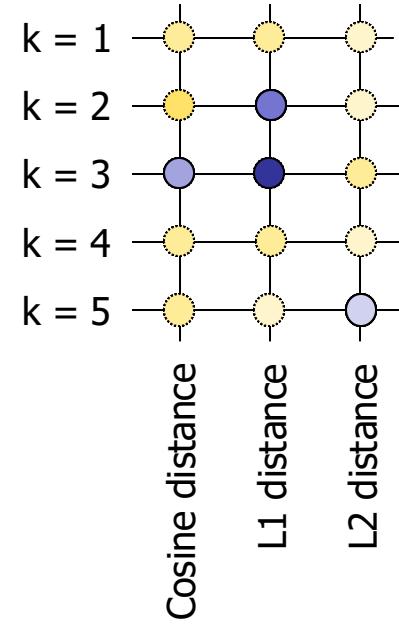
- k = {1, 2, 3, 4, 5}
- distance = {cosine, L1, L2}

configurations = 5
(user-defined budget)



Bayesian optimization

- Bayesian optimization is a strategy used to find the maximum of an **expensive, black box objective function**
 - E.g., a function that maps hyperparameters configurations to performance
- Estimate the values of the performance, with uncertainties
- Test configurations that are expected to provide an improvement
 - With an exploration/exploitation trade-off
- More efficient w.r.t. grid/random search, since it uses past information
- Not (easily) parallelizable
- Surrogate model adds computational overhead



Example, for KNN.
Hyperparams = {K, distance}
Candidate values:

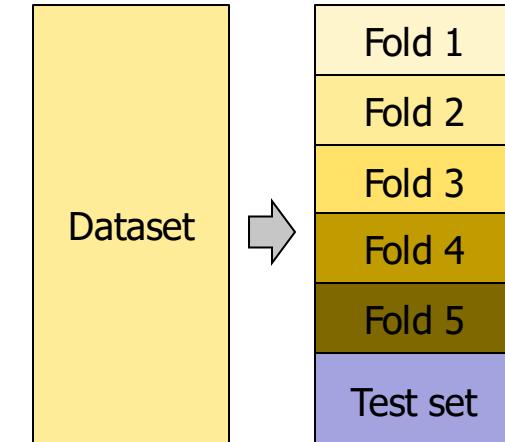
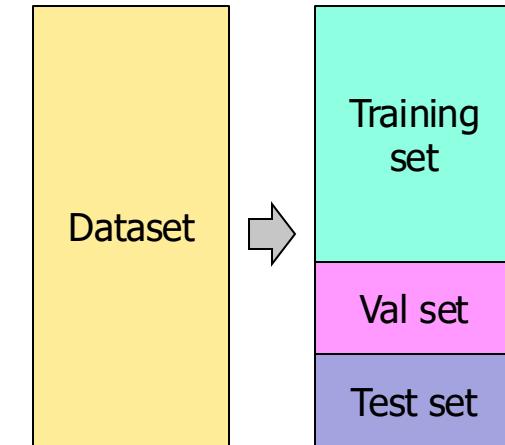
- k = {1, 2, 3, 4, 5}
- distance = {cosine, L1, L2}

configurations = 5
(user-defined budget)



Model performance estimation

- Typical dataset size
 - **Training set** 60% of labeled data
 - **Validation set** 20% of labeled data
 - **Test set** 20% of labeled data
- Splitting labeled data
 - Use hold-out to split in
 - training+validation
 - test
 - Use cross validation to split in
 - training
 - validation

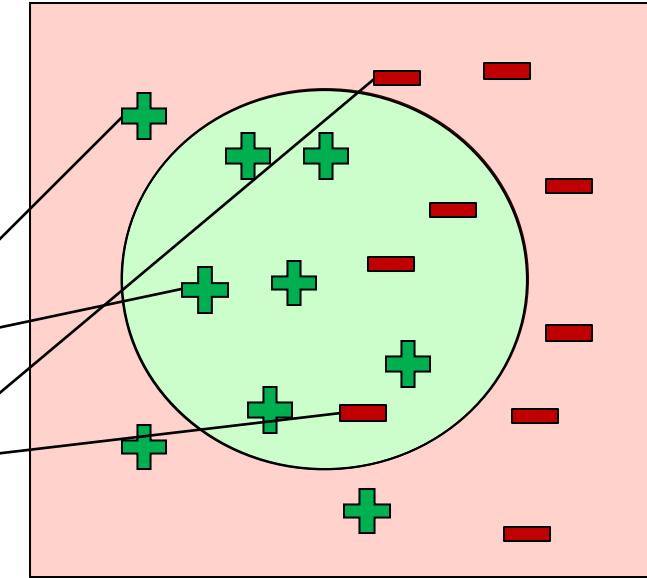




Metrics for model evaluation

- Evaluate the predictive accuracy of a model
- Confusion matrix
 - binary classifier

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	True positive (TP)	False negative (FN)
	Class>No	False positive (FP)	True negative (TN)





Accuracy

- Most widely-used metric for model evaluation

$$\text{Accuracy} = \frac{\text{Number of correctly classified objects}}{\text{Number of classified objects}}$$

- Not always a reliable metric



Accuracy

- For a binary classifier

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	True positive (TP)	False negative (FN)
	Class>No	False positive (FP)	True negative (TN)

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



Limitations of accuracy

- Consider a binary problem

- Cardinality of Class 0 = 9900
 - Cardinality of Class 1 = 100

- Model

$() \rightarrow \text{class } 0$

- Model predicts everything to be class 0
 - accuracy is $9900/10000 = 99.0\%$
- Accuracy is misleading because the model does not detect any class 1 object



Limitations of accuracy

- Classes may have different importance
 - Misclassification of objects of a given class is more important
 - e.g., ill patients erroneously assigned to the healthy patients class
- Accuracy is not appropriate for
 - unbalanced class label distribution
 - different class relevance



Class specific measures

- Evaluate separately for each class C

$$Precision(C) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects assigned to } C}$$

$$Recall(C) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects belonging to } C}$$

- Maximize

$$F_1 \text{ score}(P) = \frac{2 \cdot Recall(P) \cdot Precision(P)}{Precision(P) + Recall(P)}$$



Class specific measures

- For a binary classification problem
 - on the confusion matrix, for the positive class

$$Precision(P) = \frac{TP}{TP + FP}$$

$$Recall(P) = \frac{TP}{TP + FN}$$

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	True positive (TP)	False negative (FN)
	Class>No	False positive (FP)	True negative (TN)

$$F_1 \text{ score}(P) = \frac{2 \cdot Recall(P) \cdot Precision(P)}{Recall(P) + Precision(P)} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$



Multi-Class Metrics

- Precision, Recall, and F1-score are defined *per class*
- We can aggregate per-class metrics to get a single value
- *Macro average*: weight all classes equally
 - Regardless of their supports (number of samples)
 - Good for imbalanced datasets
- *Weighted average*: weigh classes according to their support
 - Good when performance on larger classes matters more

Precision(C1) = 0.7, 300 samples

Precision(C2) = 0.5, 200 samples

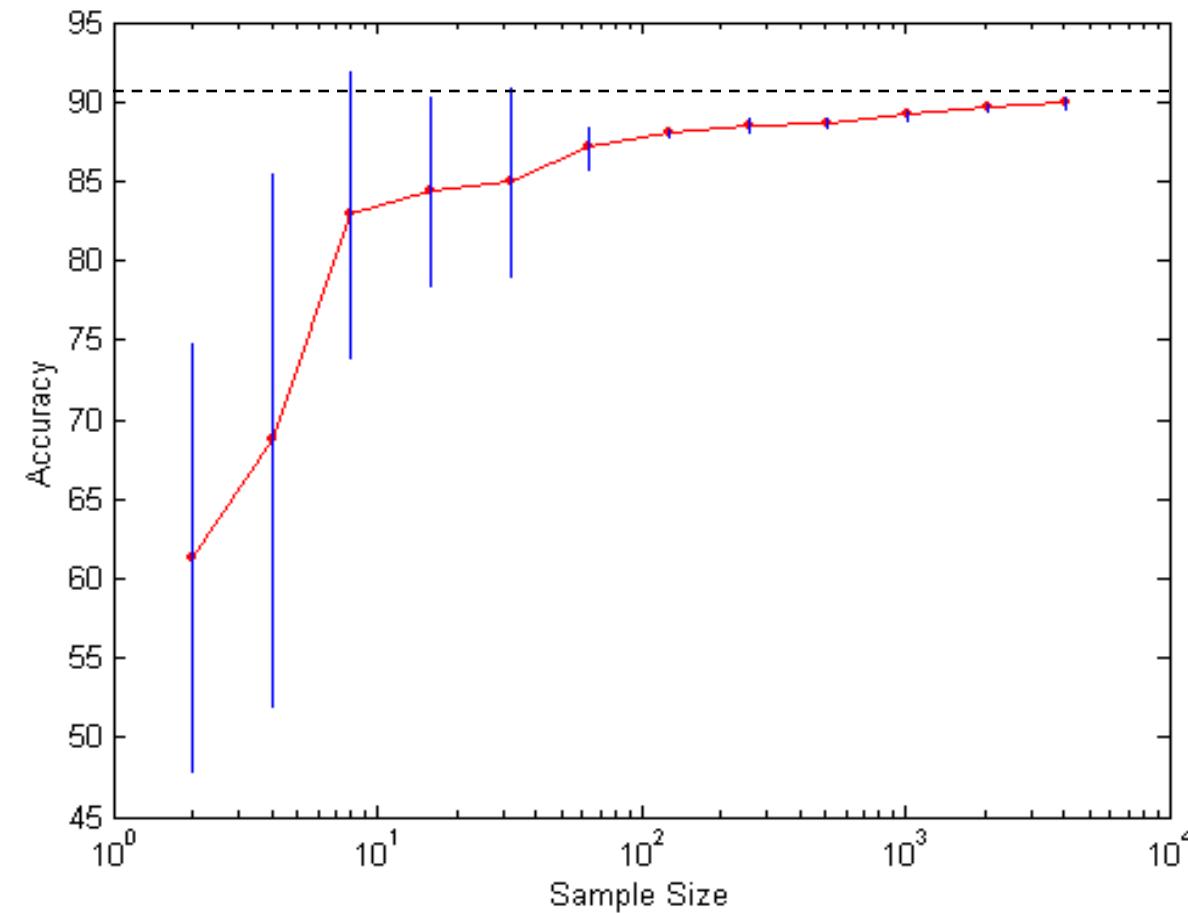
Precision(C3) = 0.6, 500 samples

Macro Precision = $(0.7 + 0.5 + 0.6) / 3 = 0.6$

Weighted Precision = $(0.7 * 0.3 + 0.5 * 0.2 + 0.6 * 0.5) = 0.61$



Learning curve



- Learning curve shows how accuracy changes with varying training sample size
- Error bars indicate the variance in performance, for different training subsets
- For small datasets, we observe high variance in results
- Increasing dataset size reduces variance and improves performance
 - But, notice the log scale!



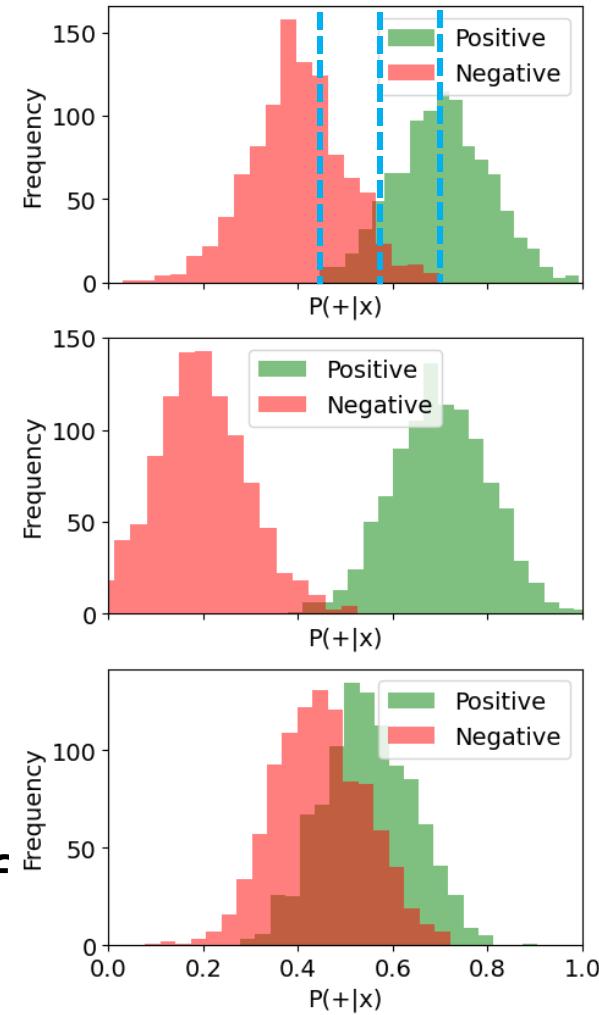
ROC (Receiver Operating Characteristic)

- Characterizes the trade-off between positive hits and false alarms
- ROC curve plots
 - TPR, **True Positive Rate** (on the y-axis)
$$TPR = TP / P = TP/(TP+FN)$$
against
 - FPR, **False Positive Rate** (on the x-axis)
$$FPR = FP / N = FP/(FP + TN)$$



ROC usage

- Classification models can generally assign a probability of membership to a class
 - E.g., in the binary case, $P(+|x)$
 - $P(-|x) = 1 - P(+|x)$
- We can choose a *thresholding value τ*
 - If $P(+|x) > \tau \rightarrow$ Predict “+”
 - Otherwise, predict “-”
- We can use the ROC curve to study how the choice of τ affects the performance of the classifier



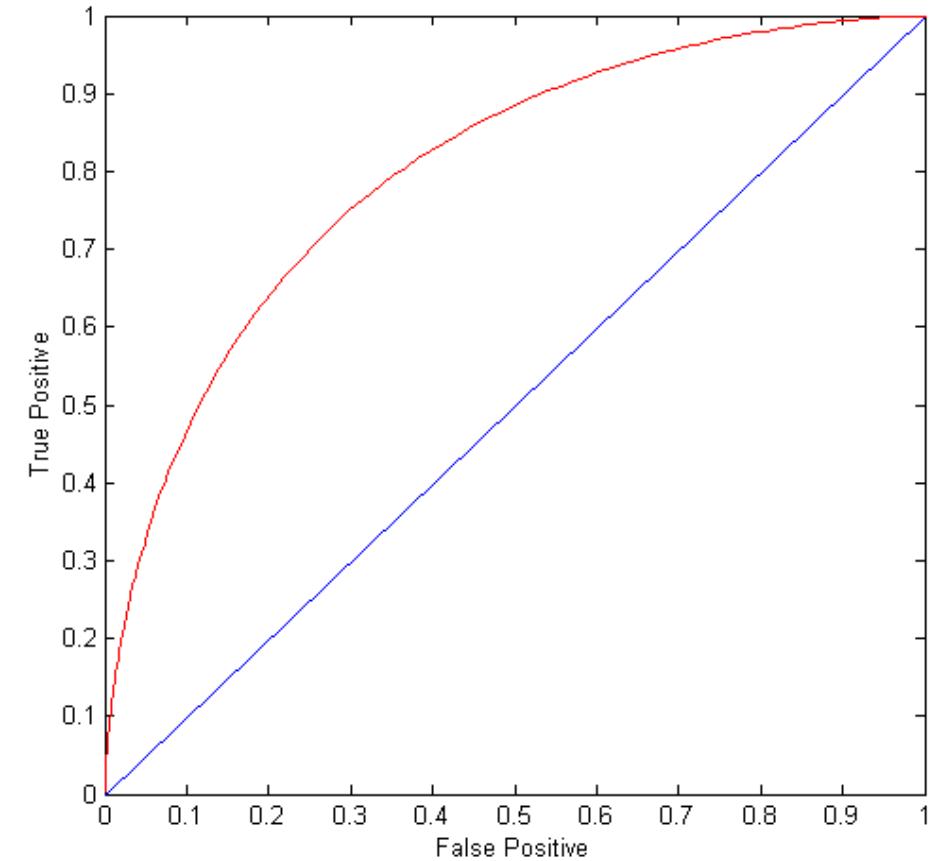


ROC curve

(FPR, TPR)

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal

- Diagonal line
 - Random guessing
 - Below diagonal line
 - prediction is opposite of the true class





How to build a ROC curve

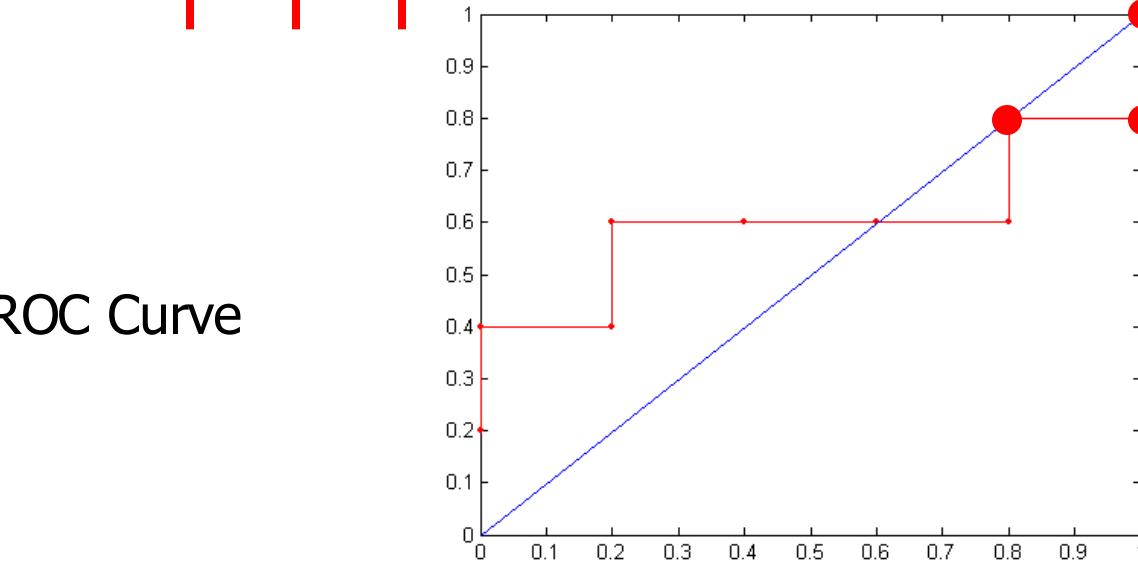
Instance	$P(+ x)$	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance $P(+|x)$
- Sort the instances according to $P(+|x)$ in decreasing order
- Apply threshold at each unique value of $P(+|x)$
- Count the number of TP, FP, TN, FN at each threshold
 - TP rate
$$TPR = TP / (TP + FN)$$
 - FP rate
$$FPR = FP / (FP + TN)$$



How to build a ROC curve

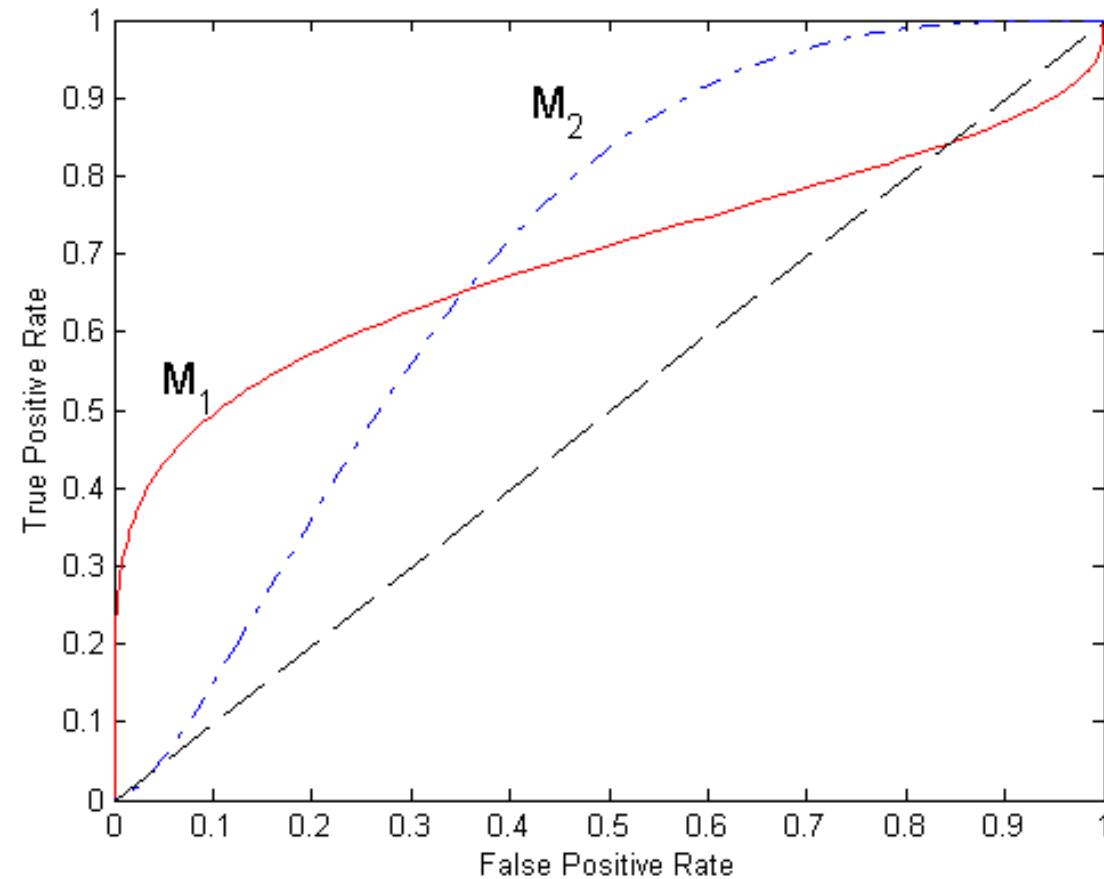
Class	+	-	+	-	-	-	+	-	+	+	
$P(+ A)$	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
→ TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
→ FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



ROC Curve



Using ROC for Model Comparison



- No model consistently outperforms the other
 - M_1 is better for small FPR
 - M_2 is better for large FPR
- Area under ROC curve
 - Ideal Area = 1.0
 - Random guess Area = 0.5

Regression

D^B_MG



Data Base and Data Mining Group of Politecnico di Torino

Flavio Giobergia
Politecnico di Torino



Introduction to regression

- **Objective:** Predict a *continuous outcome variable* based on *one or more predictor variables*
 - i.e., learn a function $f : \mathcal{X} \rightarrow \mathbb{R}$
 - We refer to the outcome as the *dependent variable*, and to the predictors as the *independent variables*
- Useful for:
 - Making predictions
 - Understanding relationships between variables
 - Identifying significant predictors

Linear regression

D^B_MG

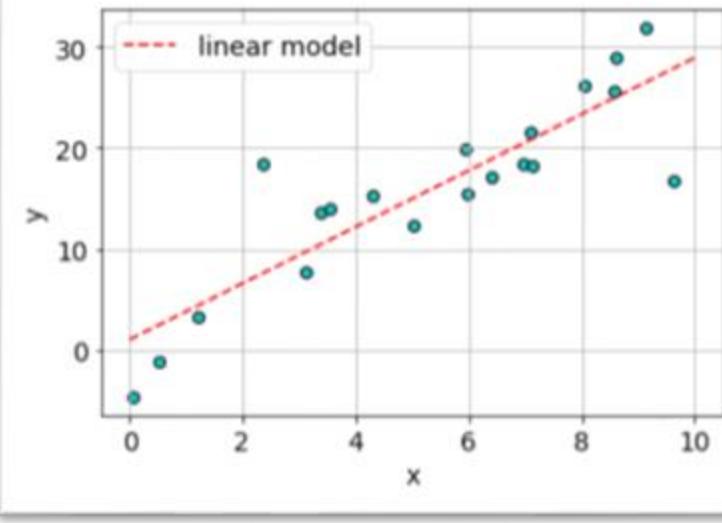


Data Base and Data Mining Group of Politecnico di Torino



Linear regression

- Used to model linear relationships between predictors and outcome
- Assumption:
 - There is a linear relation between the independent (x) and dependent (y) variables
 - $y = \theta_0 + \theta_1 x + \varepsilon$ (observation)
 - ε represents a stochasticity that we cannot model
- Simple linear regression:
 - Goal: estimate θ_0, θ_1 so that we can build our own model!
 - $\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 x$ (prediction)
- ε : residual (difference between predictions and observations)

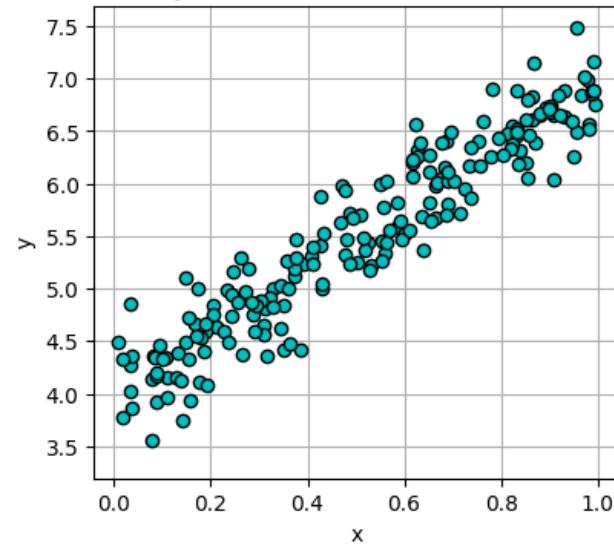




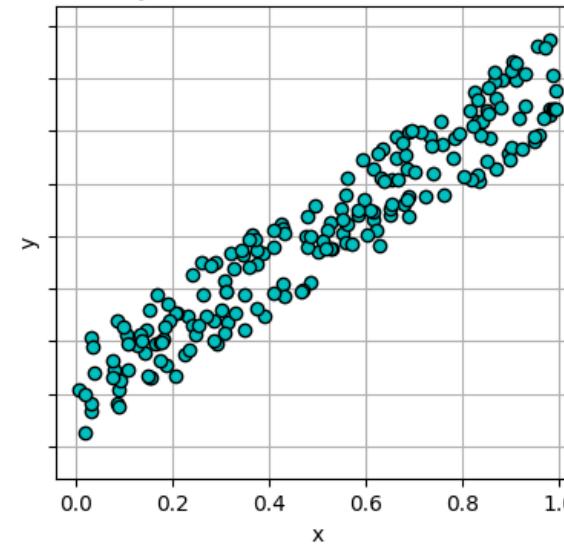
Residuals

- Residuals are expected to be:
 - Normally distributed
 - Homoskedastic

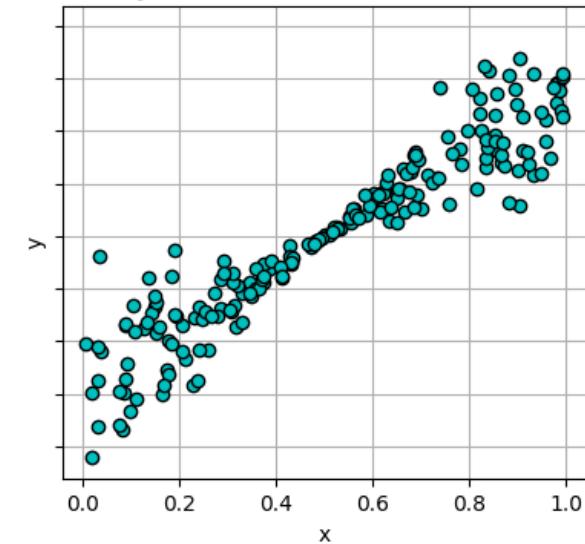
Normally distributed (✓), homoskedastic (✓)



Uniformly distributed (✗), homoskedastic (✓)



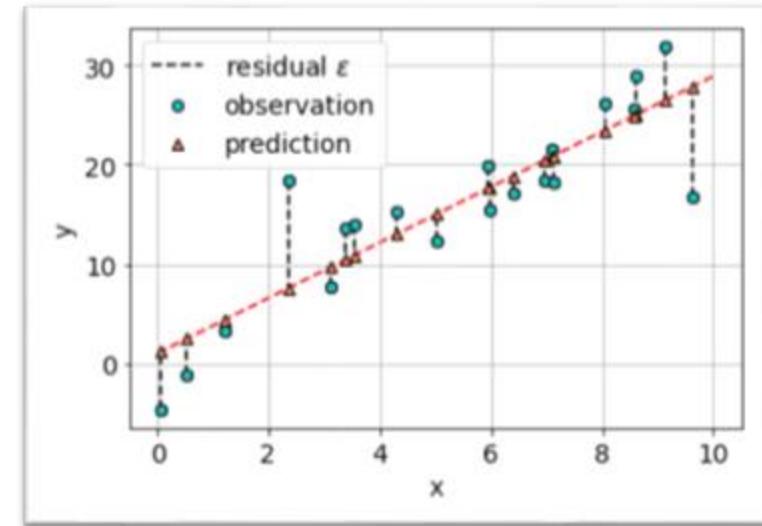
Normally distributed (✓), heteroskedastic (✗)





Residuals, error

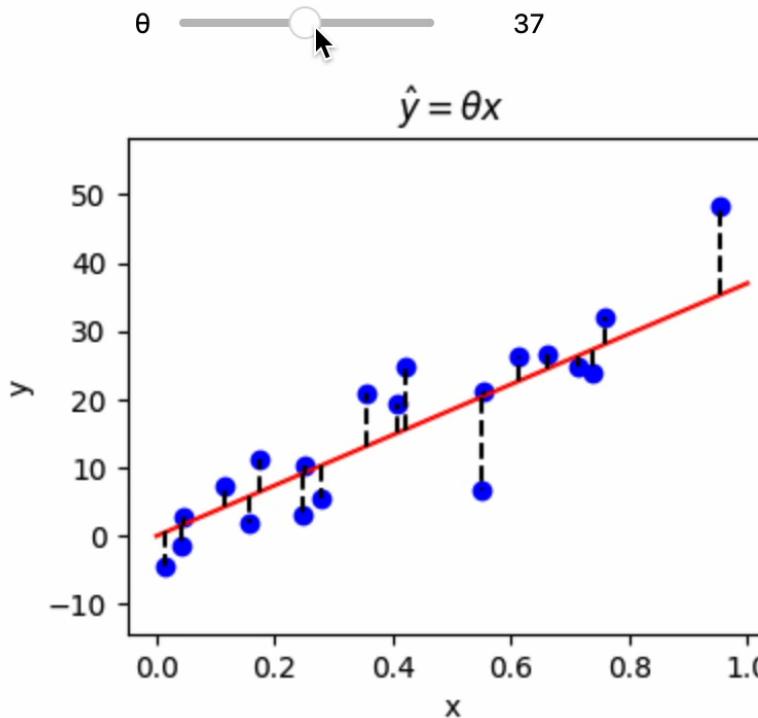
- We can compute the squared error for x_i
 - $(y_i - \hat{y}_i)^2 = \varepsilon_i^2$
- Properties of squared errors:
 - Quantify quality of prediction
 - The smaller the better!
 - Always positive
 - “Stretches” error:
 - $(\text{Large error})^2 = \text{even larger error}$
 - $(\text{Small error})^2 = \text{smaller error}$
- Error over the entire dataset: mean squared error (MSE)
 - $MSE = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$





Residuals, error

- The MSE, $\frac{1}{n} \sum_i (y_i - \theta_0 - \theta_1 x_i)^2$, is a quadratic function of the parameters θ
- So, it has a single minimum, which are the “best” values for θ





Error minimization

- $MSE(\theta_0, \theta_1) = \frac{1}{n} \sum_i (y_i - \theta_0 - \theta_1 x_i)^2$
 - “Cost function” to be minimized
- We want to find θ_0, θ_1 that minimize the MSE
- MSE is a quadratic function of θ_0, θ_1
 - Minimum for $\frac{\partial MSE}{\partial \theta_0} = 0, \frac{\partial MSE}{\partial \theta_1} = 0$
- Linear regression chooses the parameters θ_0, θ_1 that minimize the SSE
 - $\theta_0 = \bar{y} - \theta_1 \bar{x}$
 - $\theta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$



Multivariate case

- Similarly, we can define a problem with n independent variables
- $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- Multiple linear regression:
 - $\hat{y} = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$
 - $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$
 - as a scalar product of $\mathbf{x} = (1, x_1, x_2, \dots, x_n)$ and $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_n)$
- Solution:
 - $\boldsymbol{\theta} = (X^T X)^{-1} X^T Y$
- The coefficients help understand the relationship between the independent and dependent variables
 - E.g. θ_1 indicates the change in the predicted y for a one-unit increase in x_1 , all else being equal



Non-linear relationships

- We may want to model non-linear relationships
- We can *add new features*, non-linear transformations of the original one(s)
 - E.g., if we expect an inverse quadratic relationships between x and y , we introduce a new feature, $\frac{1}{x^2}$
- Then, we use a “classic” linear regression
 - The model learns a separate coefficient for each feature
 - $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \leftrightarrow \theta_0 + \theta_1 x + \theta_2 \frac{1}{x^2}$

x	y
0.4	105
0.3	84
0.2	210

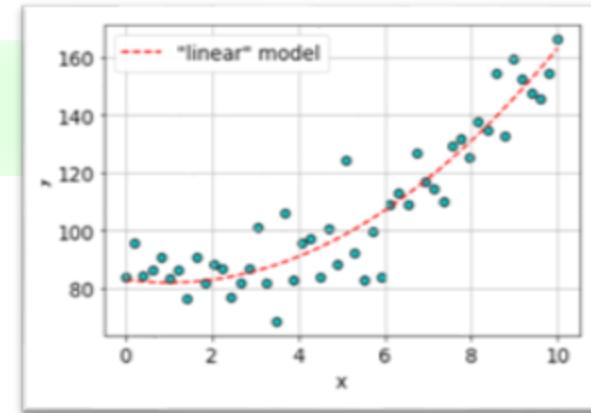


$x_1 = x$	$x_2 = \frac{1}{x^2}$
0.4	2.5
0.5	2
0.2	5



Polynomial regression

- We can introduce more flexibility in representing relationships with a **polynomial regression**
 - i.e., add new polynomial features up to degree n
 - Increases **model capacity**
 - Univariate: $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 \dots + \theta_n x^n$
- For multivariate problems, we can add either **powers**, or **interactions** (or both!)
 - Powers ($x_1^2, x_2^2, x_1^3 \dots$)
 - Interactions ($x_1 x_2, x_1 x_3, \dots$), capturing relations between variables at different polynomial degrees
 - E.g., $x_1, x_2, x_3, n = 2 \rightarrow x_1^2, x_2^2, x_3^2, x_1 x_2, x_1 x_3, x_2 x_3$
 - !The total number of features increases combinatorially!



Evaluation

D^B_MG



Data Base and Data Mining Group of Politecnico di Torino



MSE, RMSE, MAE

■ Mean squared error (MSE)

- $MSE = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$
- Sometimes not normalized by # points
 - SSE (Sum of SE)

■ RMSE (root MSE)

- $RMSE = \sqrt{MSE}$
- Same unit of measurement as the dependent var.

■ Mean absolute error (MAE)

- $MAE = \frac{1}{n} \sum_i |y_i - \hat{y}_i|$
- Penalizes more «small» errors (w.r.t. MSE)



R-squared (R^2)

- R^2 : proportion of the variance in the dependent variable that is explained by the independent variables

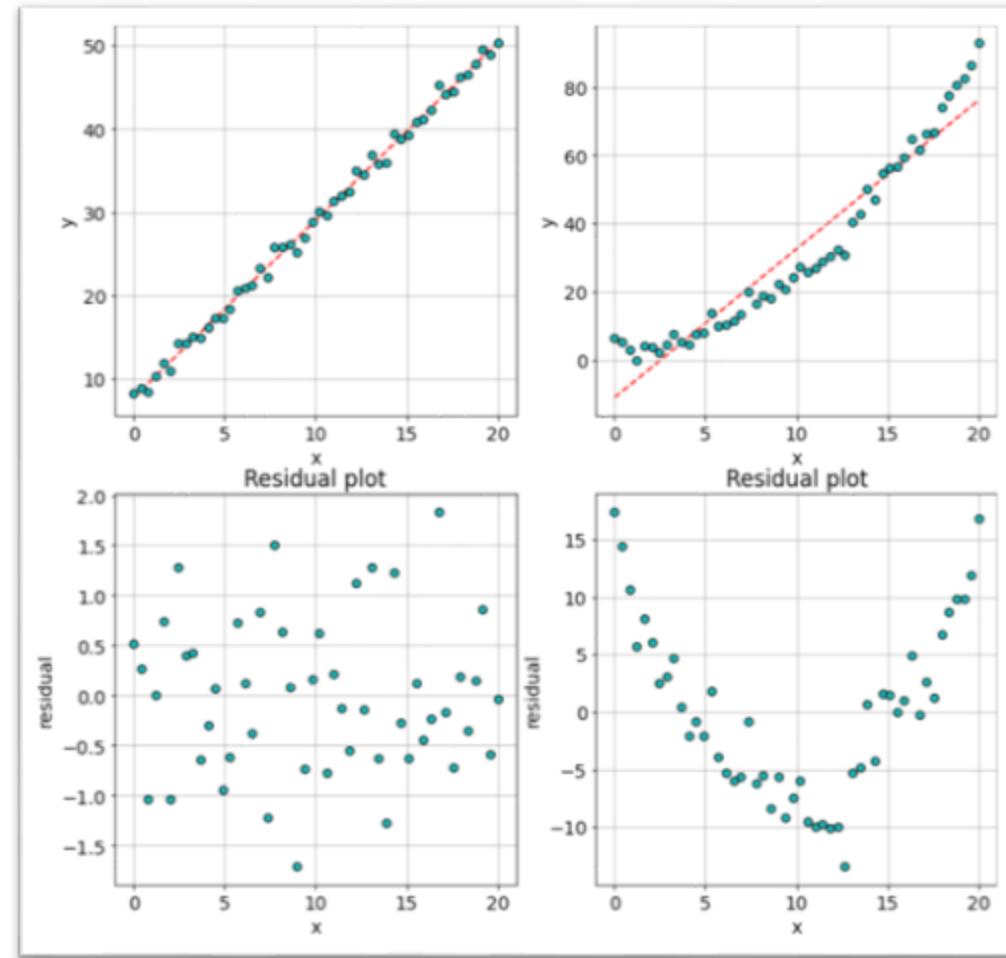
$$R^2 = 1 - \frac{MSE}{\sigma^2}$$

- Edge cases:
 - Model predicts everything perfectly
 - $MSE = 0, R^2 = 1$ (upper bound)
 - Model is no better than predicting mean value of y
 - $MSE = \sigma^2, R^2 = 0$
 - Model is worse than predicting mean value
 - $MSE > \sigma^2, R^2 < 0$



Residual plots

- Residual plots: visual assessment of the goodness of fit of a regression model
 - Expecting residuals to be random scattered around zero, with constant variance, and no patterns



Regularization

D^B_MG

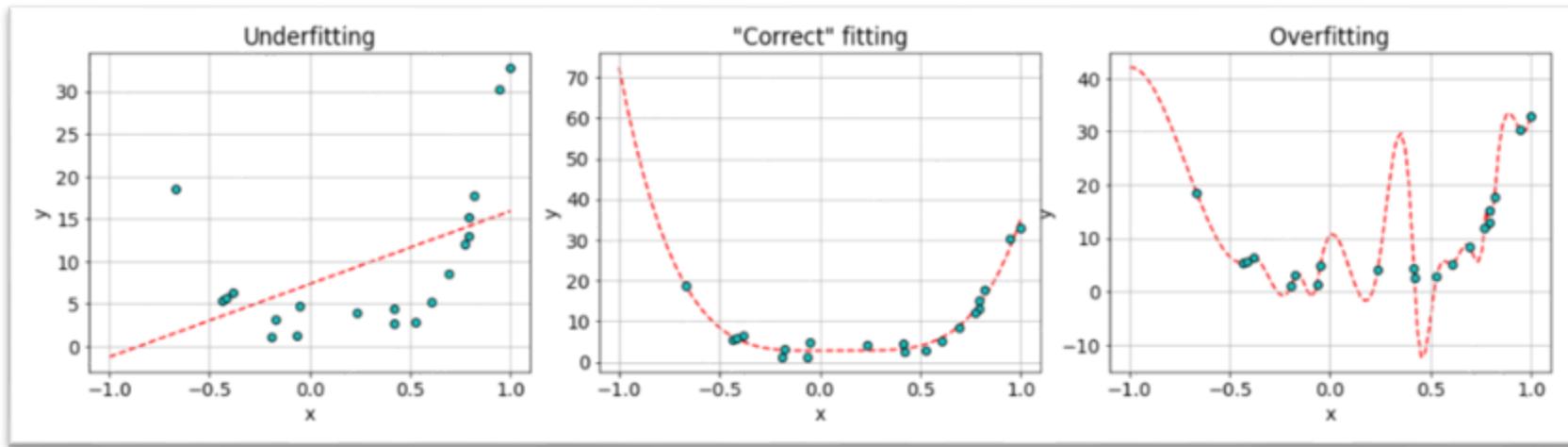


Data Base and Data Mining Group of Politecnico di Torino



Overfitting and underfitting

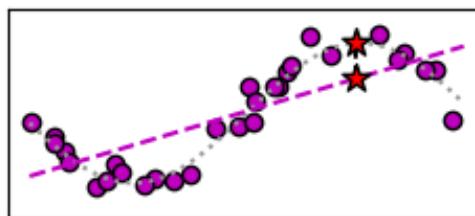
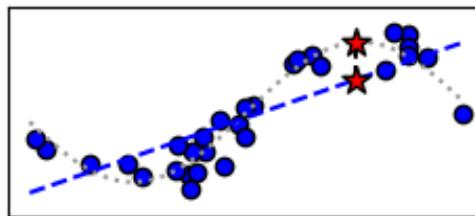
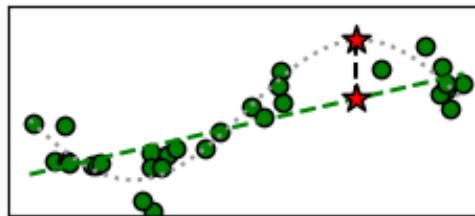
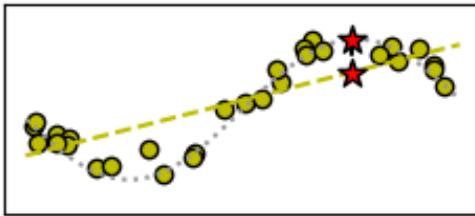
- **Overfitting:** the model is **too complex** and fits the training data too closely (**high variance**)
 - Poor performance on test data
 - May occur when using higher degree polynomials
- **Underfitting:** the model is **too simple** and does not capture the underlying relationships (**high bias**)
 - Poor performance on training and test data
 - May occur when using low degree polynomials



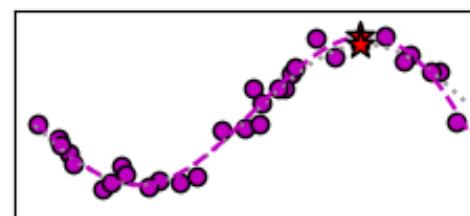
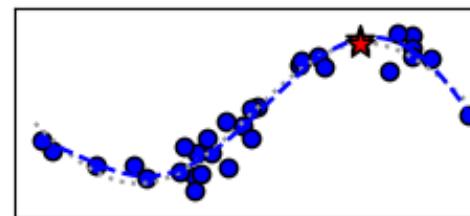
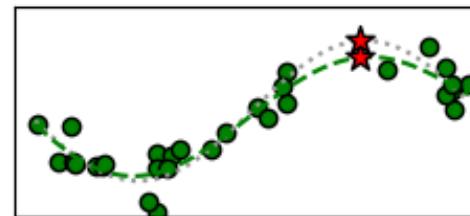
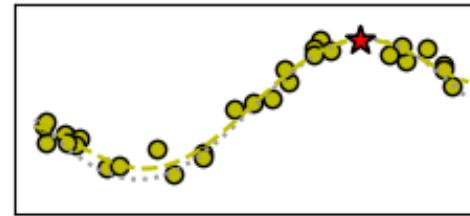


High bias vs high variance

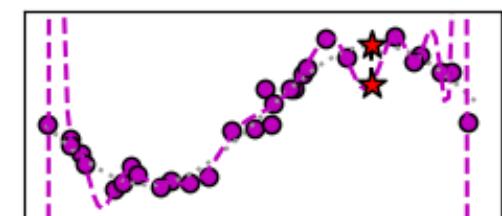
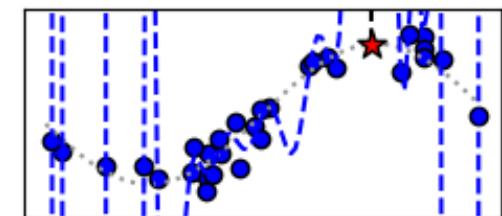
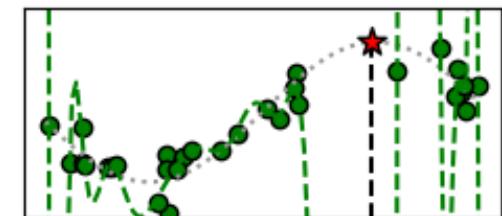
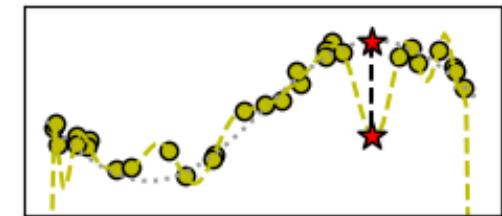
Underfitting (High bias)



"Correct" fitting



Overfitting (High variance)





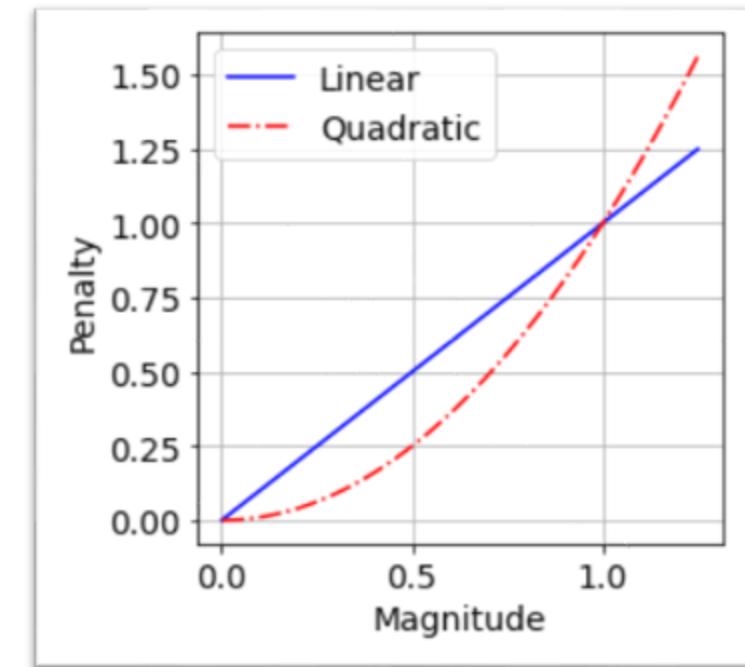
Preventing overfitting

- We can generally prevent overfitting by:
 - Reducing model capacity
 - (e.g., reduce the polynomial degree used)
 - Increasing the dataset size
 - Introducing regularization techniques



Regularization techniques

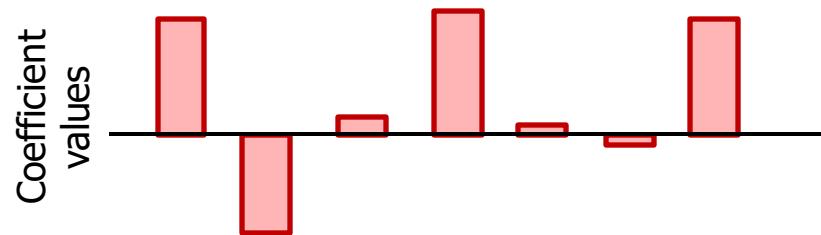
- Allow model to use high capacity, but penalize it if used unnecessarily
- Penalty term in the cost function
- L1 (Lasso) penalizes all non-zero weights linearly
 - $Cost = MSE + \lambda ||\theta||_1$
 - Bring θ values to 0 if not strictly needed
- L2 (Ridge) penalizes ≈ 0 values less than ≈ 0 values
 - $Cost = MSE + \lambda ||\theta||_2$
 - Allows θ values to be ≈ 0 for small contributions



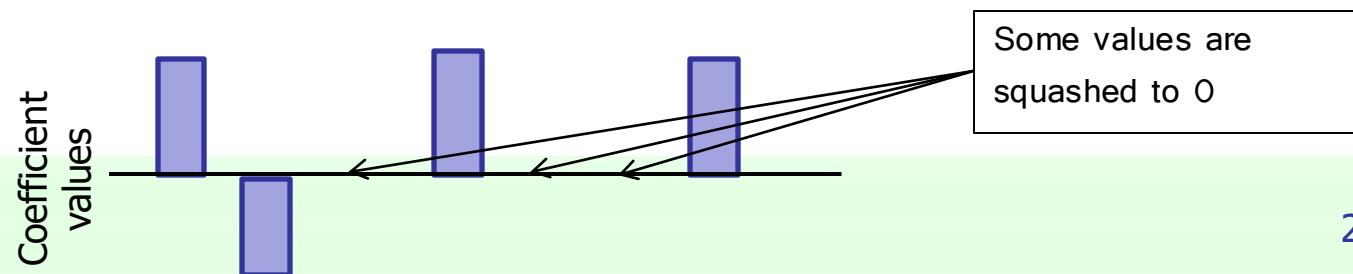


Regularization techniques

- Ridge tends to lower uniformly all the coefficients
 - Coefficients already close to 0 have little effect on the sum of **squares** (if $x \approx 0$, $x^2 < x$)



- Lasso tends to assign the value 0 to **some** coefficients (feature selection)
 - Also small coefficients affect the sum



Other regressors

D^B_MG

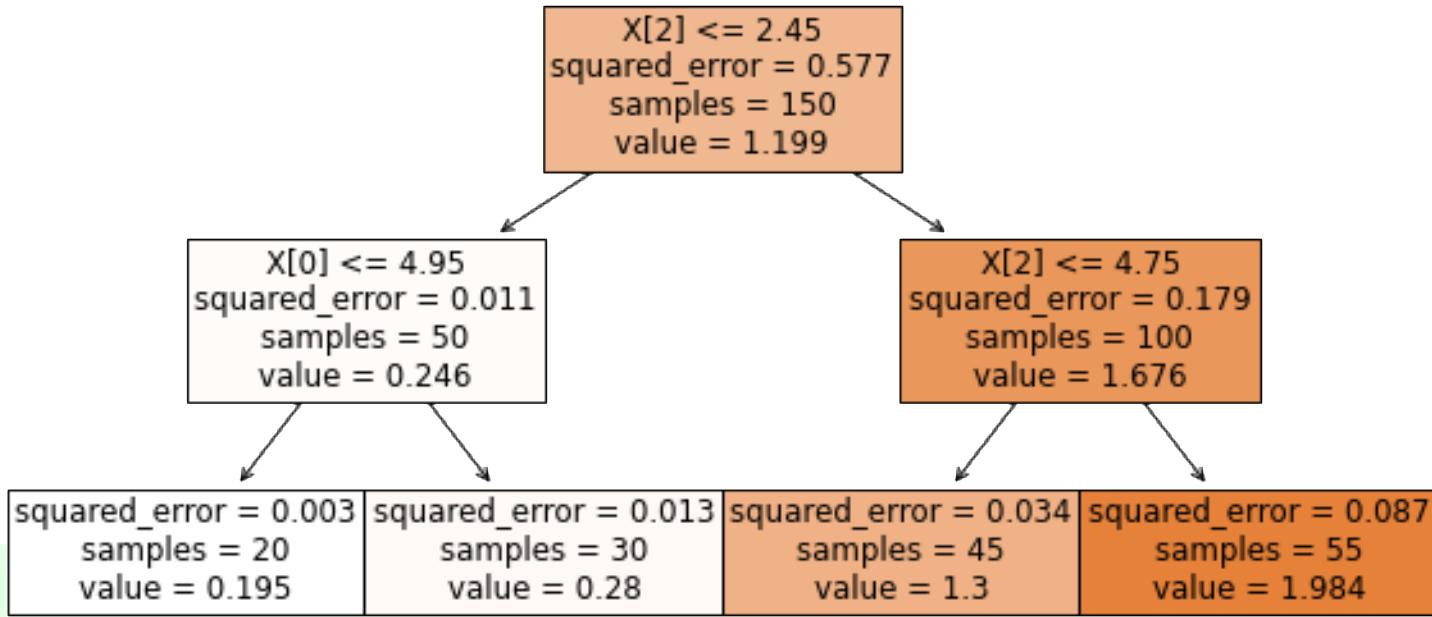


Data Base and Data Mining Group of Politecnico di Torino



Tree-based regression

- We can build decision trees for regression
 - Real values used as targets instead of classes
 - Node impurity computed as variance
 - Each leaf assigns average value of points in it





Other techniques

- Random forests can be obtained by aggregating the output of decision tree regressors (e.g., by averaging them)
- In KNN, we can produce the predicted outcome as the (possibly weighted) average of the neighbors' "votes"
- Neural networks natively produce continuous outputs

Data Science & Machine Learning Lab

Introduction to
Deep Learning

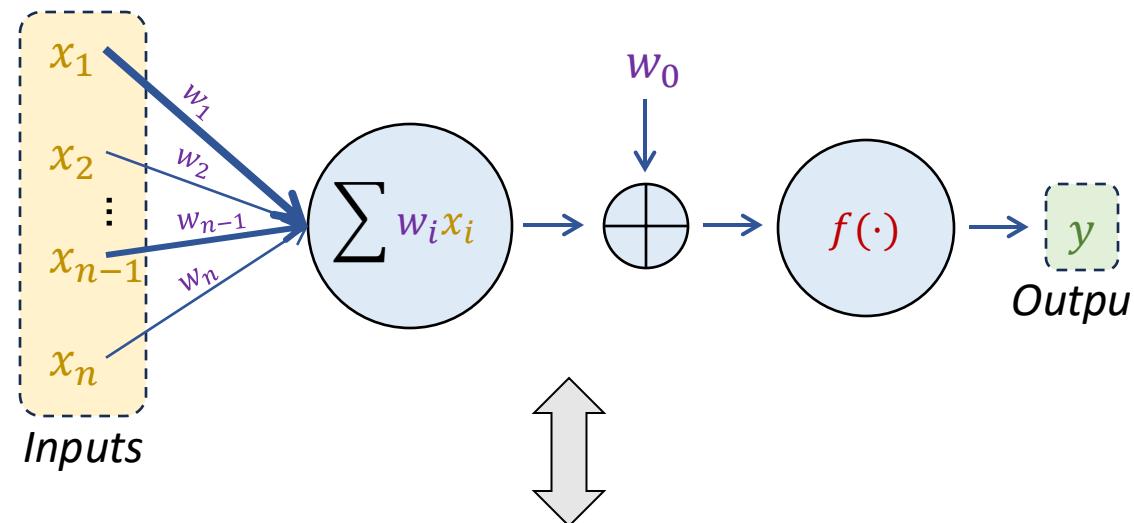
Flavio Giobergia



The perceptron

- The perceptron is the simplest unit of neural networks
- It takes an input with *multiple features*, and does the following:
 - It weights each input feature with a given *weight*,
 - It produces a weighted sum of the inputs, and
 - It applies a *function* to the output
- $y = f(w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n)$

The perceptron



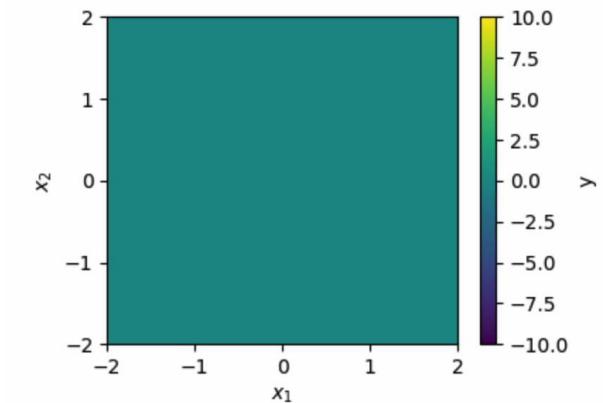
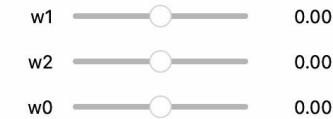
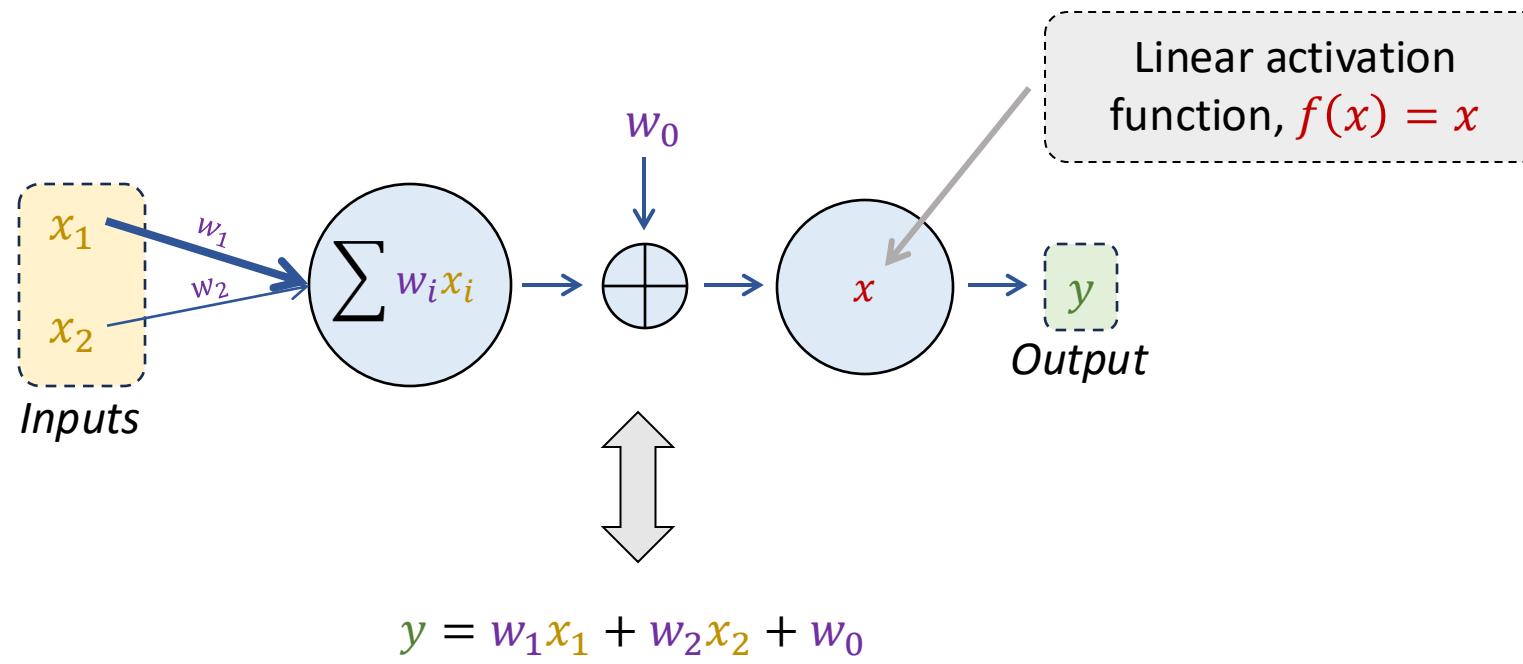
- $x = (x_1, x_2, \dots, x_n)$ is the input sample
- y represents the output of the perceptron.
- $f(\cdot)$ represents a non-linear “activation” function
- w_i (and w_0) are weights (and bias), which are “learned”

Or, in other words, $y = f\left(\sum_{i=0}^n w_i x_i\right) = f(w^T x)$ and $x_0 = 1$

Note

With the exception of $f(\cdot)$, this looks like the classic *linear regression*.
And if $f(\cdot) = \sigma(\cdot)$ (sigmoid function), this looks like the (just as classic) *logistic regression*.

The perceptron, in 2D



The perceptron can be used to represent a family of functions,
 $y = w_1 x_1 + w_2 x_2 + w_0$

Various values of w_0, w_1, w_2 define the different functions that can be learned by the perceptron.

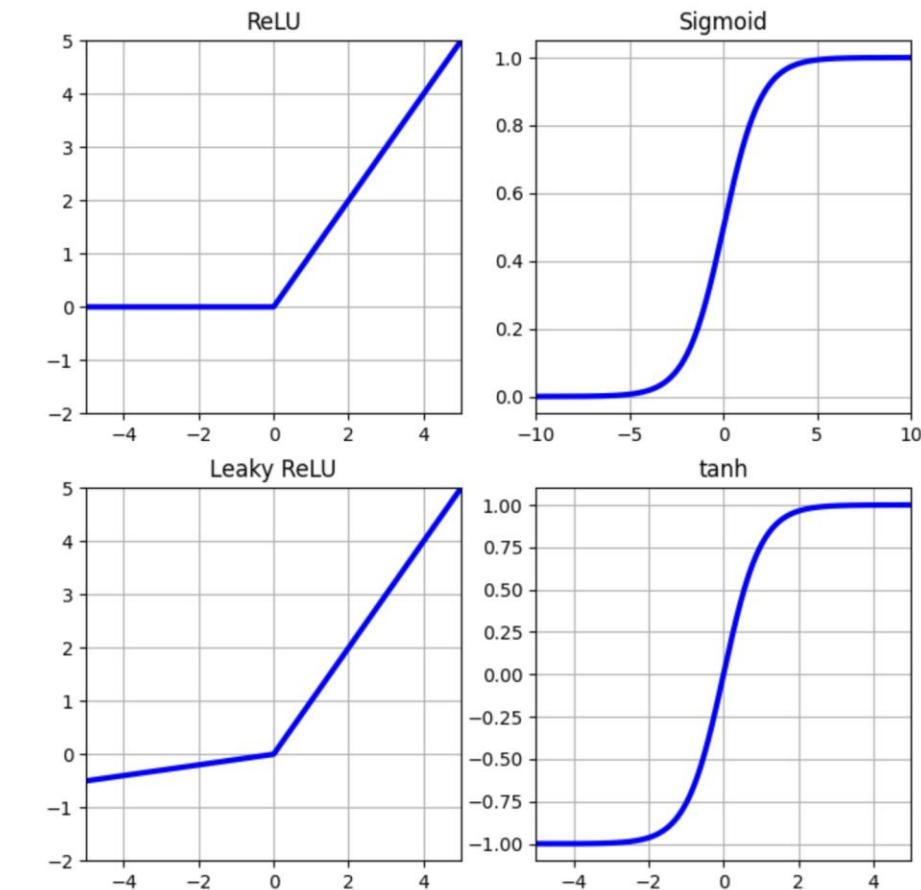
Activation functions

- Activation functions are used for two *main* reasons:

1. Enforce *properties* on perceptron's output
 - E.g., sigmoid → binds output to $[0, 1]$ range
2. Introduce *non-linearities* in the model
 - + some others (faster convergence, sparsity, ...)

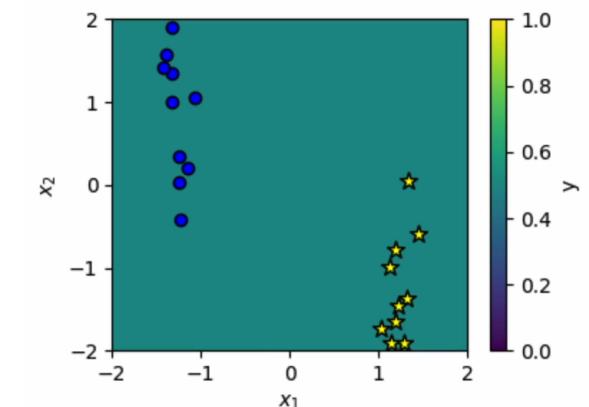
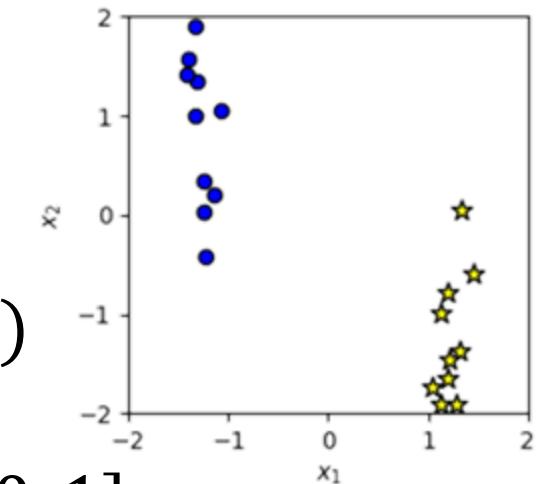
- Commonly adopted functions:

- ReLU
- Sigmoid
- Leaky ReLU
- Tanh
- Softmax
- Linear
- GeLU

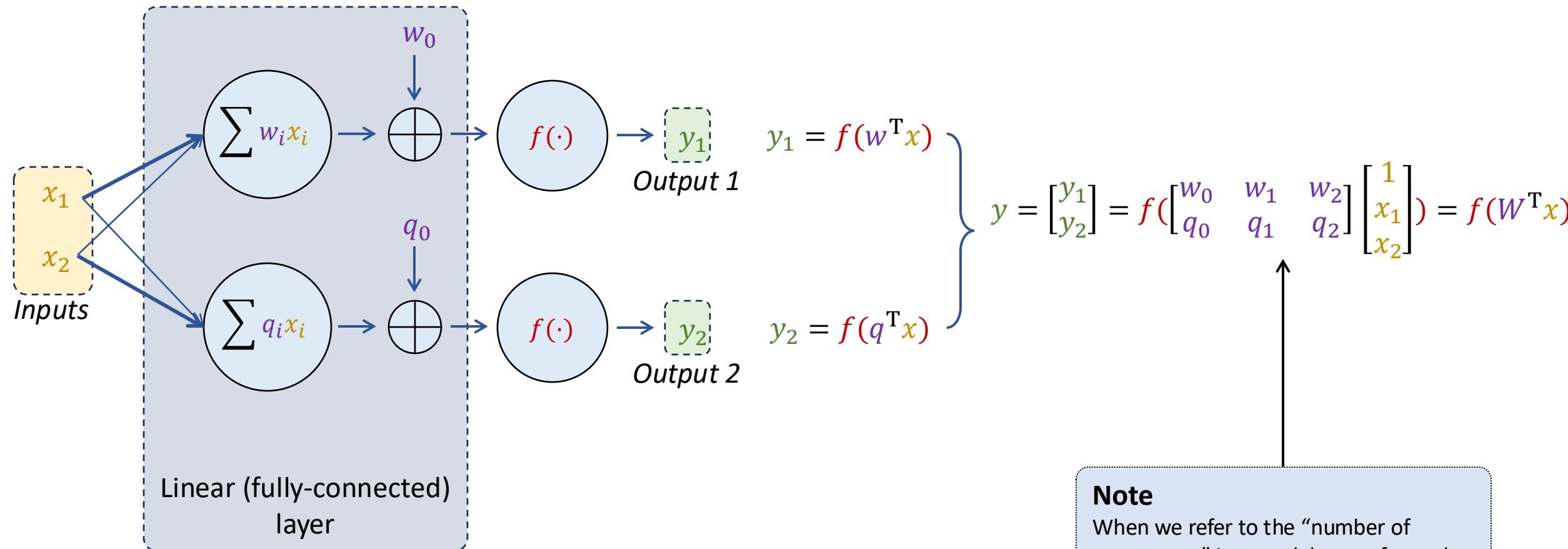


1. Enforce *properties* on perceptron's output

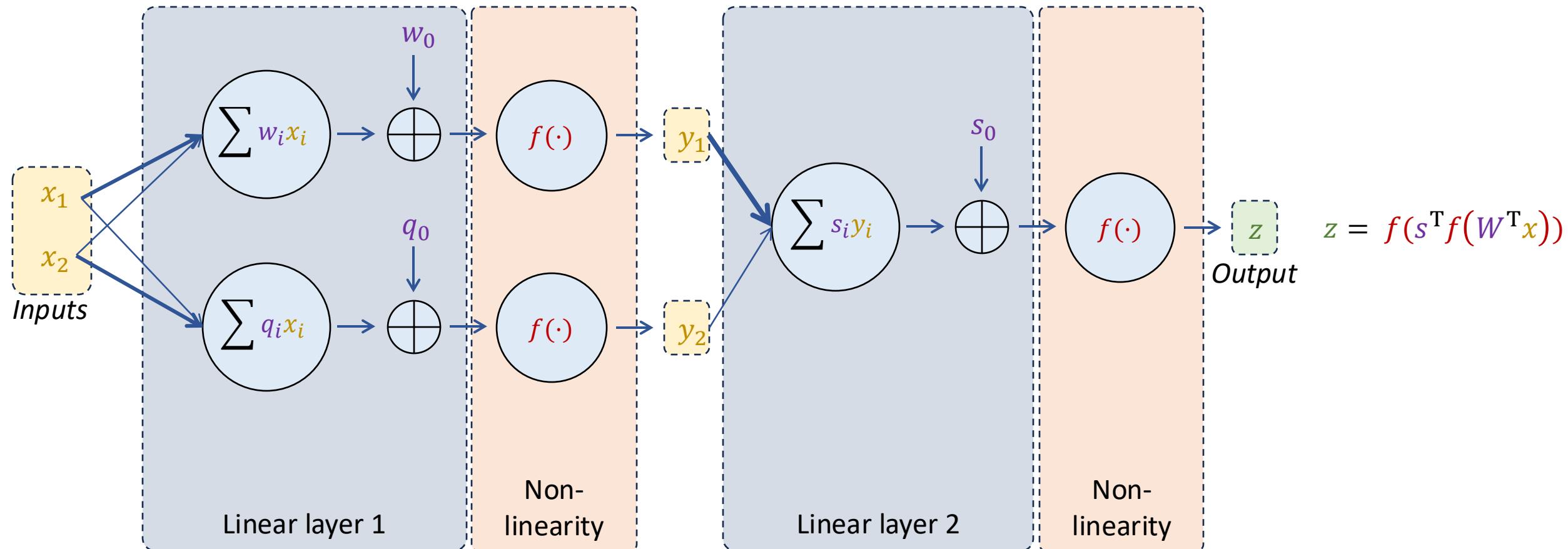
- Binary classification problem
 - Separate positive (\star) and negative (\circ) samples
- For a point $x \in \mathbb{R}^2$, the perceptron can predict $p(\star|x)$
 - For the binary case, this implies $p(\circ|x) = 1 - p(\star|x)$
- To get a valid probability, we must enforce $p(\star|x) \in [0, 1]$
 - We already have $p(\circ|x) + p(\star|x) = 1$ by construction
- The Sigmoid maps any value in \mathbb{R} to the range $[0, 1]$
 - i.e., the perceptron's output (in \mathbb{R}) is squashed to $[0, 1]$
 - $\sigma(x) = \frac{1}{1+e^{-x}}$



Adding some perceptrons

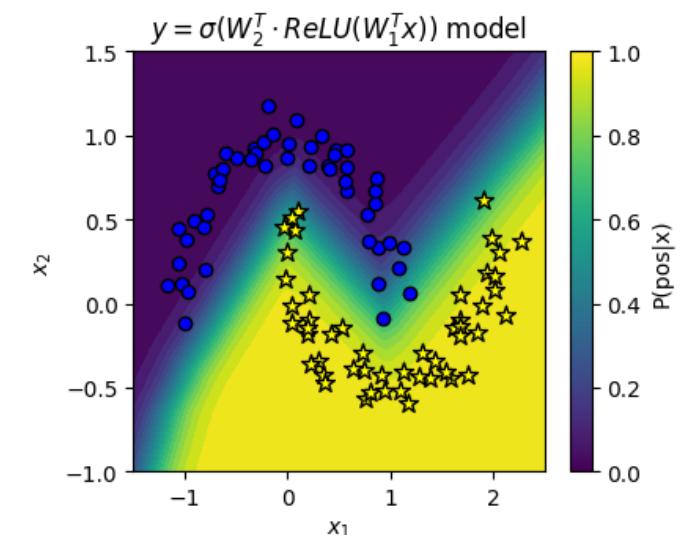
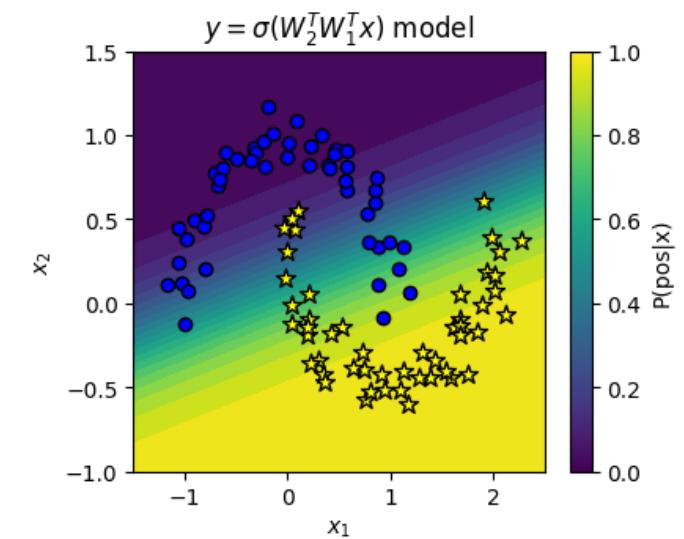


and adding other layers!



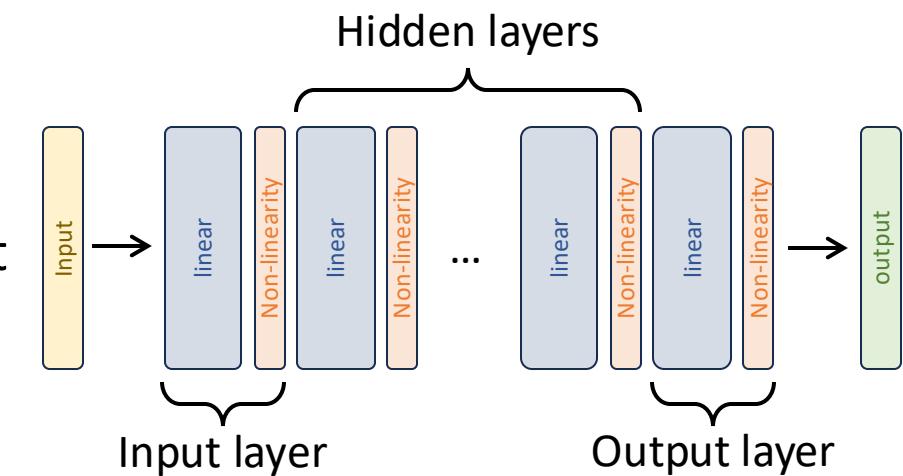
2. Introduce *non-linearities* in the model

- if $f(x) = x$ (i.e., no non-linearity is added), we get
$$z = s^T W^T x$$
- This implies:
 1. We could have used $W' = Ws$ and get the same output
 2. We wouldn't have needed a second layer!
 3. But our model is still linear
- So, we use non-linear activation functions to model more complex functions



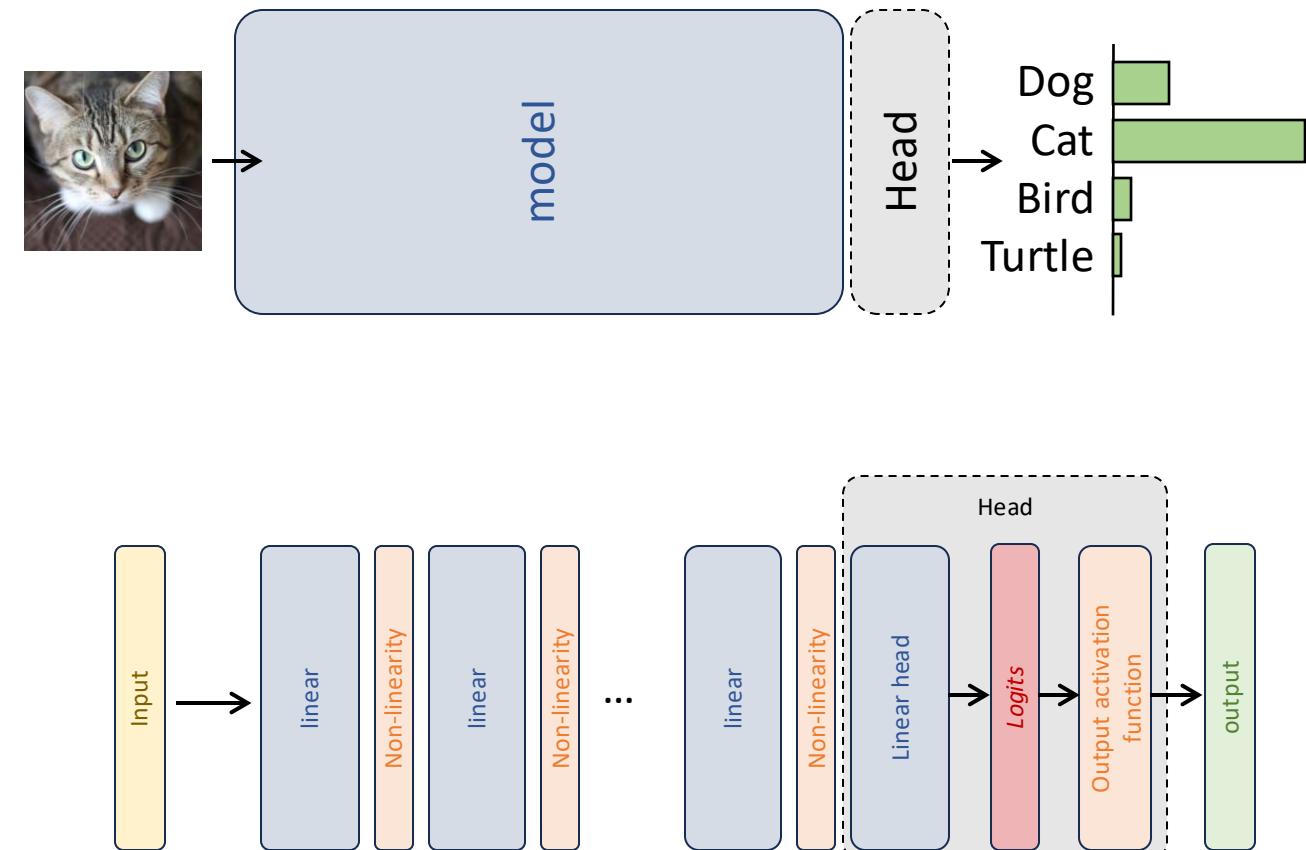
Multi-layer perceptron models

- We can stack additional *layers*
 - separated by *non-linearities* (activation functions) to prevent collapses
- *Universal Approximation Theorem* tells us that we can approximate “any” function with MLPs
 - “For any continuous function g defined on a compact subset of \mathbb{R}^n and for any $\epsilon > 0$, there exists a feedforward neural network with a single hidden layer and a finite number of neurons that can approximate g to within an arbitrary degree of accuracy ϵ ”
 - A single-layer MLP works ... but no information on the number of neurons, or the weights’ values!
 - *Deeper, narrower* networks are generally used

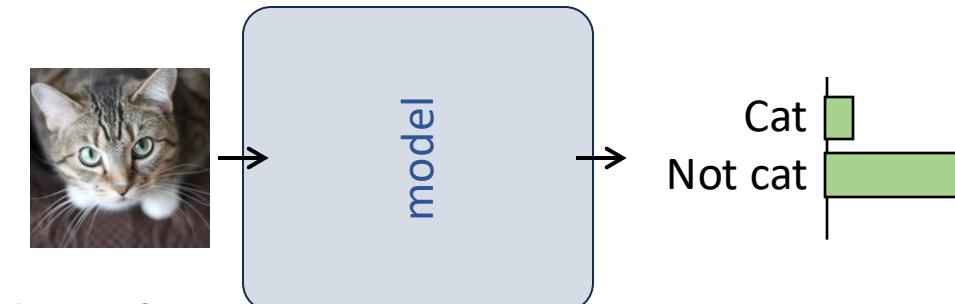


Activation functions for classification models

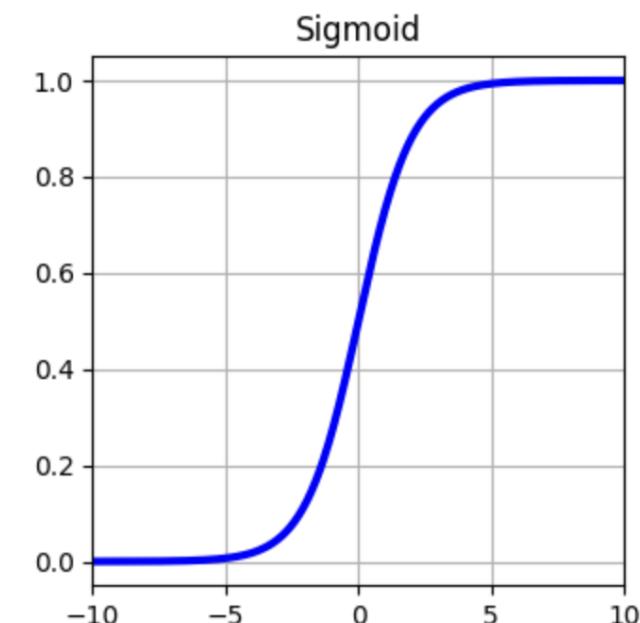
- As argued, activation functions can be used to enforce properties on the model's output
- In classification problems, the output *before* the final activation is treated as *unnormalized probabilities (logits)*
- We still need a step to convert *logits* into *valid* probabilities
 - i.e., all probabilities should sum to 1, and be in $[0, 1]$



Binary classification

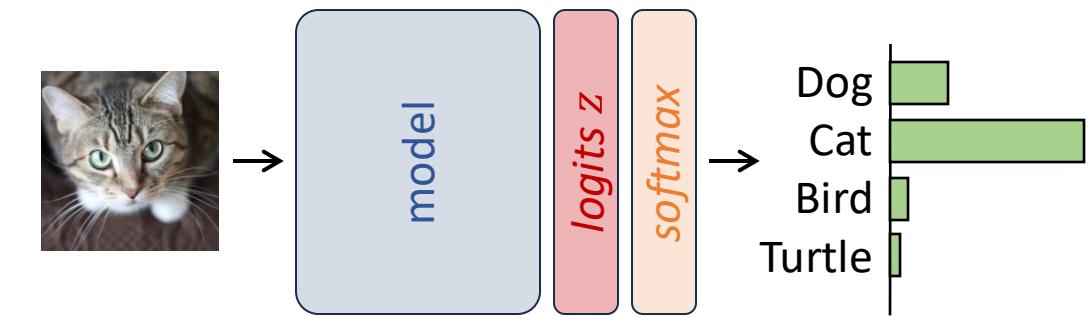


- The model predicts the probability of a single class for point x
 - As a convention, the *positive* one $P(\text{pos}|x)$ (e.g., $P(\text{cat}|x)$)
- The model produces a logit $z = \text{model}(x)$
- We use the *sigmoid function* on the output logit z
 - $\sigma(z) = \frac{1}{1+e^{-z}}$
 - This guarantees $P(\text{pos}|x) \in [0, 1]$ 
- We work out the probability of the negative class (e.g., “not cat”)
 - $P(\text{neg}|x) = 1 - P(\text{pos}|x)$
 - We can easily show that $P(\text{neg}|x) \in [0,1]$ 
- By construction, $P(\text{pos}|x) + P(\text{neg}|x) = 1$ 



Multi-class classification

- The output class is one of many (c_1, c_2, \dots, c_n)
 - E.g., (dog, cat, bird, turtle)
- The model produces a vector of n logits for a point x
 - (i.e., the last layer will have n perceptrons)
 - $z = (z_1, z_2, \dots, z_n) = \text{model}(x)$
- We need to obtain, from the logits, valid probabilities
 - $P(c_1|x), P(c_2|x), \dots, P(c_n|x)$
- The **softmax** function is applied:
 - $P(c_i|x) = \frac{e^{z_i}}{\sum_j e^{z_j}}$
- It can be easily shown that:
 - $P(c_i|x) \in [0, 1]$ 
 - $\sum_i P(c_i|x) = 1$ 



Activation functions for regression models

- In regression, models generally predict real numbers
- Typically, there is no need to enforce properties
- Output activation function can be the identity function
 - $f(x) = x$
 - Generally the only situation where it makes sense to use it!

Defining weights (parameters)

- So far, we assumed all weights and biases (let's call them θ) to be known
 - *But, we still need to figure out how we find them!*
- We pick a function (objective, or loss), $\mathcal{L}(\theta)$, that we want to minimize
 - e.g., in Linear Regression we minimize the Mean Squared Error
 - $\mathcal{L}(\theta) = MSE(\theta) = \frac{1}{n} \sum (\mathbf{y}_i - \theta^T \mathbf{x}_i)^2$
 - Then, we pick θ that minimizes it

Note

\mathcal{L} also depends on the training points $\mathbf{x}_i, \mathbf{y}_i$, so we should refer to it as $\mathcal{L}(\theta, \mathbf{X}, \mathbf{y})$.

However, the training set \mathbf{X}, \mathbf{y} is generally fixed. Thus, we only have control over θ , so we use the notation $\mathcal{L}(\theta)$.

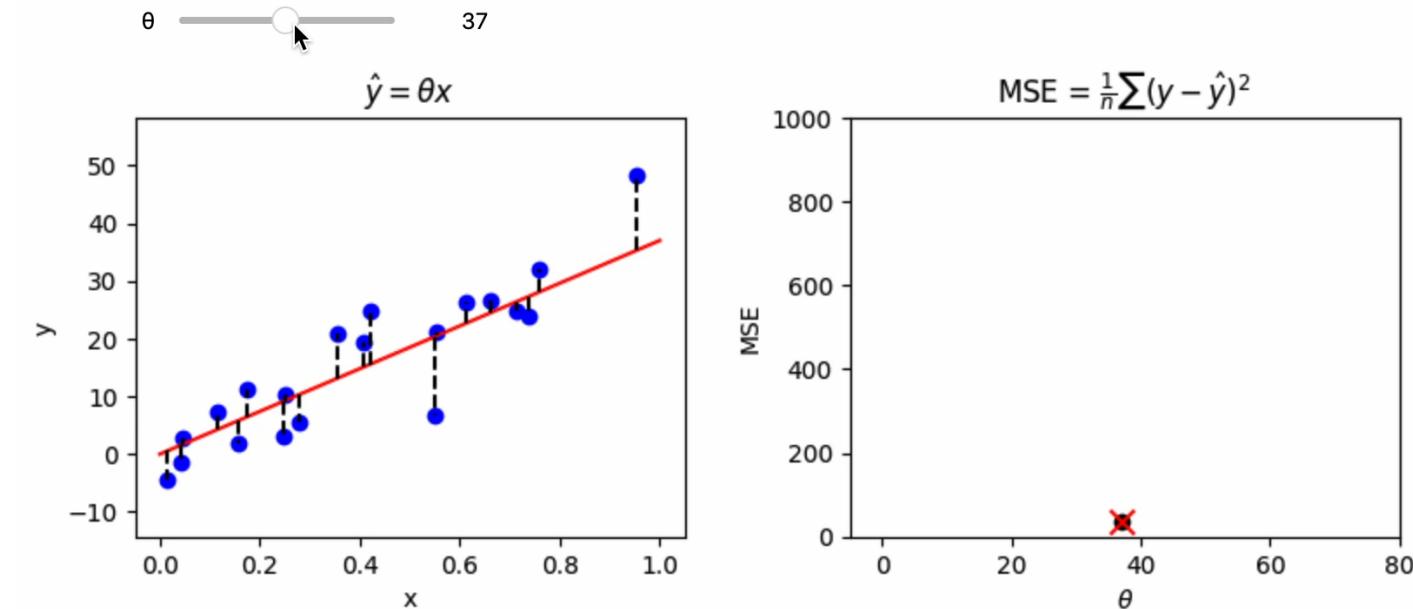
Linear regression

- For *simple* models, we can find the optimal weights in closed form
 - $\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{\partial MSE(\theta)}{\partial \theta} = 0$
 - Quadratic in θ , can be solved easily!
- Or, we can evaluate the loss function for a bunch of θ 's, and find the “best” one

Note

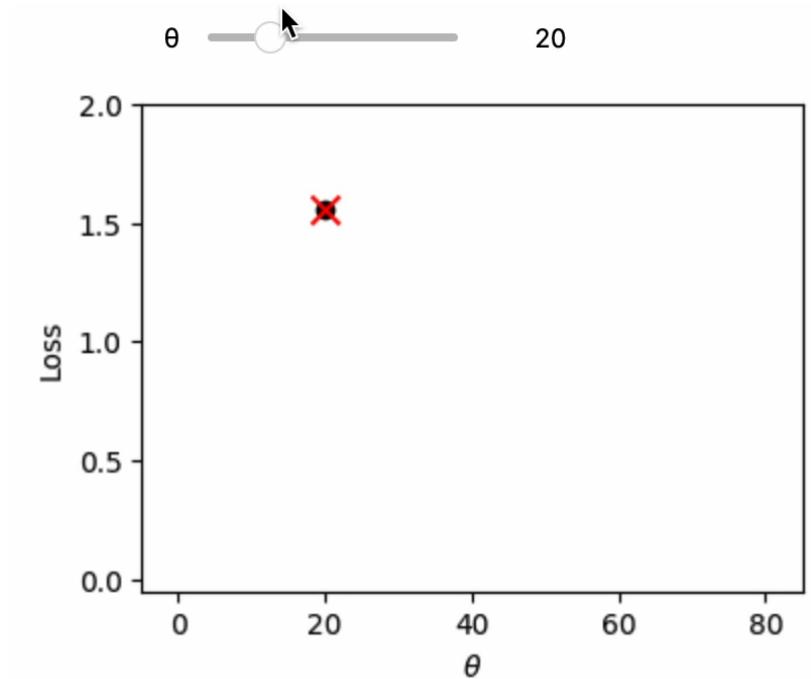
For linear regression, we don't try a bunch of θ since we can easily find the best value in closed form.

However, this provides the intuition for what we will do next with more complex loss functions/models.



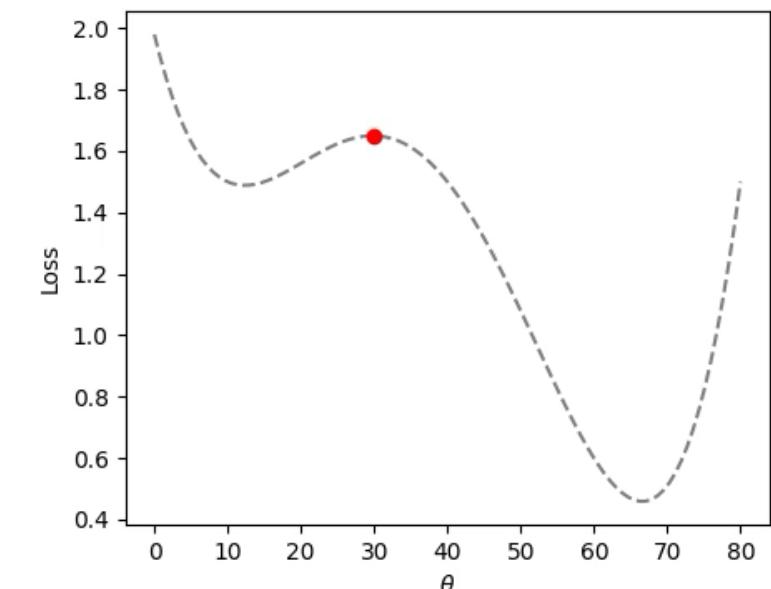
More complex losses/models

- For *more complex* loss functions/models, we may not be able to solve the problem in closed form
 - But we can evaluate $\mathcal{L}(\theta)$ for various values of θ
- We can iteratively update θ to reach a local minimum:
 - We start from a random value θ , then
 - we “move around” according to “some policy”



We “move around” according to “some policy”

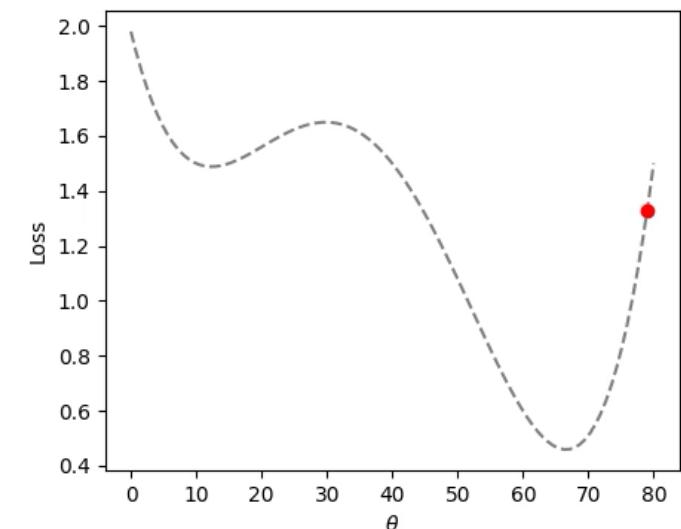
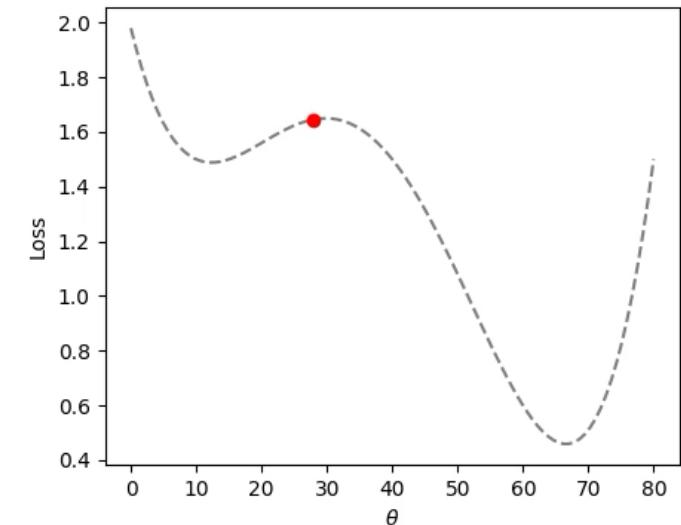
- Move around = update θ incrementally, based on its current value
 - The new value of θ at any step depends on the previous step’s value
 - $\theta_{t+1} := \theta_t + update$
- Some policy = we take a small step in the direction where the function decreases locally
 - i.e. in the *opposite* direction of the gradient
 - $\theta_{t+1} := \theta_t - \alpha \nabla_{\theta} \mathcal{L}(\theta_t)$
 - for 1-dimensional θ , we have $\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$
 - α : learning rate, controls the “size” of the step
- Gradient Descent!



Some limitations of GD

- GD is sensitive to weight initialization
 - Different initializations can lead to different solutions!
 - GD can get stuck in local minima
- Various solutions to help prevent local minima:
 - Adding momentum
 - Adaptive learning rates
 - Learning rate schedules

Note
Different initializations will lead to the global minimum for convex loss functions. However, that represents a trivial situation we typically do not encounter.



Backpropagation

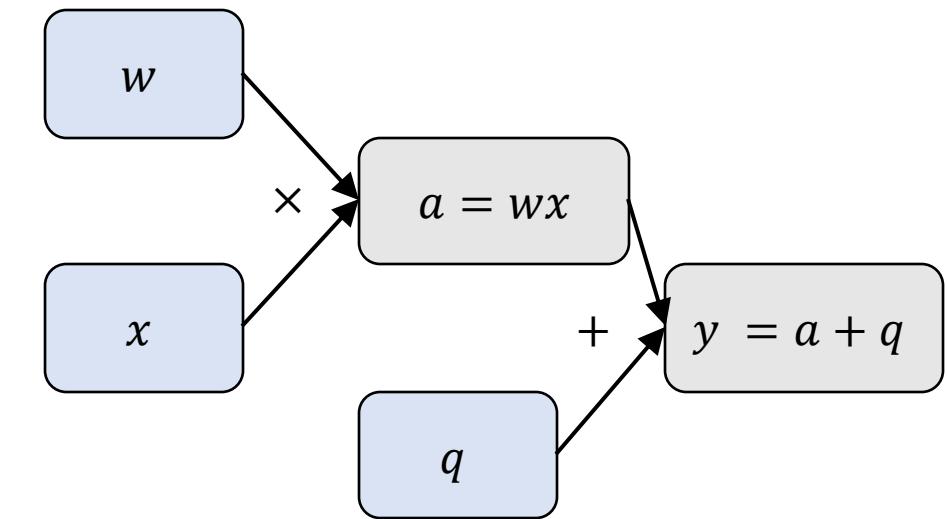
- So far, we assumed we were able to compute $\nabla_{\theta} \mathcal{L}(\theta)$
- However, any loss/model combination would need a different gradient computation!
- We can use backpropagation to compute the gradient of the loss w.r.t. any weight!
 - Backpropagation is just a fancy word for “using the chain rule”

Using the chain rule

- We use the chain rule from calculus, $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$
 - Sometimes known as $f(g(x))' = f'(g(x)) \cdot g'(x)$
- And apply it from the end of the computational graph, backwards
 - (hence the name, backpropagation)

Computational graph

- A computational graph is a directed graph
 - Each node corresponds to an operation
 - Each edge represents the flow of data between nodes
- For instance, we may want to compute $y = wx + q$
 - We start from three variables, w , x and y
 - The computational graph performs one operation at a time
 - First, compute the intermediate variable $a = wx$
 - Then, compute the output variable $z = a + y = wx + y$

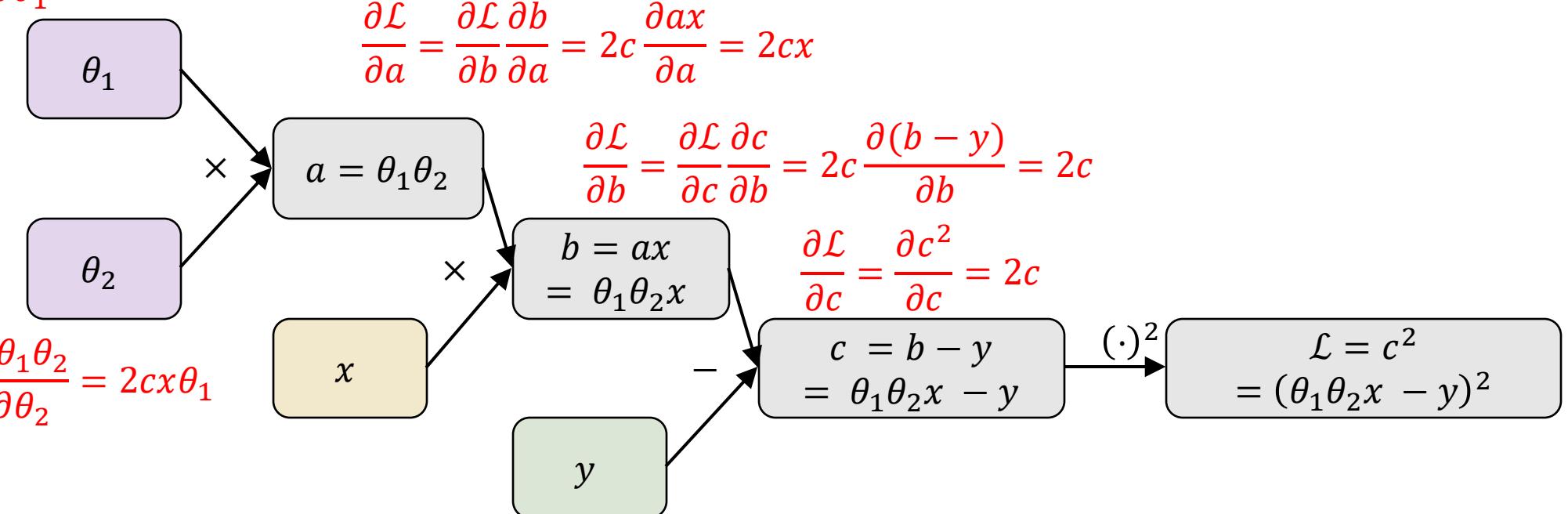


Backpropagation example

- Let's say:
 - Our dataset has one point, (x, y)
 - Our (weird) model has two parameters, θ_1 and θ_2 , and predicts $\theta_1 \theta_2 x$
 - Our loss function will be $\mathcal{L} = (\theta_1 \theta_2 x - y)^2$
- We build a computational graph with all operations and intermediate variables
 - $a = \theta_1 \theta_2$
 - $b = ax = \theta_1 \theta_2 x$
 - $c = b - y = ax - y = \theta_1 \theta_2 x - y$
 - $\mathcal{L} = c^2 = (b - y)^2 = (ax - y)^2 = (\theta_1 \theta_2 x - y)^2$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial a} \frac{\partial a}{\partial \theta_1} = 2cx \frac{\partial \theta_1 \theta_2}{\partial \theta_1} = 2cx \theta_2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial a} \frac{\partial a}{\partial \theta_2} = 2cx \frac{\partial \theta_1 \theta_2}{\partial \theta_2} = 2cx \theta_1$$



Forward step

- The loss \mathcal{L} is computed starting from the “inputs” θ_1, θ_2, x, y

Backward step (backpropagation)

- The loss \mathcal{L} is used to compute the derivative w.r.t. $c \rightarrow \frac{\partial \mathcal{L}}{\partial c}$
- The derivative $\frac{\partial \mathcal{L}}{\partial c}$ is used to compute the derivative w.r.t. $b \rightarrow \frac{\partial \mathcal{L}}{\partial b}$
- The derivative $\frac{\partial \mathcal{L}}{\partial b}$ is used to compute the derivative w.r.t. $a \rightarrow \frac{\partial \mathcal{L}}{\partial a}$
- The derivative $\frac{\partial \mathcal{L}}{\partial a}$ is used to compute the derivative w.r.t. $\theta_1, \theta_2 \rightarrow \frac{\partial \mathcal{L}}{\partial \theta_1}, \frac{\partial \mathcal{L}}{\partial \theta_2}$

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = 2(\theta_1 \theta_2 x - y)x\theta_2$$

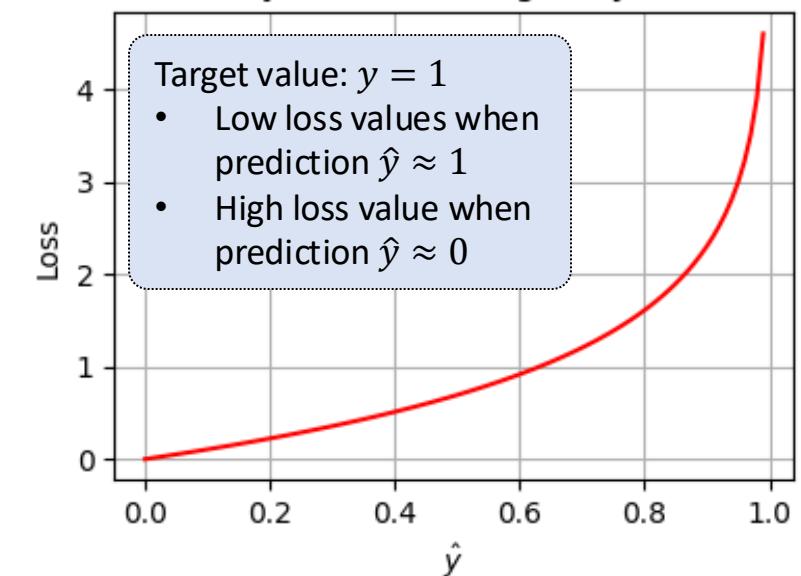
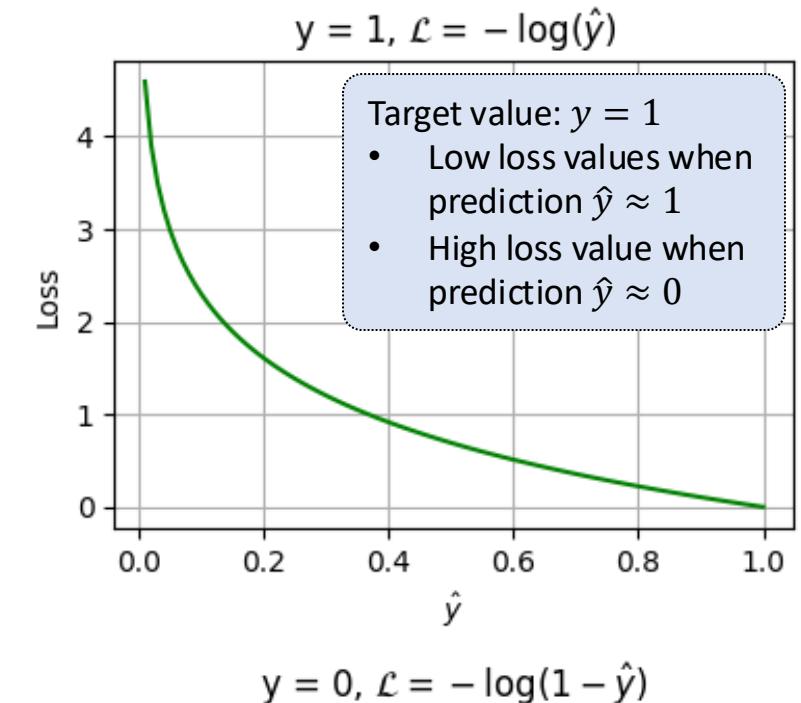
$$\frac{\partial \mathcal{L}}{\partial \theta_2} = 2(\theta_1 \theta_2 x - y)x\theta_1$$

Loss functions

- Regression
 - *Mean Squared Error, Mean Absolute Error*
- Binary
 - *Binary Cross-Entropy (BCE)*
 - $y = \{0, 1\} \rightarrow$ ground truth
 - $\hat{y} = \text{model}(x) \in [0, 1] \rightarrow$ predicted value

$$\mathcal{L} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- y (ground truth) acts as a “selector” of the loss term to be applied
 - $y = 1 \rightarrow \mathcal{L} = -\log(\hat{y})$
 - $y = 0 \rightarrow \mathcal{L} = -\log(1 - \hat{y})$



Loss functions

- Multi-class classification
 - Cross-Entropy
 - Generalization to multiple classes of BCE
 - $y_i = 1$ when ground truth is the i th class, 0 otherwise
 - y_i plays the same “selector” mechanism as in BCE

$$\mathcal{L} = - \sum_i y_i \log(\hat{y}_i)$$

Association Rules Fundamentals



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Silvia Chiusano
Politecnico di Torino



Association rules

- Objective
 - extraction of frequent correlations or pattern from a transactional database

Tickets at a supermarket counter

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk
...	...

- Association rule

$\text{diapers} \Rightarrow \text{beer}$

- 2% of transactions contains both items
- 30% of transactions containing diapers also contains beer



Association rule mining

- A collection of transactions is given
 - a transaction is a set of items
 - items in a transaction are
not ordered
- Association rule
 - $A, B \Rightarrow C$
 - A, B = items in the rule body
 - C = item in the rule head
- The \Rightarrow means co-occurrence
 - *not* causality
- Example
 - coke, diapers \Rightarrow milk

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk
...	...



Transactional formats

- Association rule extraction is an *exploratory technique* that can be applied to any data type
- A transaction can be any set of items
 - Market basket data
 - Textual data
 - Structured data
 - ...



Transactional formats

- Textual data
 - A document is a transaction
 - Words in a document are items in the transaction
- Data example
 - Doc1: algorithm analysis customer data mining relationship
 - Doc2: customer data management relationship
 - Doc3: analysis customer data mining relationship social
- Rule example

customer, relationship \Rightarrow data, mining





Transactional formats

■ Structured data

- A table row is a transaction
- Pairs (attribute, value) are items in the transaction

■ Data example

Refund	Marital Status	Taxable Income	Cheat
No	Married	< 80K	No



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Transaction

Refund=no, MaritalStatus=married, TaxableIncome<80K, Cheat=No

■ Rule example

Refund=No, MaritalStatus=Married \Rightarrow Cheat = No



Definitions

- **Itemset** is a set including one or more items
 - Example: {Beer, Diapers}
- **k-itemset** is an itemset that contains k items
- **Support count (#)** is the frequency of occurrence of an itemset
 - Example: #{{Beer,Diapers}} = 2
- **Support** is the fraction of transactions that contain an itemset
 - Example: sup({Beer, Diapers}) = 2/5
- **Frequent itemset** is an itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk



Rule quality metrics

- Given the association rule

$$A \Rightarrow B$$

- A, B are itemsets
- Support** is the fraction of transactions containing both A and B

$$\frac{\#\{A,B\}}{|T|}$$

- |T| is the cardinality of the transactional database
- a priori probability of itemset AB
- rule frequency in the database
- Confidence** is the frequency of B in transactions containing A

$$\frac{\text{sup}(A,B)}{\text{sup}(A)}$$

- conditional probability of finding B having found A
- “strength” of the “ \Rightarrow ”



Rule quality metrics: example

- From itemset {Milk, Diapers} the following rules may be derived
- Rule: Milk \Rightarrow Diapers
 - support
 $sup = \#\{\text{Milk, Diapers}\} / \#\text{trans.} = 3/5 = 60\%$
 - confidence
 $conf = \#\{\text{Milk, Diapers}\} / \#\{\text{Milk}\} = 3/4 = 75\%$
- Rule: Diapers \Rightarrow Milk
 - same support
 $s=60\%$
 - confidence
 $conf = \#\{\text{Milk, Diapers}\} / \#\{\text{Diapers}\} = 3/3 = 100\%$

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Coke, Diapers, Milk



Association rule extraction

- Given a set of transactions T , association rule mining is the extraction of the rules satisfying the constraints
 - support $\geq \text{minsup}$ threshold
 - confidence $\geq \text{minconf}$ threshold
- The result is
 - complete (*all* rules satisfying both constraints)
 - correct (*only* the rules satisfying both constraints)
- May add other more complex constraints



Association rule extraction

- Brute-force approach
 - enumerate all possible permutations (i.e., association rules)
 - compute support and confidence for each rule
 - prune the rules that do not satisfy the *minsup* and *minconf* constraints
- Computationally *unfeasible*
- Given an itemset, the extraction process may be split
 - first generate frequent itemsets
 - next generate rules from each frequent itemset
- Example
 - Itemset
 $\{\text{Milk, Diapers}\}$ sup=60%
 - Rules
 $\text{Milk} \Rightarrow \text{Diapers}$ (conf=75%)
 $\text{Diapers} \Rightarrow \text{Milk}$ (conf=100%)



Association rule extraction

(1) Extraction of frequent itemsets

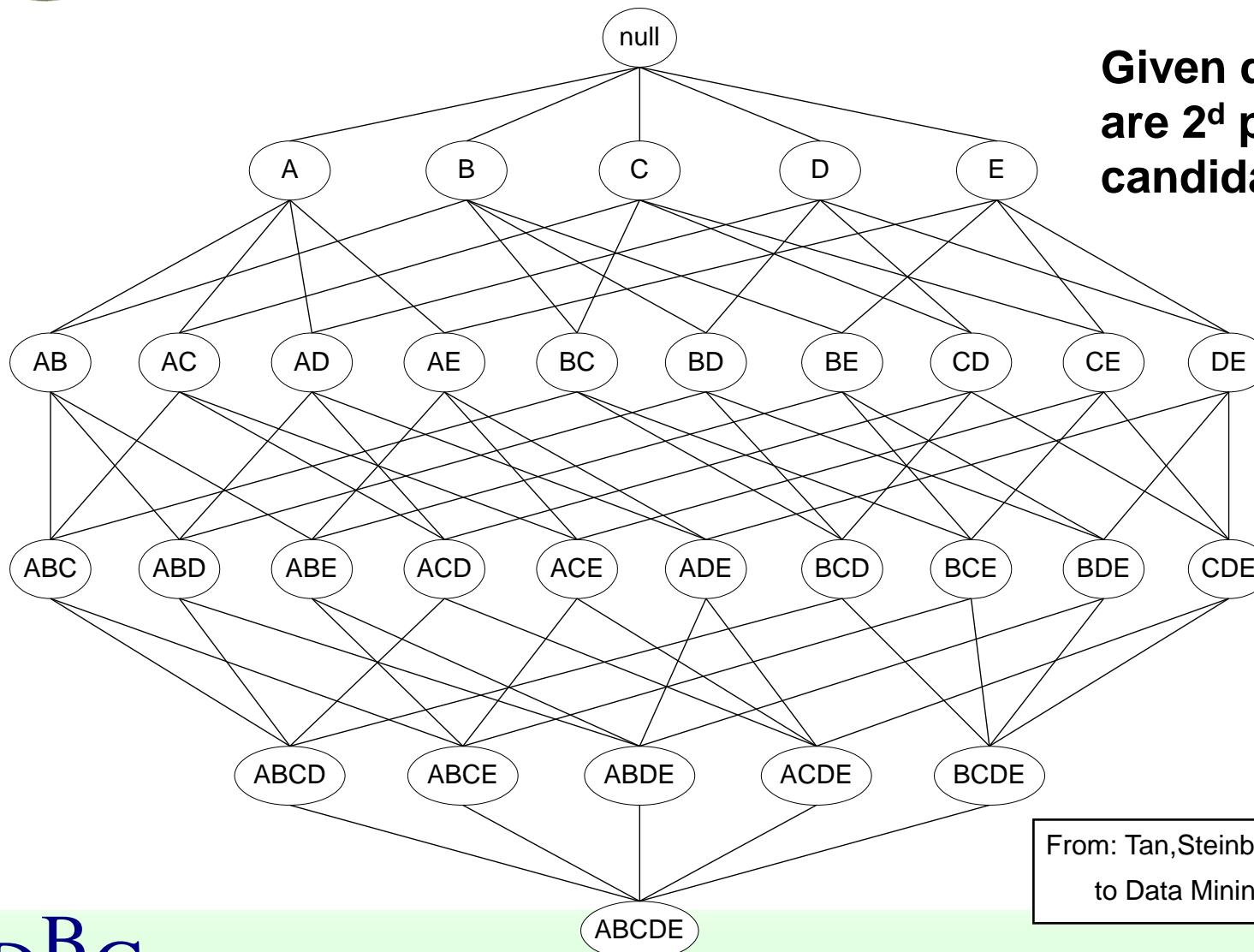
- many different techniques
 - level-wise approaches (Apriori, ...)
 - approaches without candidate generation (FP-growth, ...)
 - other approaches
- most computationally expensive step
 - limit extraction time by means of support threshold

(2) Extraction of association rules

- generation of all possible binary partitioning of each frequent itemset
 - possibly enforcing a confidence threshold



Frequent Itemset Generation



Given d items, there
are 2^d possible
candidate itemsets

From: Tan, Steinbach, Kumar, Introduction
to Data Mining, McGraw Hill 2006



Frequent Itemset Generation

■ Brute-force approach

- each itemset in the lattice is a *candidate* frequent itemset
- scan the database to count the support of each candidate
 - match each transaction against every candidate
- Complexity $\sim O(|T| 2^d w)$
 - $|T|$ is number of transactions
 - d is number of items
 - w is transaction length



Improving Efficiency

- Reduce the number of candidates
 - Prune the search space
 - complete set of candidates is 2^d
- Reduce the number of transactions
 - Prune transactions as the size of itemsets increases
 - reduce $|T|$
- Reduce the number of comparisons
 - Equal to $|T| \cdot 2^d$
 - Use efficient data structures to store the candidates or transactions



The Apriori Principle

"If an itemset is frequent, then all of its subsets must also be frequent"

- The support of an itemset can never exceed the support of any of its subsets
- It holds due to the antimonotone property of the support measure
 - Given two arbitrary itemsets A and B
if $A \subseteq B$ then $\text{sup}(A) \geq \text{sup}(B)$
- It reduces the number of candidates

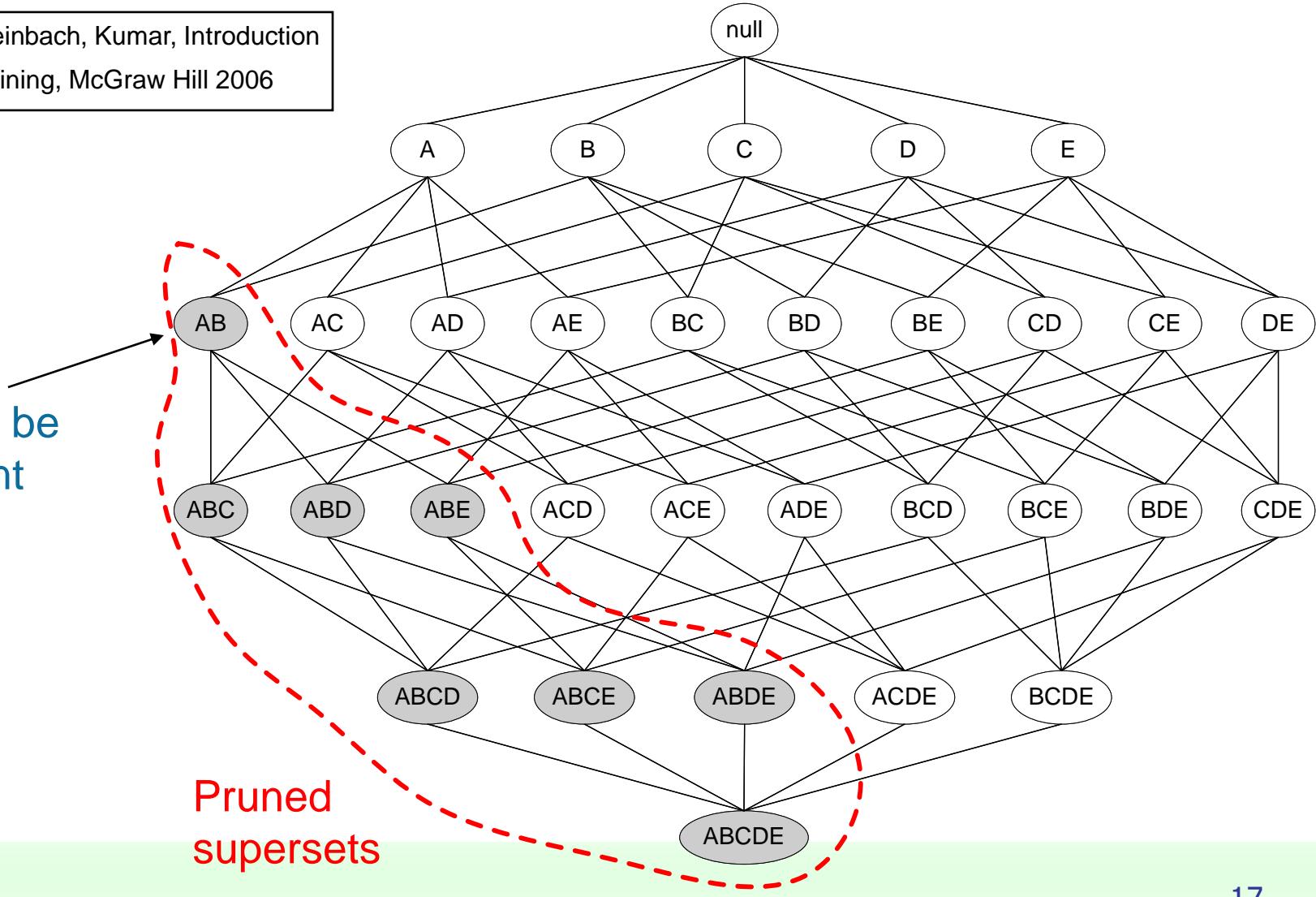


The Apriori Principle

From: Tan, Steinbach, Kumar, Introduction
to Data Mining, McGraw Hill 2006

Found to be
Infrequent

Pruned
supersets





Apriori Algorithm [Agr94]

- Level-based approach
 - at each iteration extracts itemsets of a given length k
- Two main steps for each level
 - (1) Candidate generation
 - Join Step
 - generate candidates of length $k+1$ by joining frequent itemsets of length k
 - Prune Step
 - apply Apriori principle: prune length $k+1$ candidate itemsets that contain at least one k-itemset that is not frequent
 - (2) Frequent itemset generation
 - scan DB to count support for $k+1$ candidates
 - prune candidates below minsup



Apriori Algorithm [Agr94]

■ Pseudo-code

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1$; $L_k \neq \emptyset$; $k++$) **do**

begin

C_{k+1} = candidates generated from L_k ;

for each transaction t in database **do**

increment the count of all candidates in C_{k+1}

that are contained in t

L_{k+1} = candidates in C_{k+1} satisfying minsup

end

return $\cup_k L_k$;



Generating Candidates

- Sort L_k candidates in lexicographical order
- For each candidate of length k
 - Self-join with each candidate sharing same L_{k-1} prefix
 - Prune candidates by applying Apriori principle
- Example: given $L_3 = \{abc, abd, acd, ace, bcd\}$
 - Self-join
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
 - Prune by applying Apriori principle
 - $acde$ is removed because ade, cde are not in L_3
 - $C_4 = \{abcd\}$



Apriori Algorithm: Example

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

$\text{minsup} > 1$



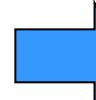
Generate candidate 1-itemsets

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

C_1

1st DB
scan



itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

$\text{minsup} > 1$

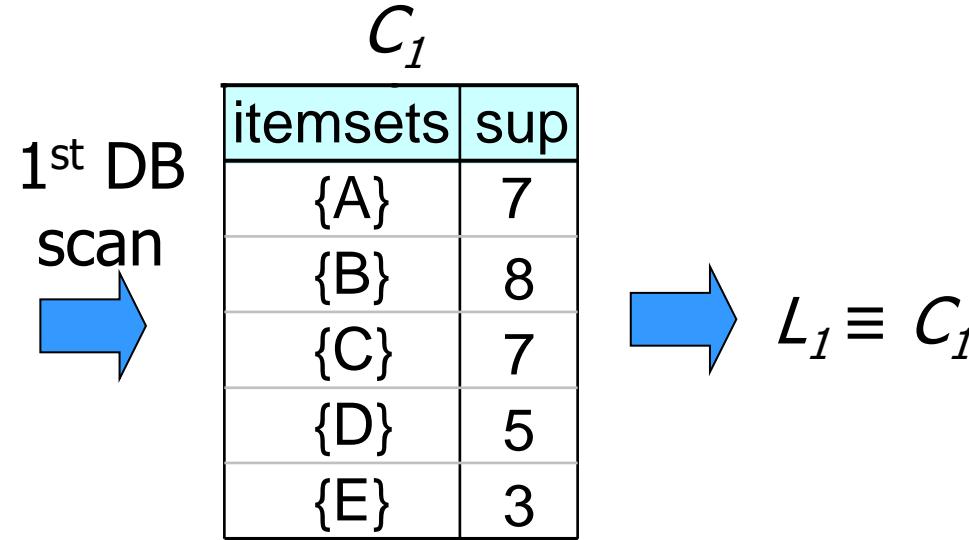


Prune infrequent candidates in C_1

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

$\text{minsup} > 1$



- All itemsets in set C_1 are frequent according to $\text{minsup} > 1$



Generate candidates from L_1

L_1	
itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

C_2
itemsets
{A,B}
{A,C}
{A,D}
{A,E}
{B,C}
{B,D}
{B,E}
{C,D}
{C,E}
{D,E}



Count support for candidates in C_2

L_1	
itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

C_2	
itemsets	
{A,B}	
{A,C}	
{A,D}	
{A,E}	
{B,C}	
{B,D}	
{B,E}	
{C,D}	
{C,E}	
{D,E}	

2nd
DB
scan

C_2	
itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{B,E}	1
{C,D}	3
{C,E}	2
{D,E}	2



Prune infrequent candidates in C_2

L_1	
itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

C_2	
itemsets	
{A,B}	
{A,C}	
{A,D}	
{A,E}	
{B,C}	
{B,D}	
{B,E}	
{C,D}	
{C,E}	
{D,E}	

2nd
DB
scan

C_2	
itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{B,E}	1
{C,D}	3
{C,E}	2
{D,E}	2

L_2	
itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2



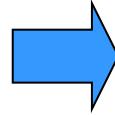
Generate candidates from L_2

L_2

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

C_3

itemsets
{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{C,D,E}





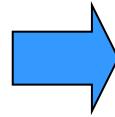
Apply Apriori principle on C_3

L_2

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

C_3

itemsets
{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{C,D,E}



- Prune {A,B,E}
 - Its subset {B,E} is infrequent ({B,E} is not in L_2)



Count support for candidates in C_3

L_2

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

C_3

itemsets
{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{C,D,E}

3rd
DB
scan

C_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,C,E}	1
{A,D,E}	2
{B,C,D}	2
{C,D,E}	1



Prune infrequent candidates in C_3

L_2

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

C_3

itemsets
{A,B,C}
{A,B,D}
{A,B,E}
{A,C,D}
{A,C,E}
{A,D,E}
{B,C,D}
{C,D,E}

3rd
DB
scan

C_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,C,E}	1
{A,D,E}	2
{B,C,D}	2
{C,D,E}	1

L_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

- {A,C,E} and {C,D,E} are actually infrequent
 - They are discarded from C_3



Generate candidates from L_3

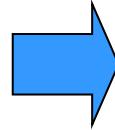
L_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

C_4

itemsets

{A,B,C,D}





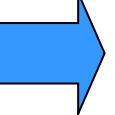
Apply Apriori principle on C_4

itemsets	sup	L_3
{A,B,C}	3	
{A,B,D}	2	
{A,C,D}	2	
{A,D,E}	2	
{B,C,D}	2	

C_4

itemsets

{A,B,C,D}



- Check if $\{A,C,D\}$ and $\{B,C,D\}$ belong to L_3
 - L_3 contains all 3-itemset subsets of $\{A,B,C,D\}$
 - $\{A,B,C,D\}$ is potentially frequent



Count support for candidates in C_4

L_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

C_4

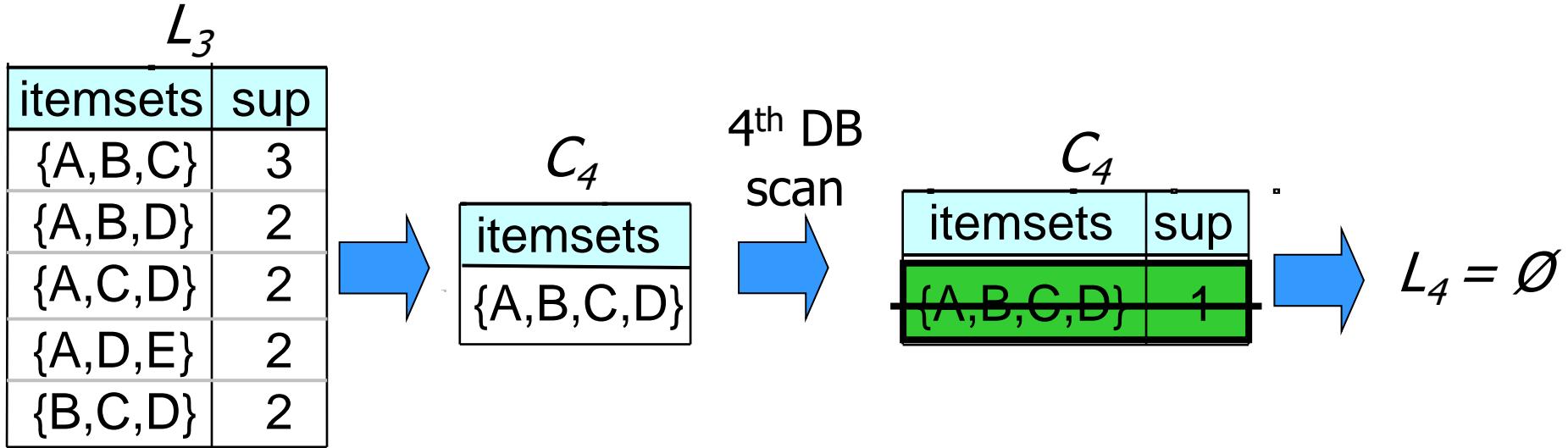
itemsets
{A,B,C,D}

4th DB
scan

itemsets	sup
{A,B,C,D}	1



Prune infrequent candidates in C_4



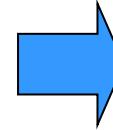
- {A,B,C,D} is actually infrequent
 - {A,B,C,D} is discarded from C_4



Final set of frequent itemsets

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



L_1

itemsets	sup
{A}	7
{B}	8
{C}	7
{D}	5
{E}	3

L_2

itemsets	sup
{A,B}	5
{A,C}	4
{A,D}	4
{A,E}	2
{B,C}	6
{B,D}	3
{C,D}	3
{C,E}	2
{D,E}	2

L_3

itemsets	sup
{A,B,C}	3
{A,B,D}	2
{A,C,D}	2
{A,D,E}	2
{B,C,D}	2

minsup>1

D^B_MG



Counting Support of Candidates

- Scan transaction database to count support of each itemset
 - total number of candidates may be large
 - one transaction may contain many candidates
- Approach [Agr94]
 - candidate itemsets are stored in a *hash-tree*
 - *leaf* node of hash-tree contains a list of itemsets and counts
 - *interior* node contains a hash table
 - subset function finds all candidates contained in a transaction
 - match transaction subsets to candidates in hash tree



Performance Issues in Apriori

- Candidate generation
 - Candidate sets may be huge
 - 2-itemset candidate generation is the most critical step
 - extracting long frequent itemsets requires generating all frequent subsets
- Multiple database scans
 - $n + 1$ scans when longest frequent pattern length is n



Factors Affecting Performance

- Minimum support threshold
 - lower support threshold increases number of frequent itemsets
 - larger number of candidates
 - larger (max) length of frequent itemsets
- Dimensionality (number of items) of the data set
 - more space is needed to store support count of each item
 - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
 - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
 - transaction width increases in dense data sets
 - may increase max length of frequent itemsets and traversals of hash tree
 - number of subsets in a transaction increases with its width



Improving Apriori Efficiency

- Hash-based itemset counting [Yu95]
 - A k -itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction [Yu95]
 - A transaction that does not contain any frequent k -itemset is useless in subsequent scans
- Partitioning [Sav96]
 - Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB



Improving Apriori Efficiency

- Sampling [Toi96]
 - mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic Itemset Counting [Motw98]
 - add new candidate itemsets only when all of their subsets are estimated to be frequent



FP-growth Algorithm [Han00]

- Exploits a main memory compressed representation of the database, the FP-tree
 - high compression for dense data distributions
 - less so for sparse data distributions
 - complete representation for frequent pattern mining
 - enforces support constraint
- Frequent pattern mining by means of FP-growth
 - recursive visit of FP-tree
 - applies divide-and-conquer approach
 - decomposes mining task into smaller subtasks
- Only two database scans
 - count item supports + build FP-tree



FP-tree construction

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

$\text{minsup} > 1$

- (1) Count item support and prune items below minsup threshold
- (2) Build Header Table by sorting items in decreasing support order

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3



FP-tree construction

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

$\text{minsup} > 1$

- (1) Count item support and prune items below minsup threshold
- (2) Build Header Table by sorting items in decreasing support order
- (3) Create FP-tree

For each transaction t in DB

- order transaction t items in decreasing support order
 - same order as Header Table
- insert transaction t in FP-tree
 - use existing path for common prefix
 - create new branch when path becomes different

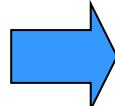


FP-tree construction

Transaction

TID	Items
1	{A,B}

Sorted transaction

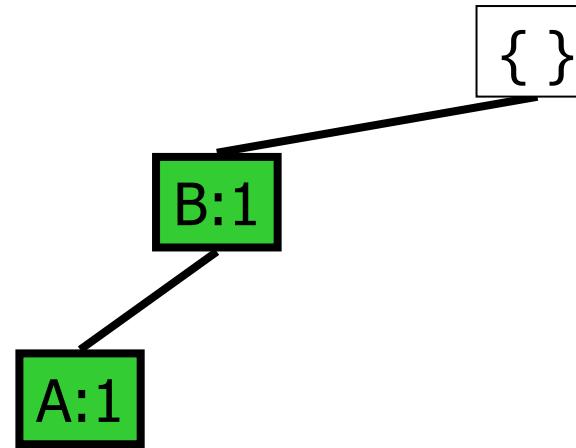


TID	Items
1	{B,A}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



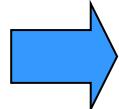


FP-tree construction

Transaction

TID	Items
2	{B,C,D}

Sorted transaction

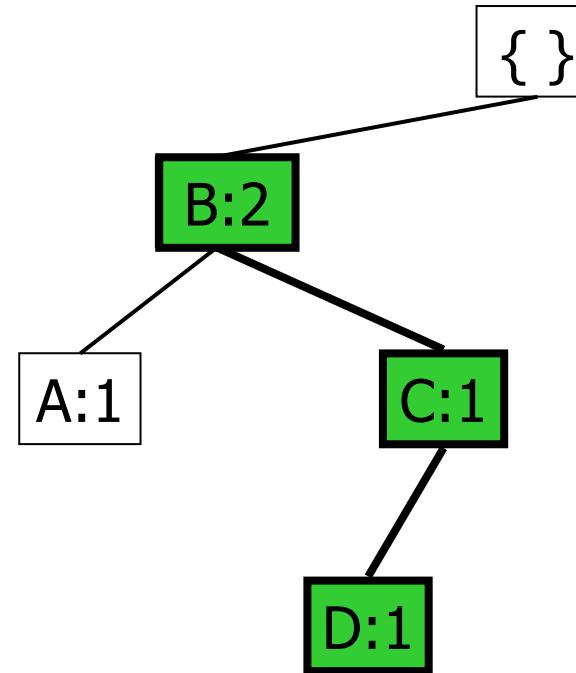


TID	Items
2	{B,C,D}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



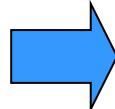


FP-tree construction

Transaction

TID	Items
3	{A,C,D,E}

Sorted transaction



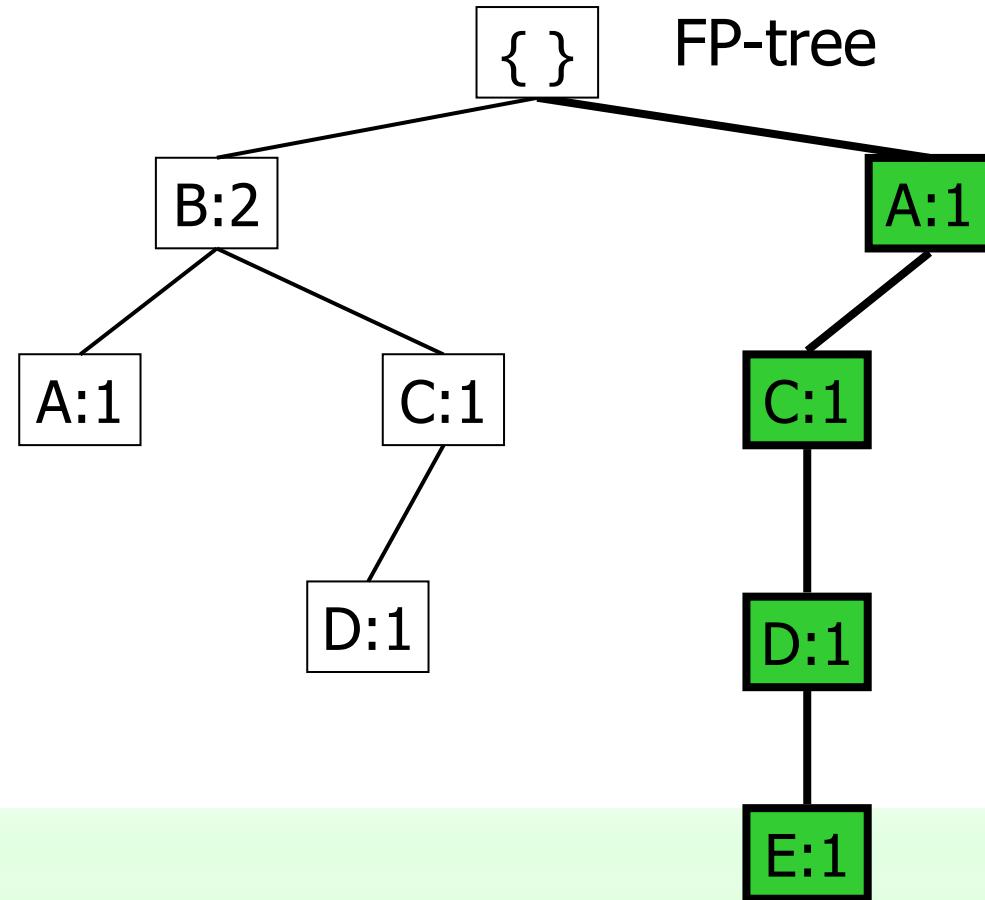
TID	Items
3	{A,C,D,E}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

{ }

FP-tree



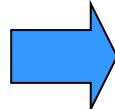


FP-tree construction

Transaction

TID	Items
4	{A,D,E}

Sorted transaction



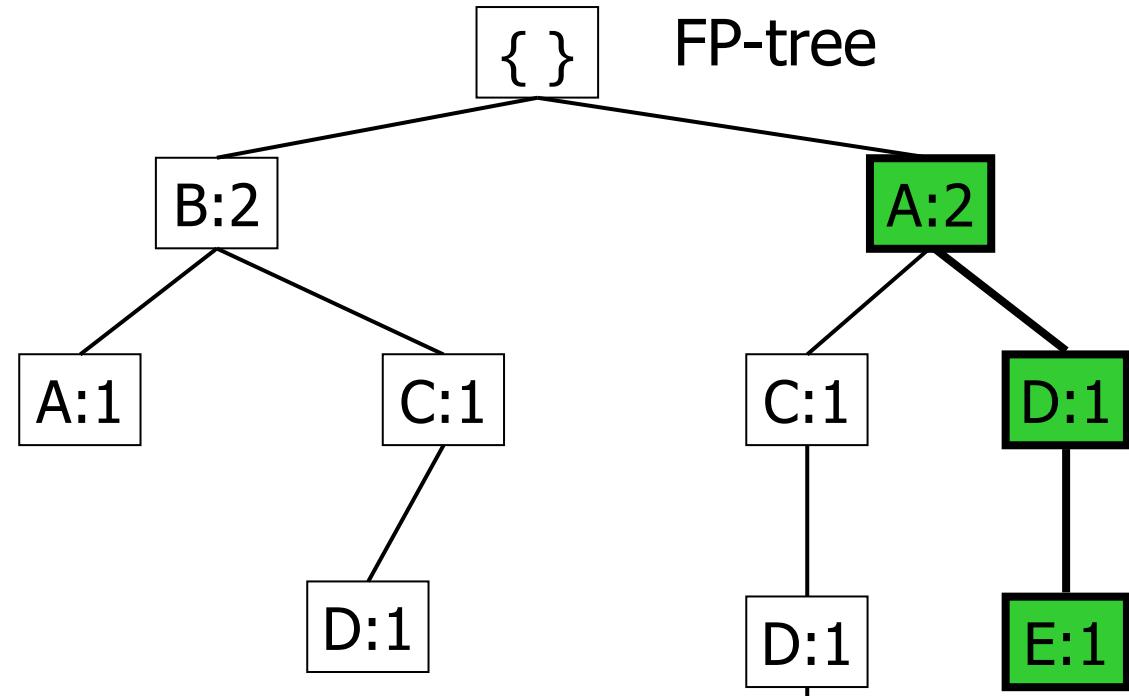
TID	Items
4	{A,D,E}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

{ }

FP-tree



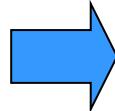


FP-tree construction

Transaction

TID	Items
5	{A,B,C}

Sorted transaction

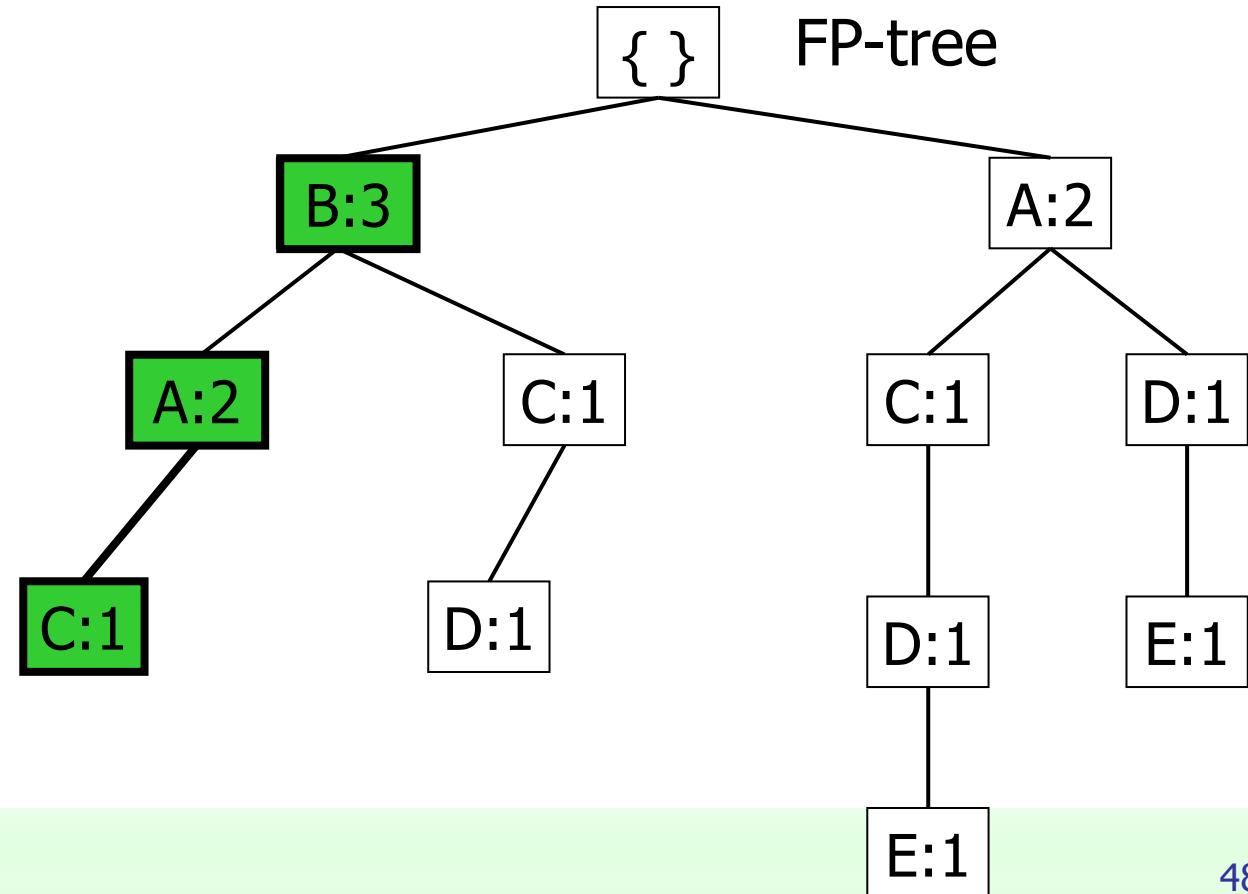


TID	Items
5	{B,A,C}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



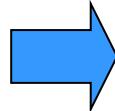


FP-tree construction

Transaction

TID	Items
6	{A,B,C,D}

Sorted transaction

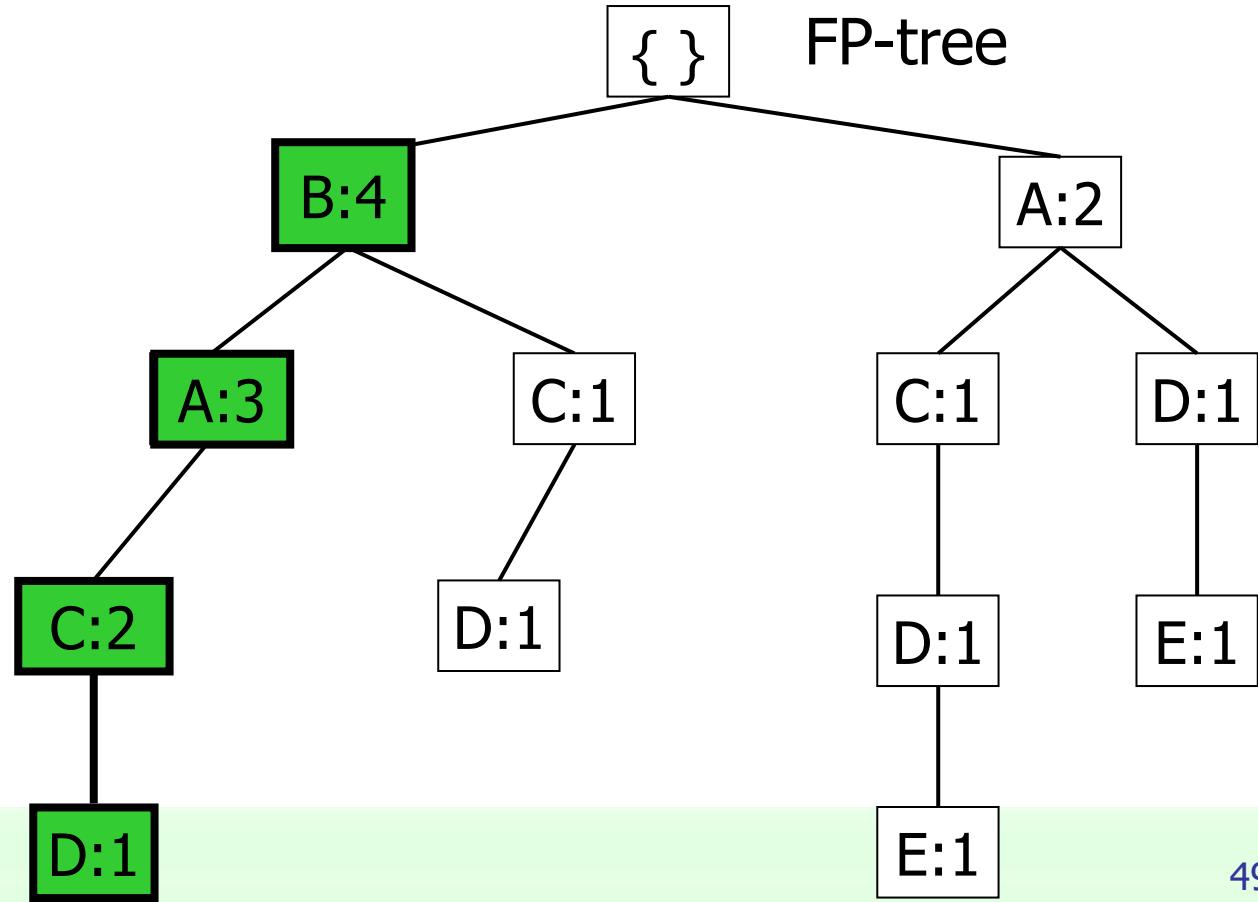


TID	Items
6	{B,A,C,D}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



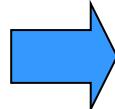


FP-tree construction

Transaction

TID	Items
7	{B,C}

Sorted transaction

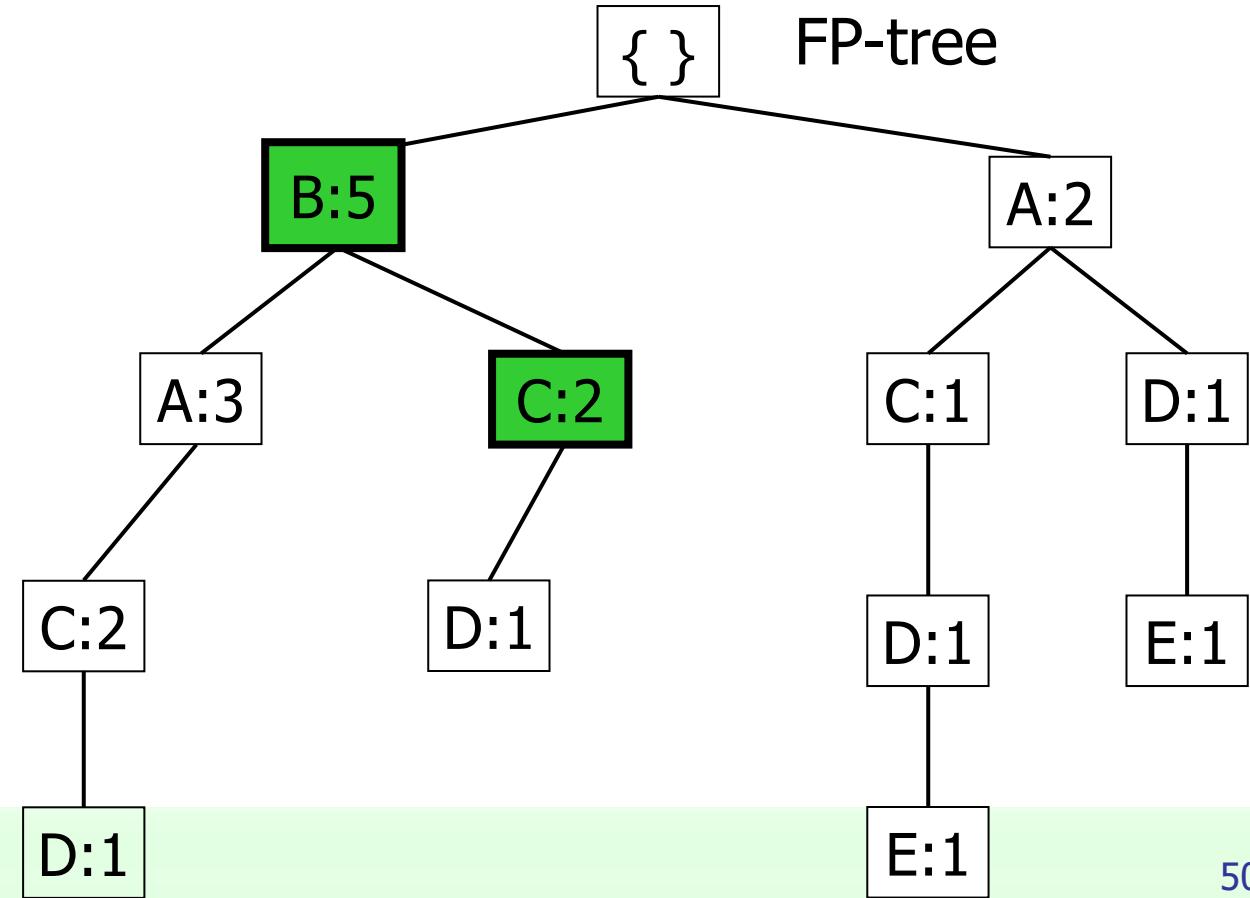


TID	Items
7	{B,C}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



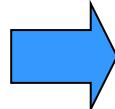


FP-tree construction

Transaction

TID	Items
8	{A,B,C}

Sorted transaction

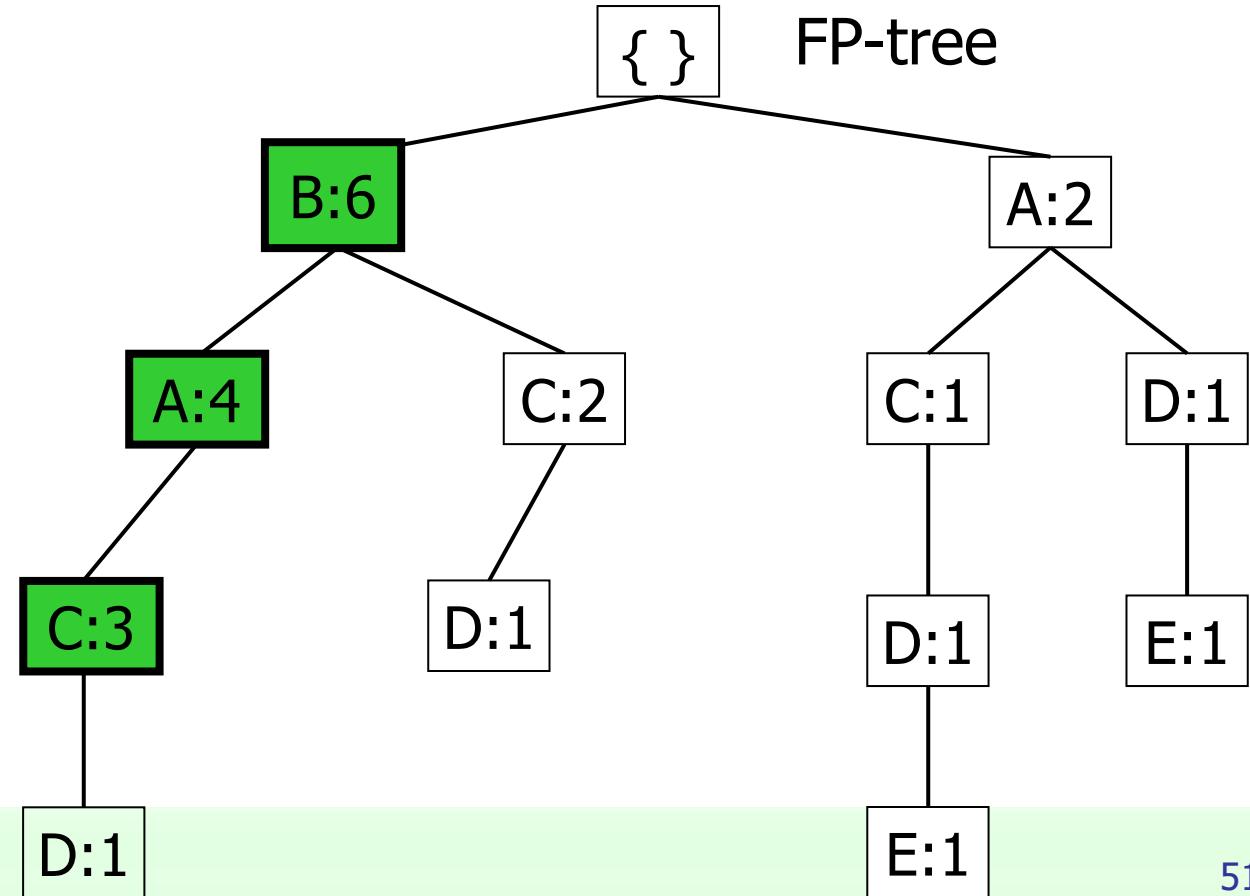


TID	Items
8	{B,A,C}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



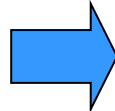


FP-tree construction

Transaction

TID	Items
9	{A,B,D}

Sorted transaction

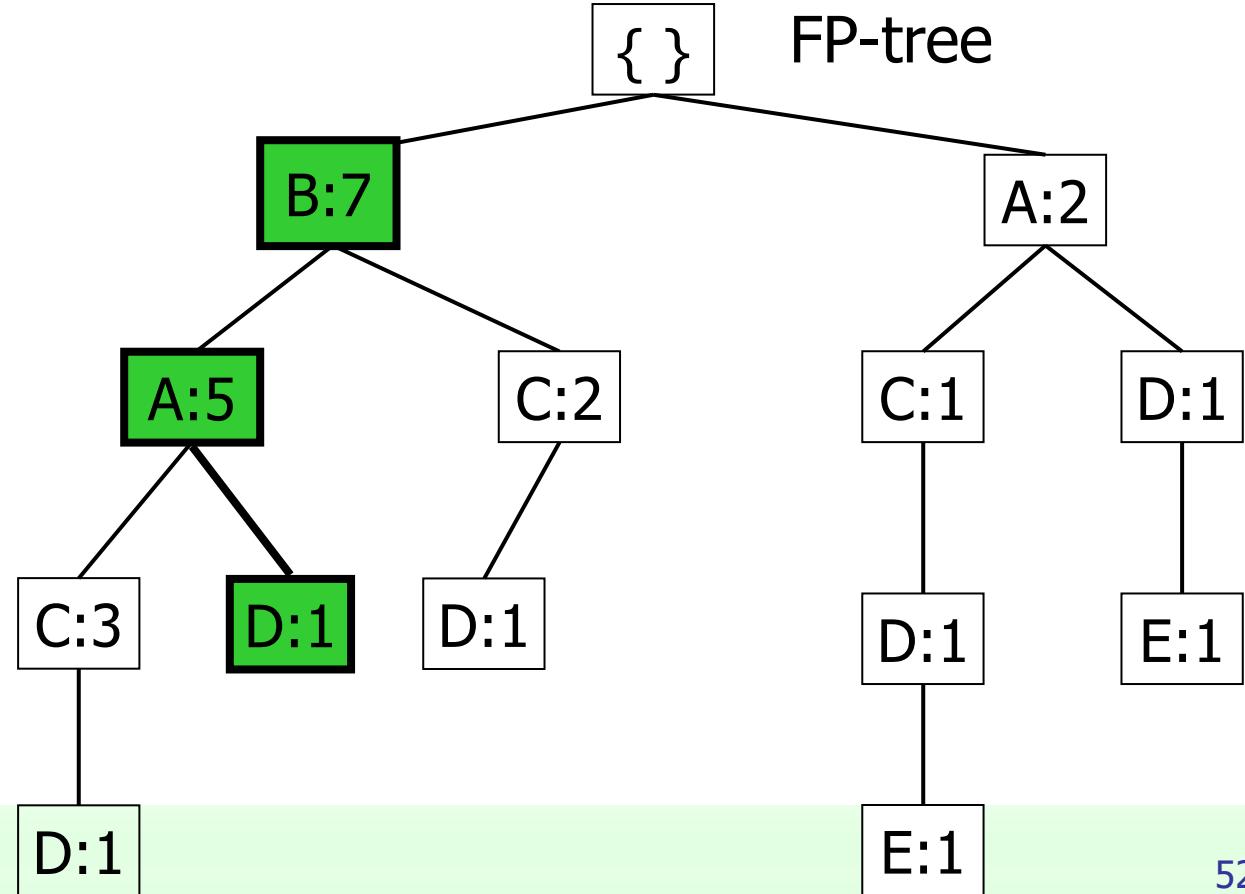


TID	Items
9	{B,A,D}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree



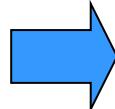


FP-tree construction

Transaction

TID	Items
10	{B,C,E}

Sorted transaction

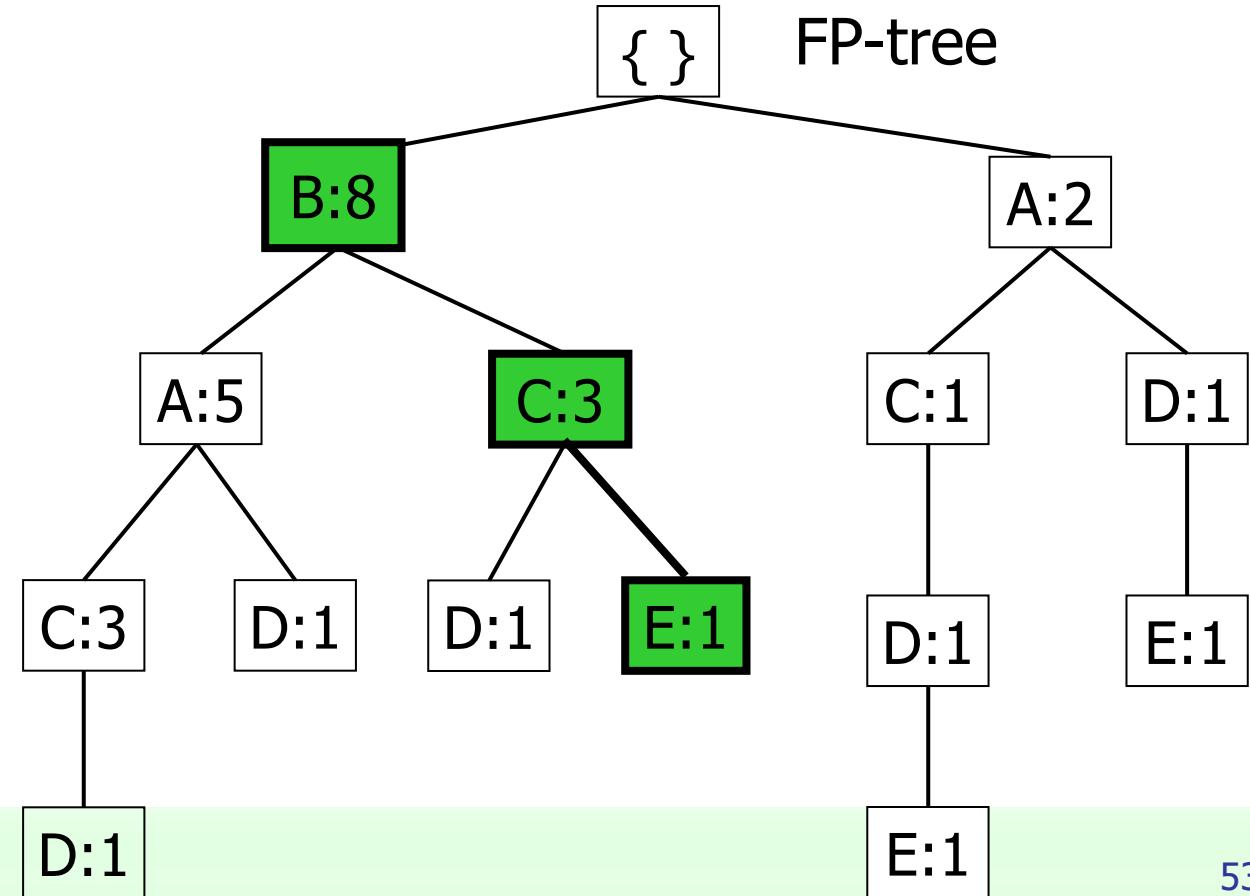


TID	Items
10	{B,C,E}

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

FP-tree

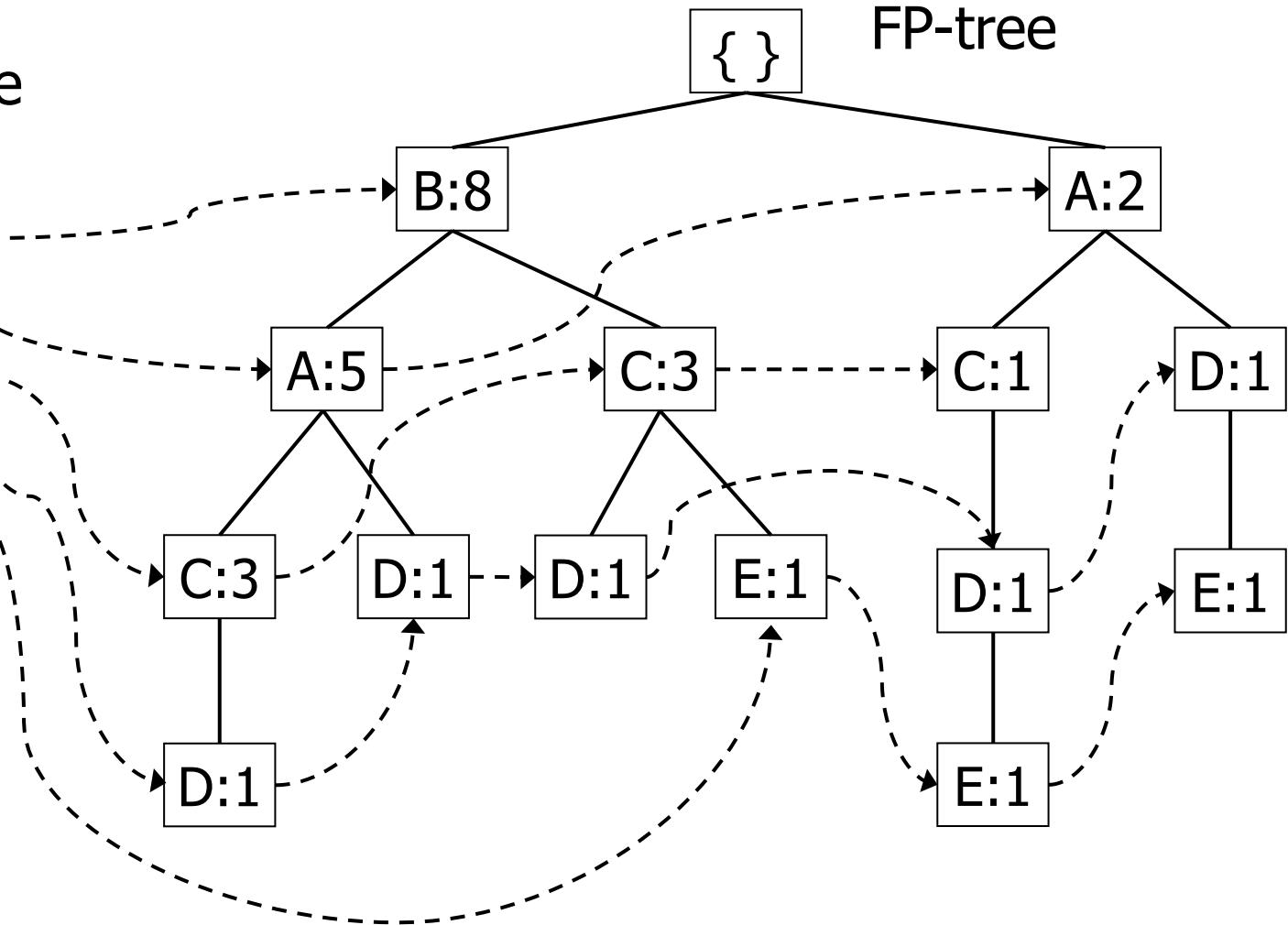




Final FP-tree

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3



Item pointers are used to assist frequent itemset generation



FP-growth Algorithm

- Scan Header Table from lowest support item up
- For each item i in Header Table extract frequent itemsets including item i and items preceding it in Header Table
 - (1) build Conditional Pattern Base for item i (i -CPB)
 - Select prefix-paths of item i from FP-tree
 - (2) recursive invocation of FP-growth on i -CPB

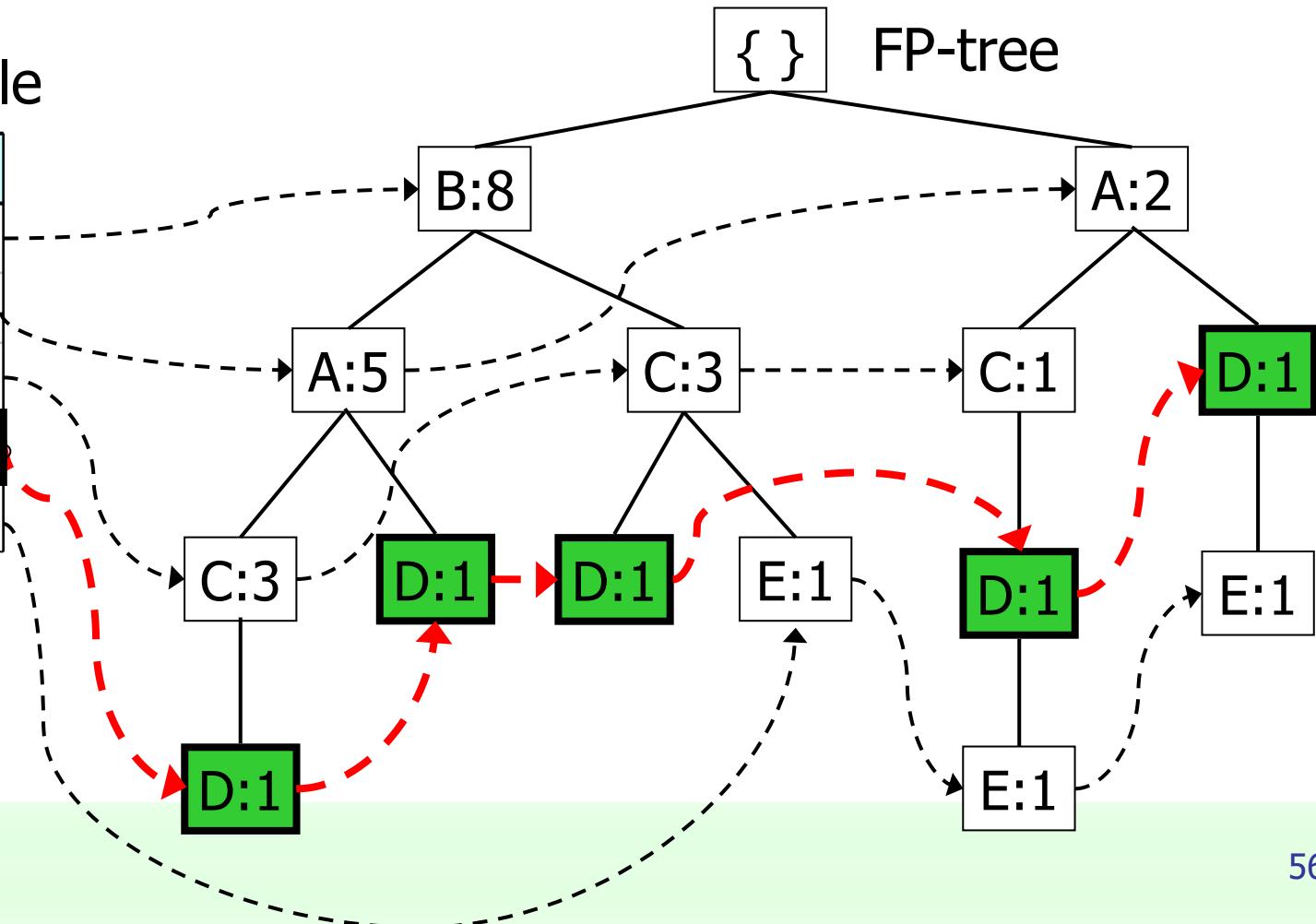


Example

- Consider item D and extract frequent itemsets including
 - D and supported combinations of items A, B, C

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3





Conditional Pattern Base of D

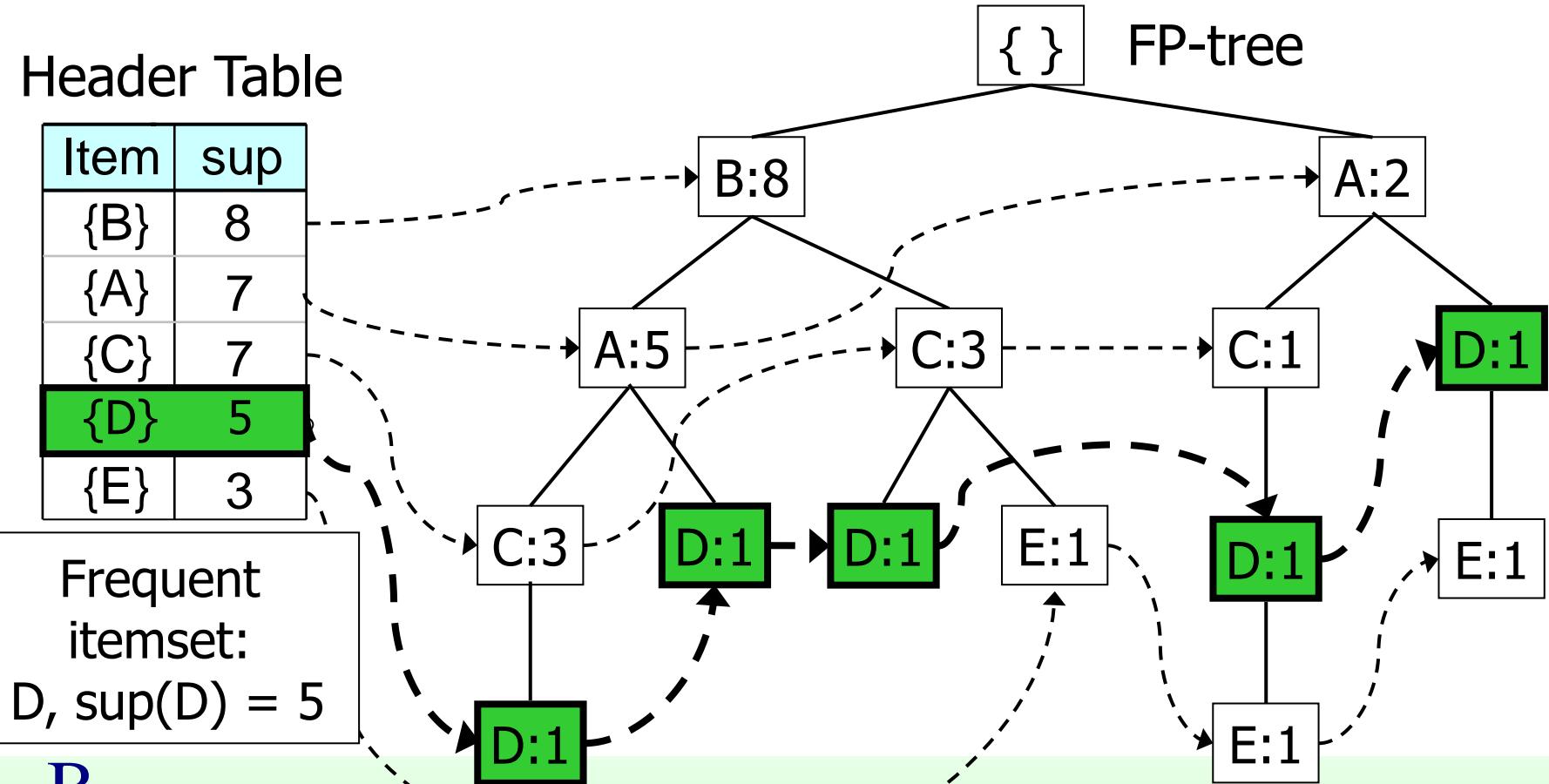
- (1) Build D-CPB
 - Select prefix-paths of item D from FP-tree

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

Frequent itemset:
D, $\text{sup}(D) = 5$

D^B_{MG}





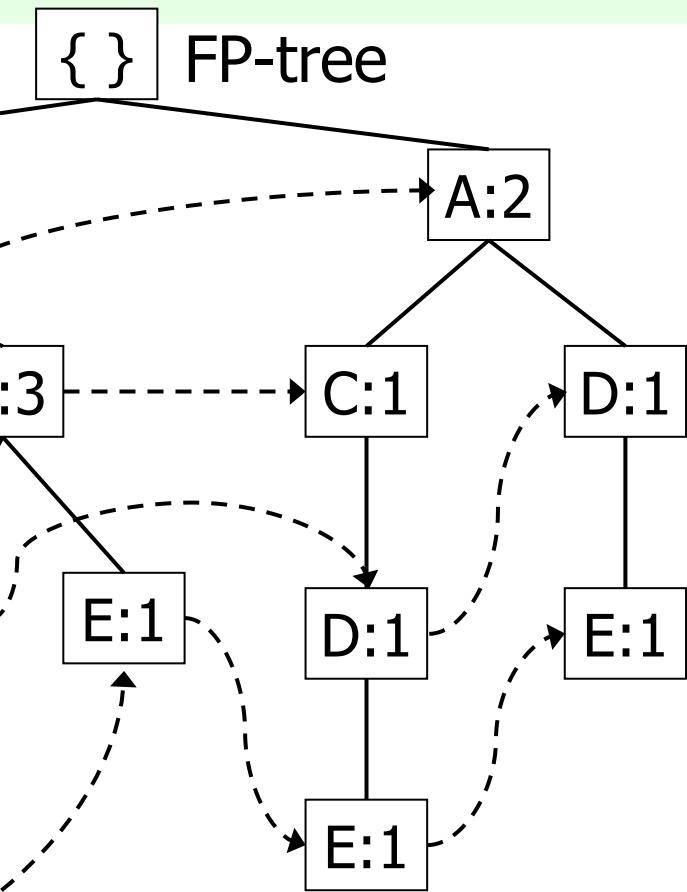
Conditional Pattern Base of D

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

D-CPB

Items	sup
{B,A,C}	1





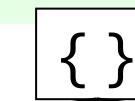
Conditional Pattern Base of D

Header Table

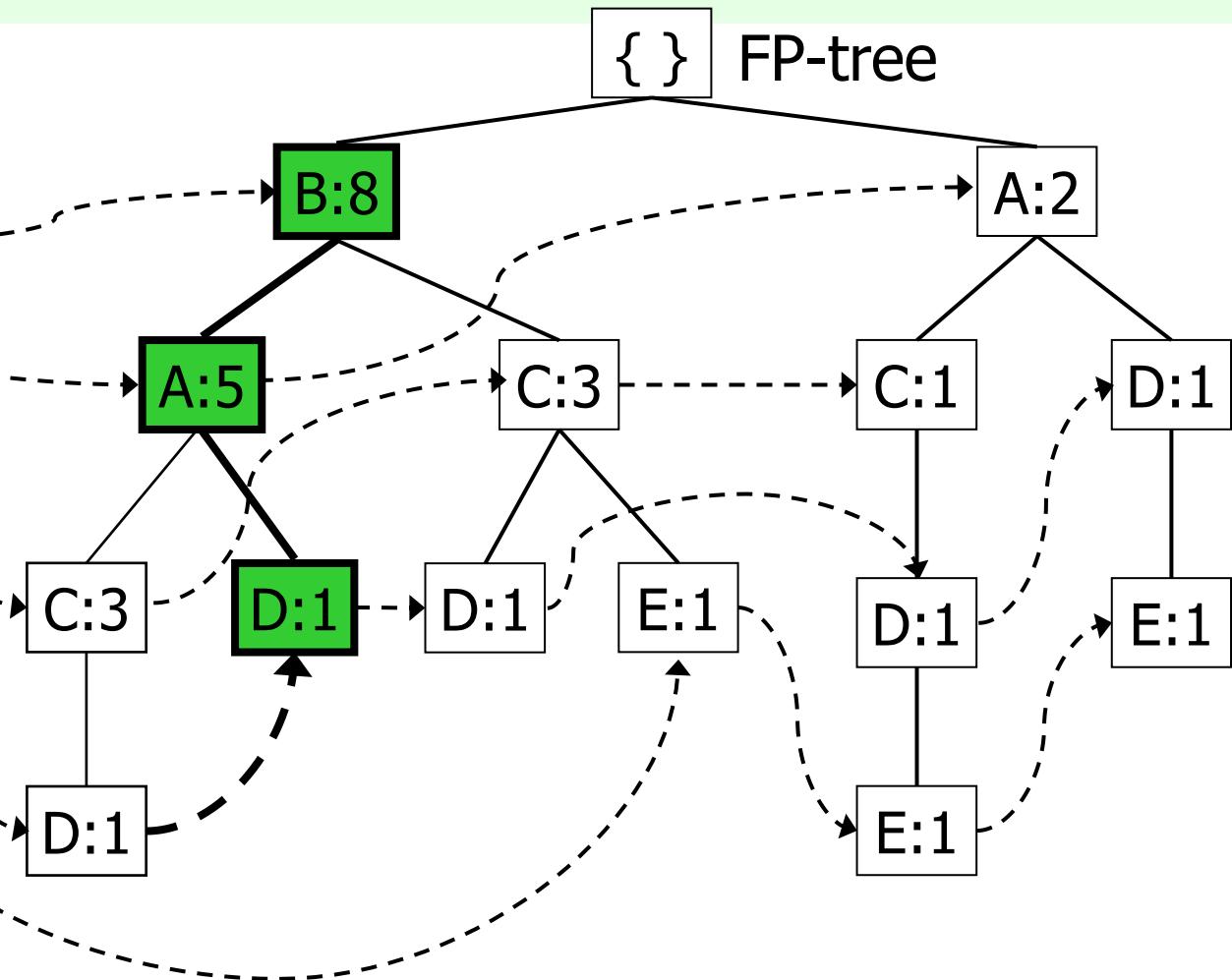
Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

D-CPB

Items	sup
{B,A,C}	1
{B,A}	1



FP-tree





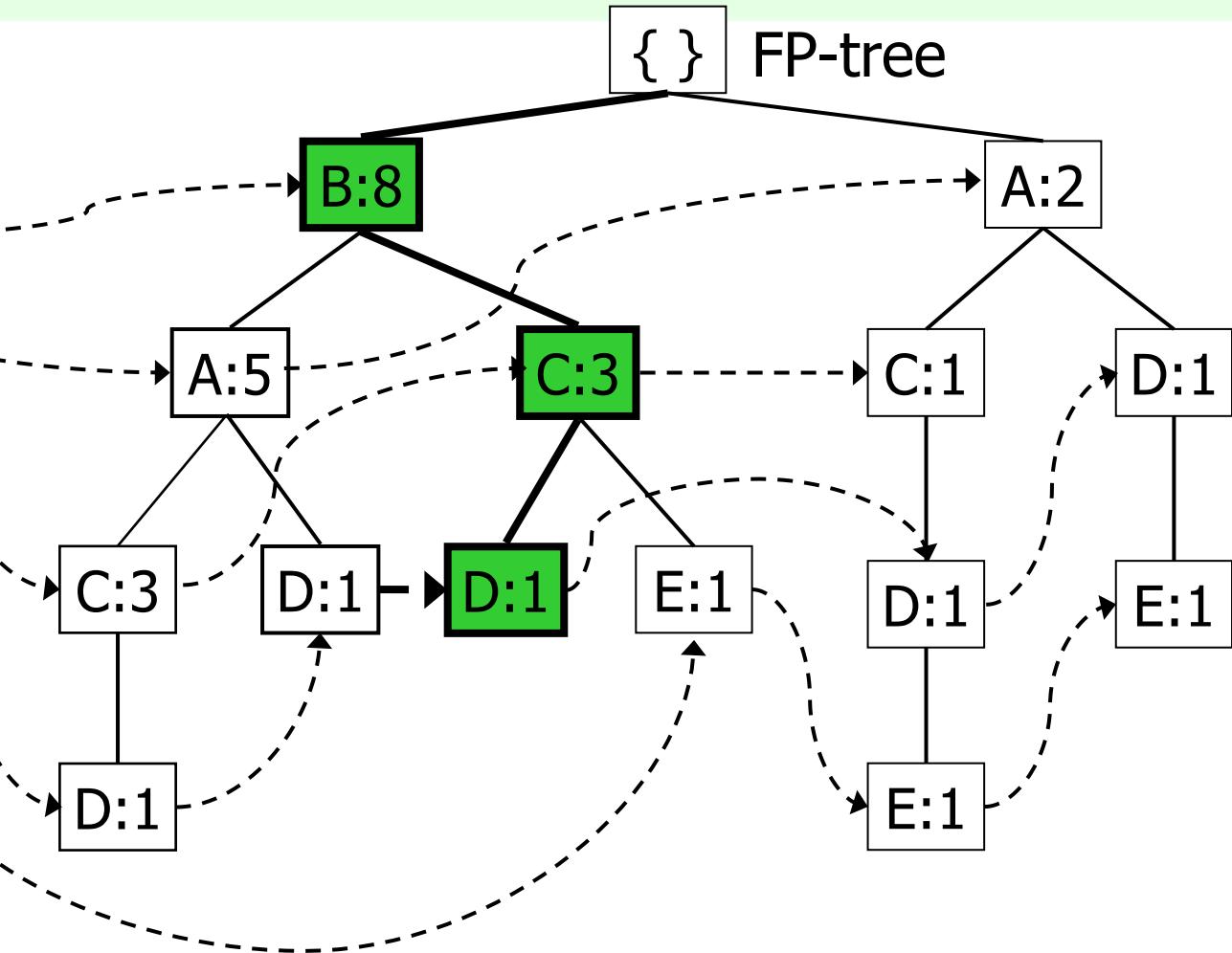
Conditional Pattern Base of D

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

D-CPB

Items	sup
{B,A,C}	1
{B,A}	1
{B,C}	1





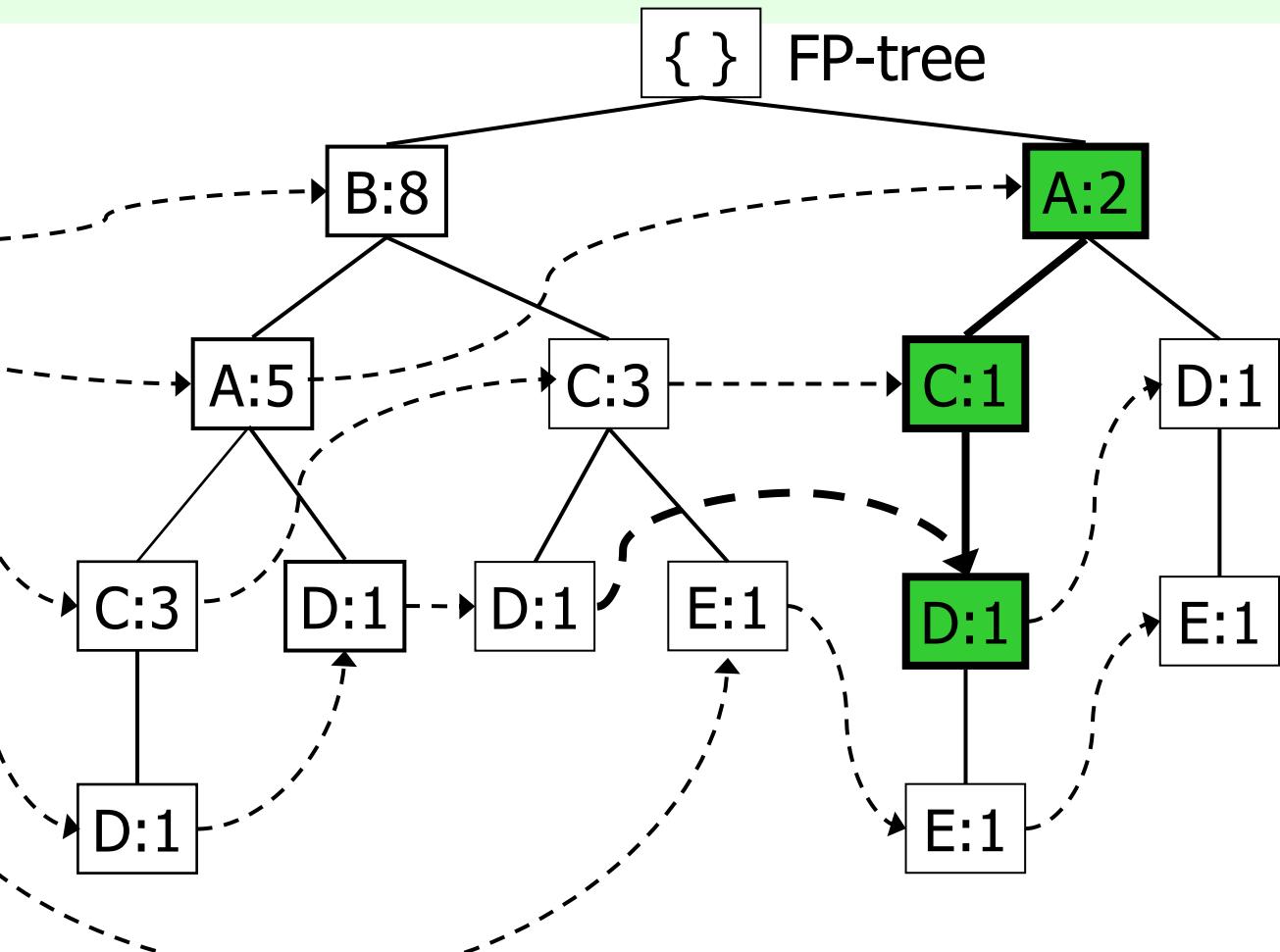
Conditional Pattern Base of D

Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

D-CPB

Items	sup
{B,A,C}	1
{B,A}	1
{B,C}	1
{A,C}	1





Conditional Pattern Base of D

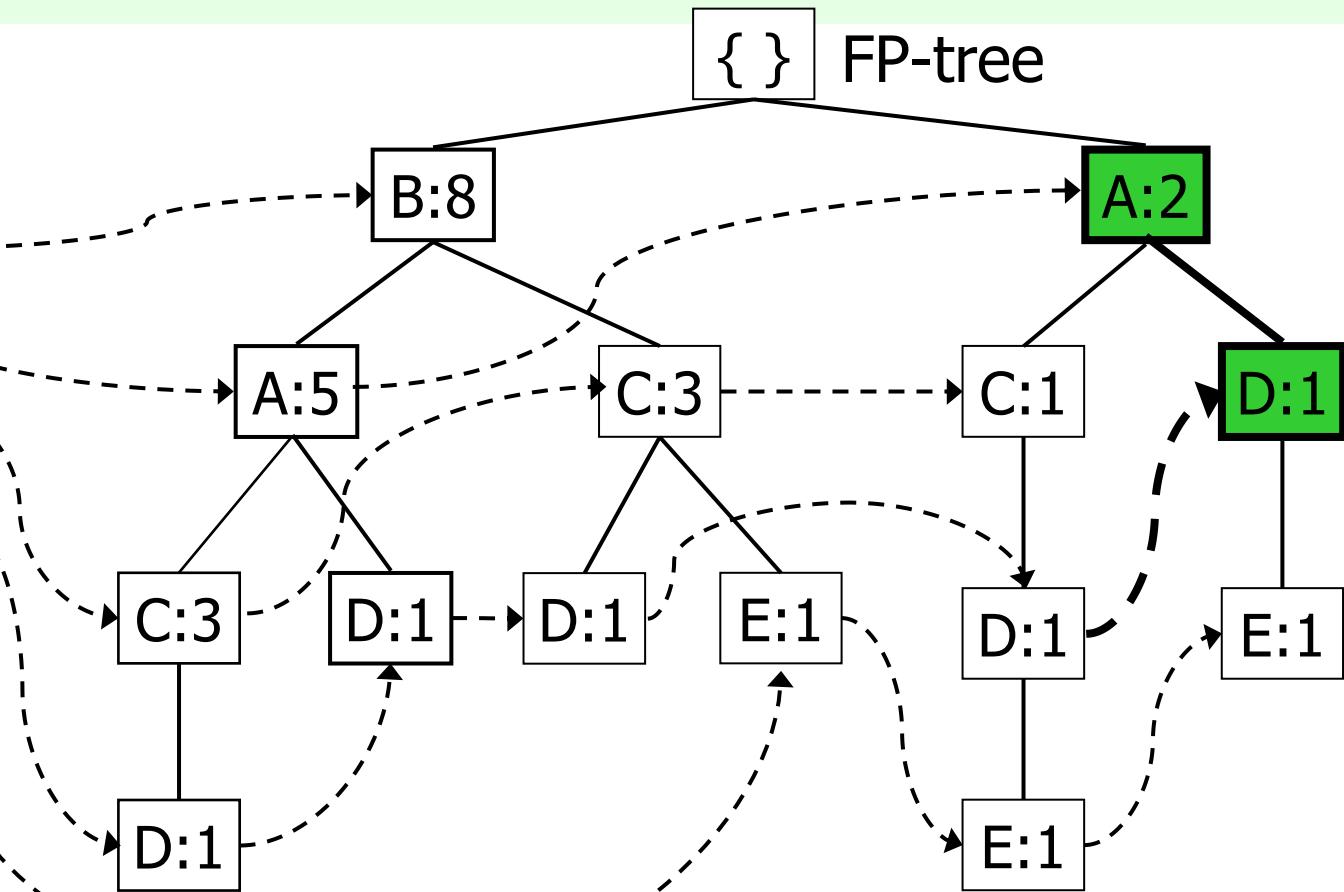
Header Table

Item	sup
{B}	8
{A}	7
{C}	7
{D}	5
{E}	3

D-CPB

Items	sup
{B,A,C}	1
{B,A}	1
{B,C}	1
{A,C}	1
{A}	1

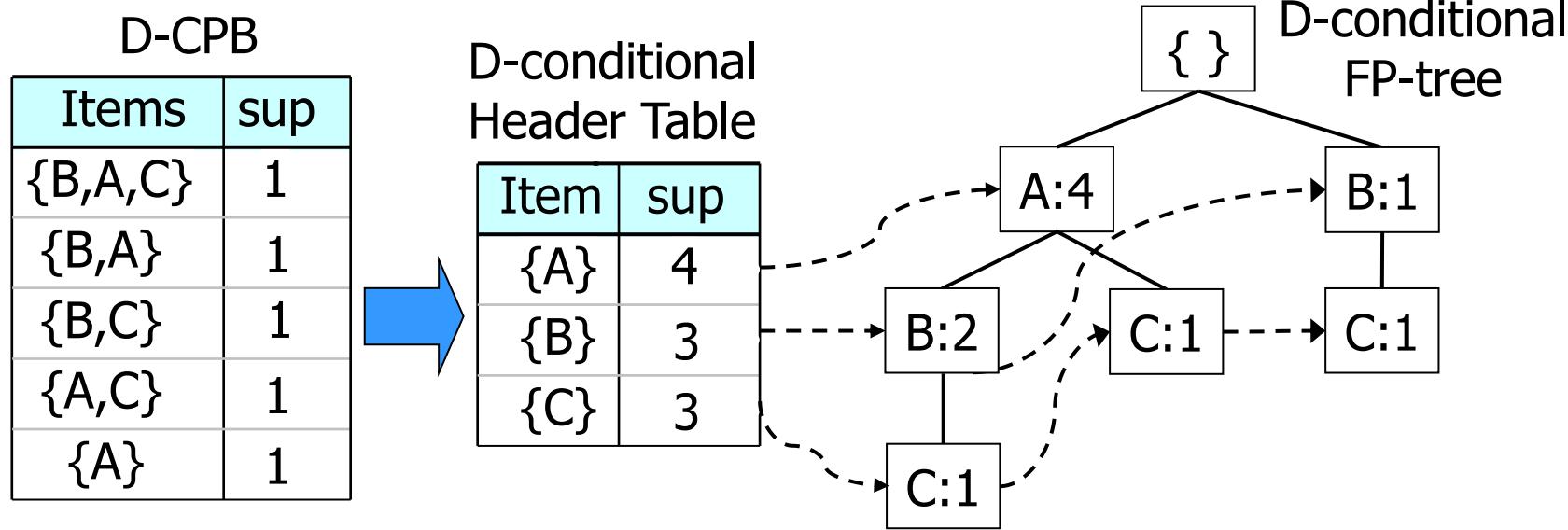
D^B_{MG}





Conditional Pattern Base of D

- (1) Build D-CPB
 - Select prefix-paths of item D from FP-tree



- (2) Recursive invocation of FP-growth on D-CPB



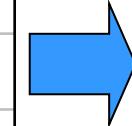
Conditional Pattern Base of DC

- (1) Build DC-CPB
 - Select prefix-paths of item C from D-conditional FP-tree

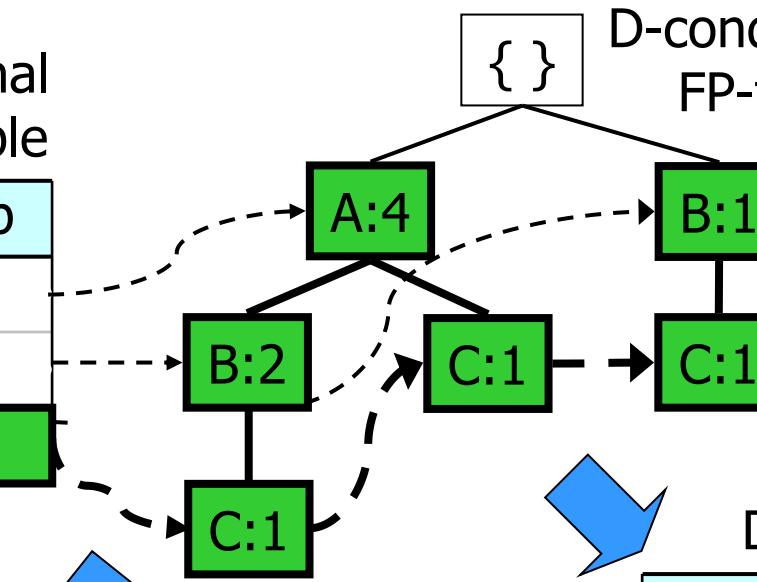
Items	sup
{B,A,C}	1
{B,A}	1
{B,C}	1
{A,C}	1
{A}	1

D-conditional Header Table

Item	sup
{A}	4
{B}	3
{C}	3



D-conditional FP-tree



DC-CPB

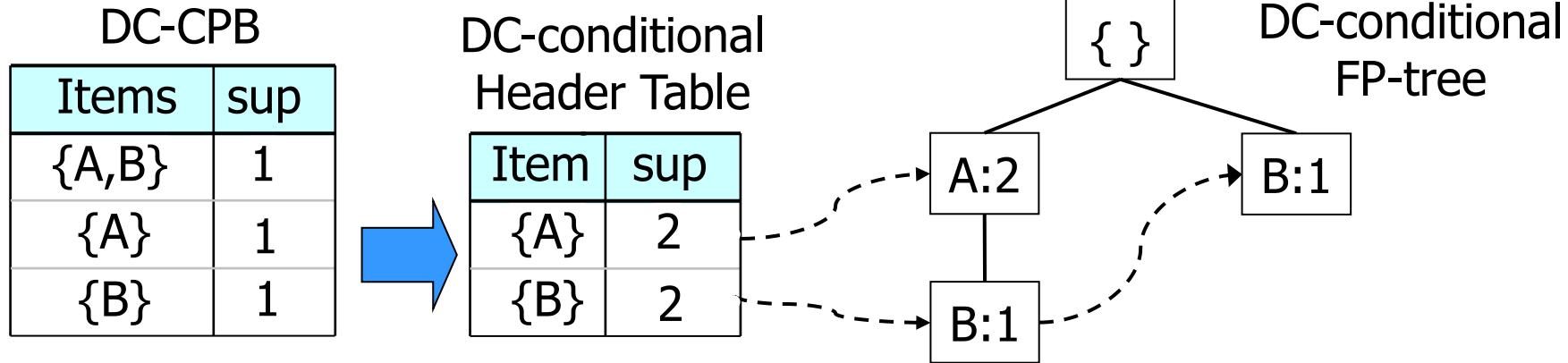
Items	sup
{A,B}	1
{A}	1
{B}	1

Frequent itemset:
DC, sup(DC) = 3



Conditional Pattern Base of DC

- (1) Build DC-CPB
 - Select prefix-paths of item C from D-conditional FP-tree

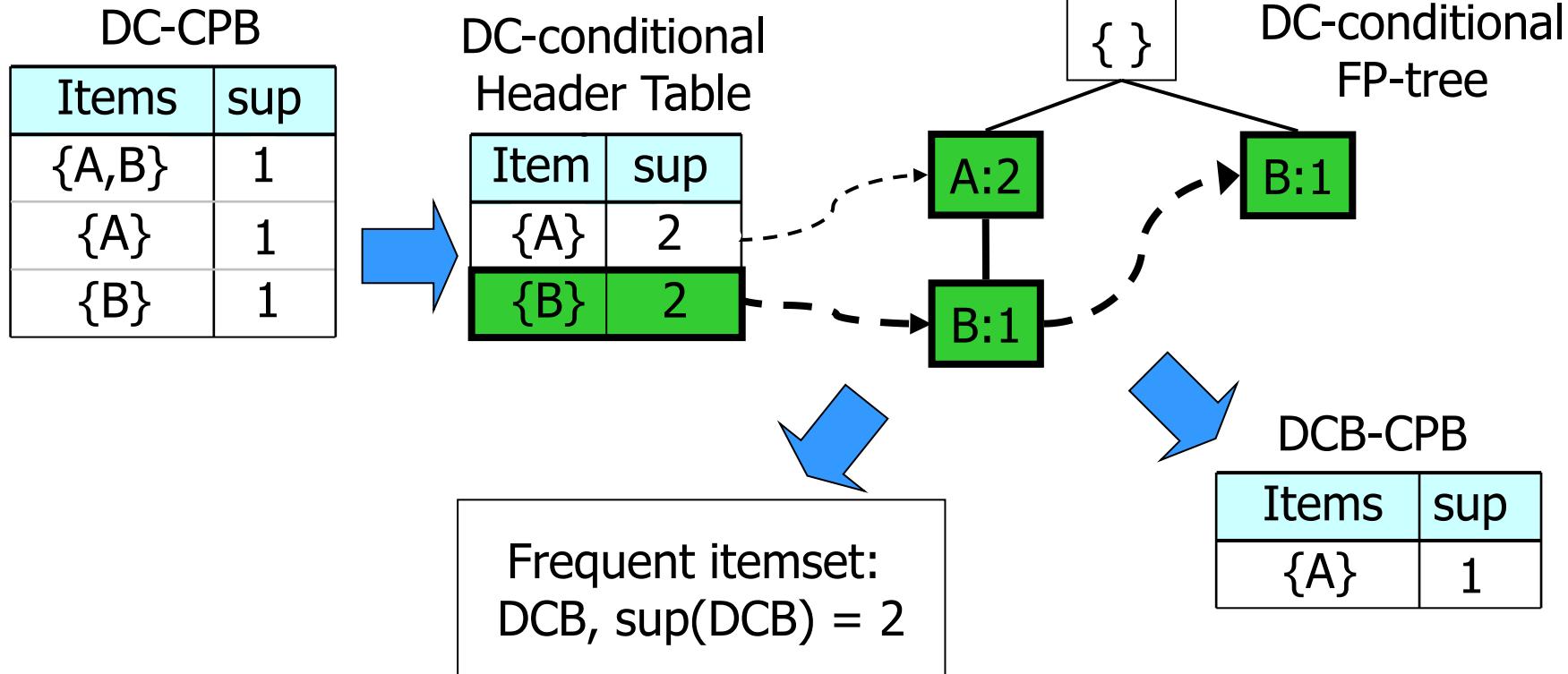


- (2) Recursive invocation of FP-growth on DC-CPB



Conditional Pattern Base of DCB

- (1) Build DCB-CPB
 - Select prefix-paths of item B from DC-conditional FP-tree



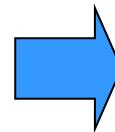


Conditional Pattern Base of DCB

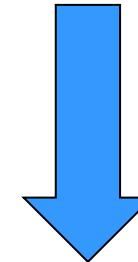
- (1) Build DCB-CPB
 - Select prefix-paths of item B from DC-conditional FP-tree

DCB-CPB

Items	sup
{A}	1



- Item A is infrequent in DCB-CPB
 - A is removed from DCB-CPB
 - DCB-CPB is empty

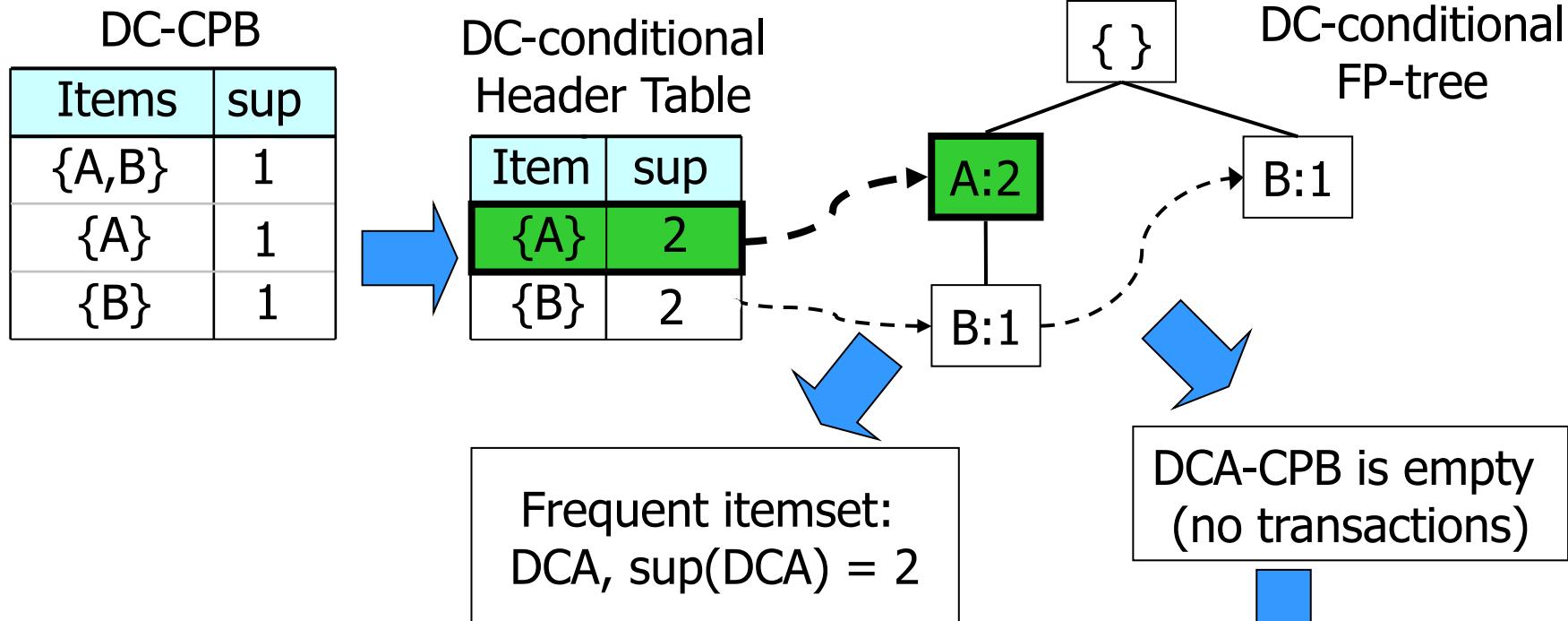


- (2) The search backtracks to DC-CPB



Conditional Pattern Base of DCA

- (1) Build DCA-CPB
 - Select prefix-paths of item A from DC-conditional FP-tree

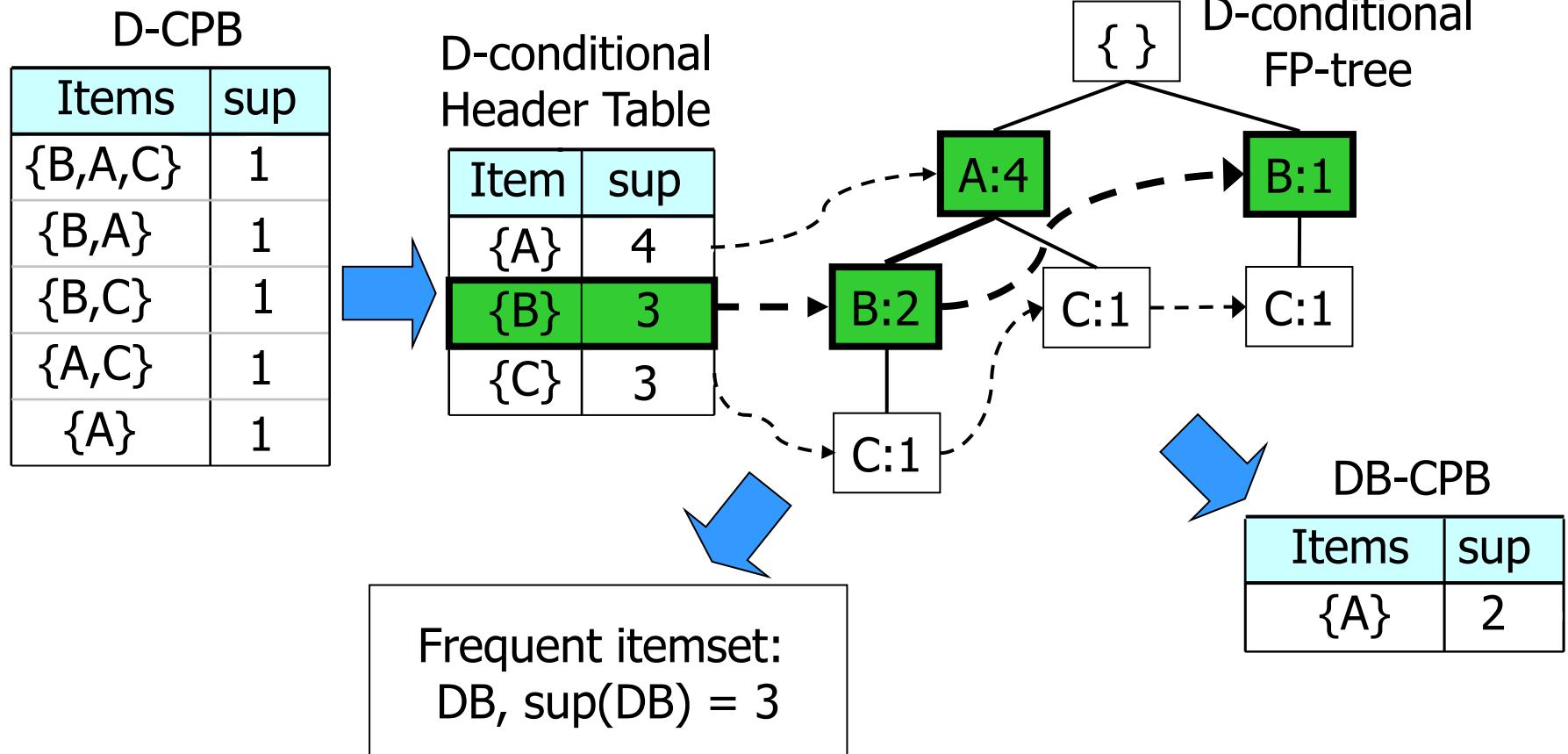


- (2) The search backtracks to D-CBP



Conditional Pattern Base of DB

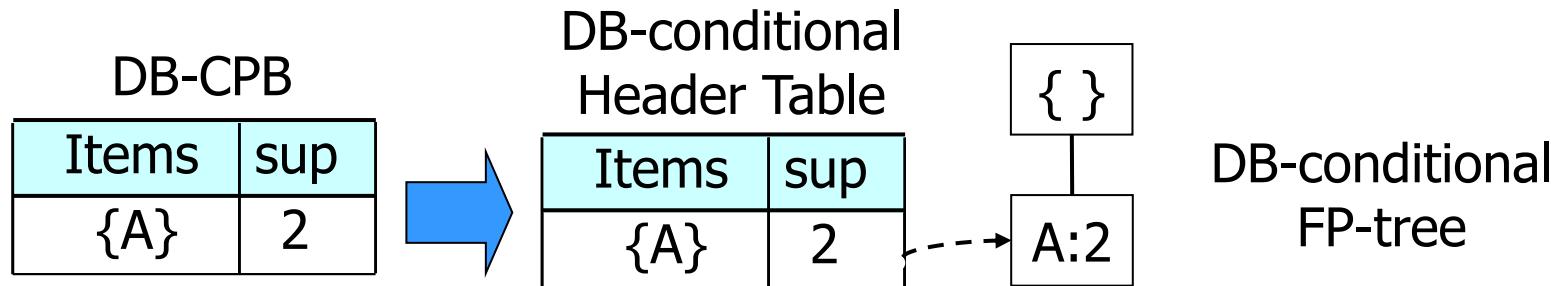
- (1) Build DB-CPB
 - Select prefix-paths of item B from D-conditional FP-tree





Conditional Pattern Base of DB

- (1) Build DB-CPB
 - Select prefix-paths of item B from D-conditional FP-tree

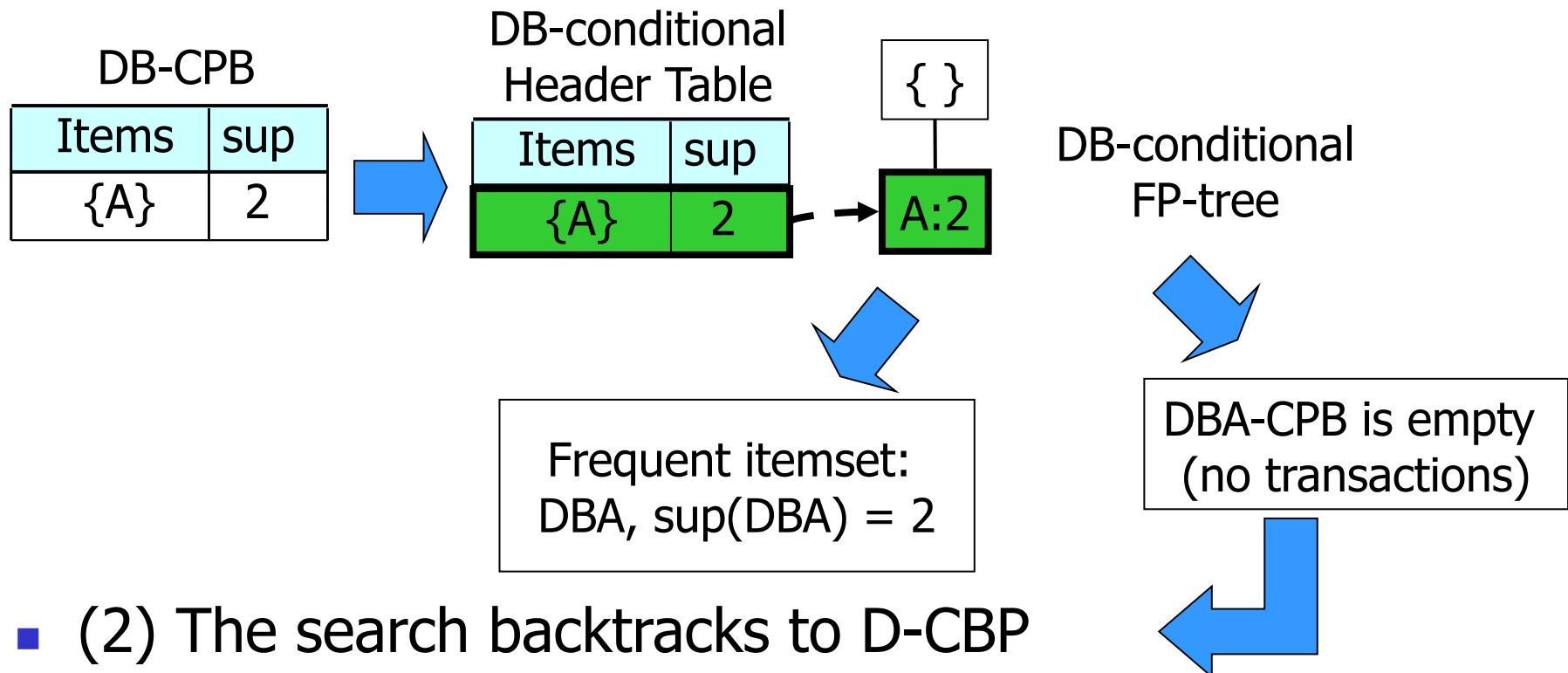


- (2) Recursive invocation of FP-growth on DB-CPB



Conditional Pattern Base of DBA

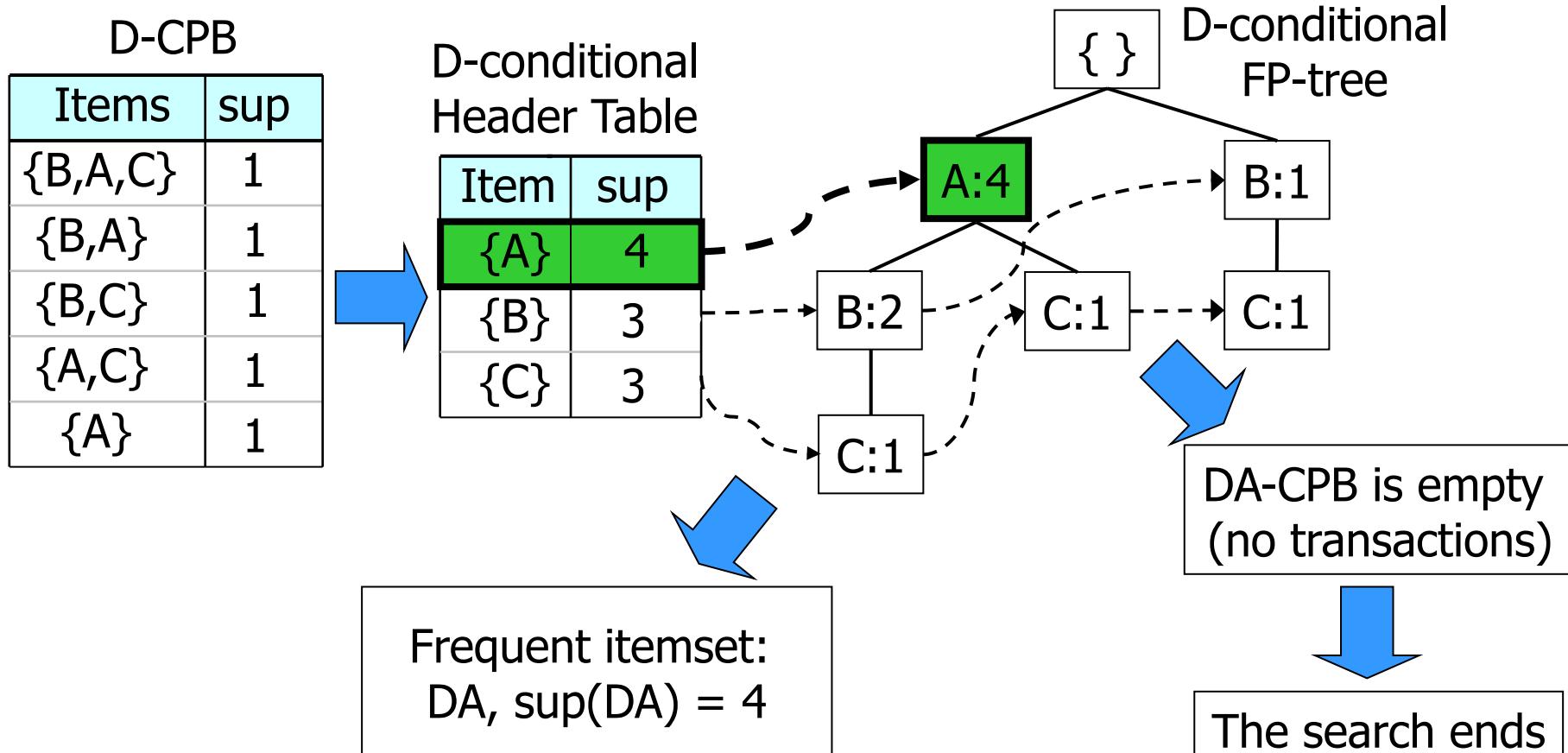
- (1) Build DBA-CPB
 - Select prefix-paths of item A from DB-conditional FP-tree





Conditional Pattern Base of DA

- (1) Build DA-CPB
 - Select prefix-paths of item A from D-conditional FP-tree



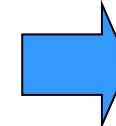


Frequent itemsets with prefix D

- Frequent itemsets including D and supported combinations of items B,A,C

Example DB

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



itemsets	sup
{D}	5
{A,D}	4
{B,D}	3
{C,D}	3
{A,B,D}	2
{A,C,D}	2
{B,C,D}	2

$\text{minsup} > 1$



Other approaches

- Many other approaches to frequent itemset extraction
- May exploit a different database representation
 - represent the tidset of each item [Zak00]

Horizontal
Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				



Compact Representations

- Some itemsets are redundant because they have identical support as their supersets

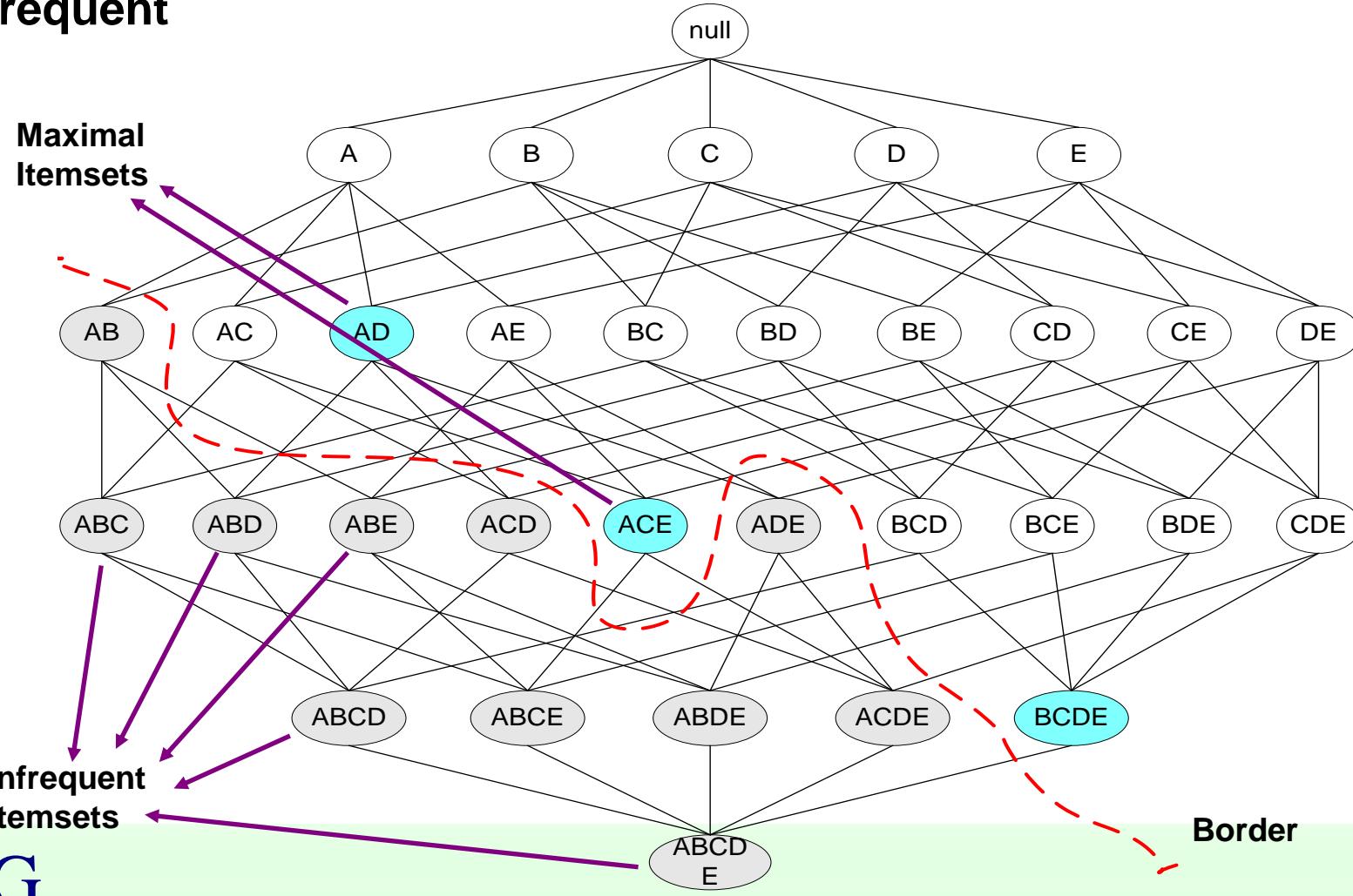
TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	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	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	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
12	0	0	0	0	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
13	0	0	0	0	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
14	0	0	0	0	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
15	0	0	0	0	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

- Number of frequent itemsets = $3 \times \sum_{k=1}^{10} \binom{10}{k}$
- A compact representation is needed



Maximal Frequent Itemset

An itemset is frequent maximal if none of its immediate supersets is frequent





Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

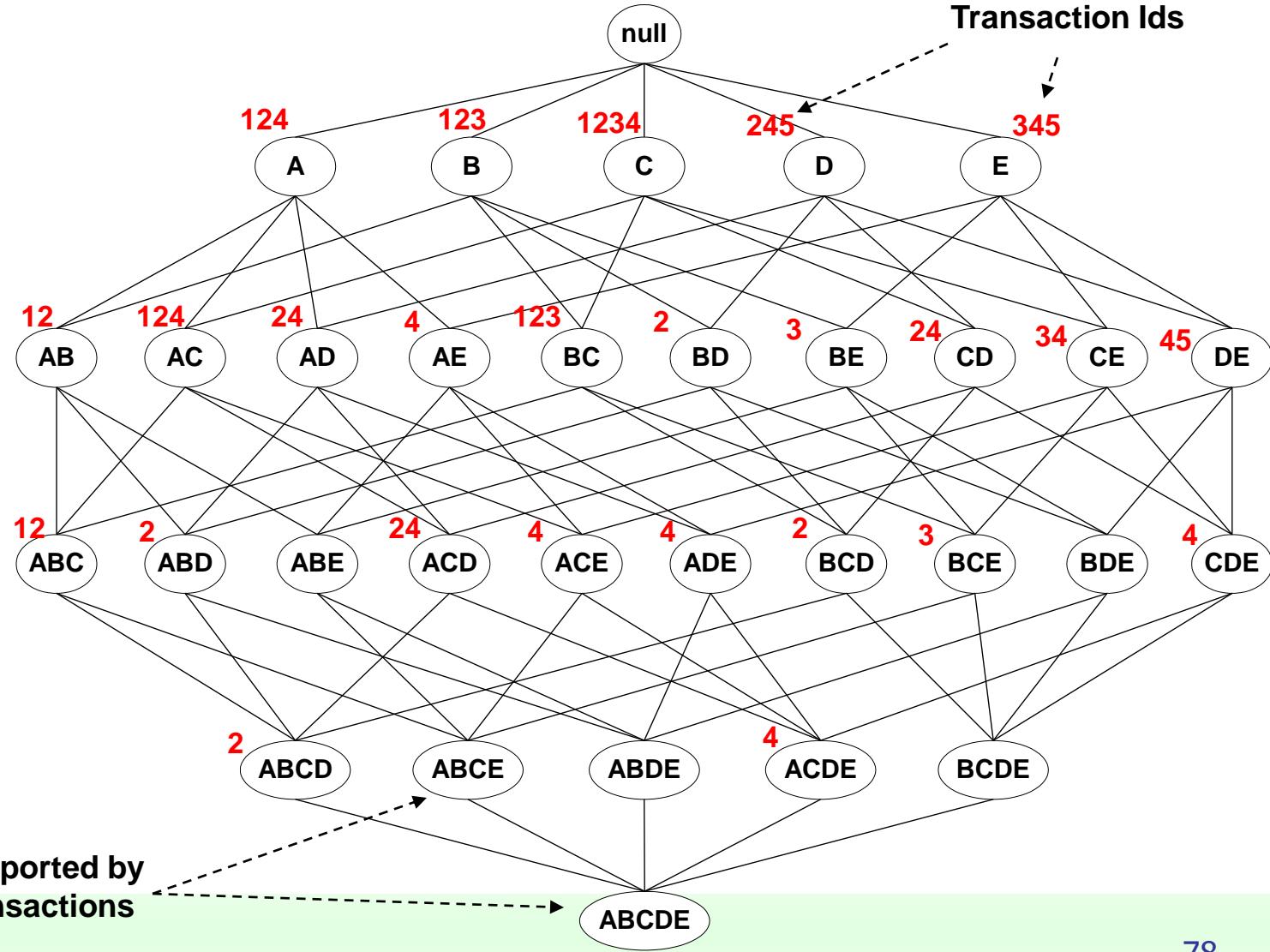
itemset	sup
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

itemset	sup
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2



Maximal vs Closed Itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

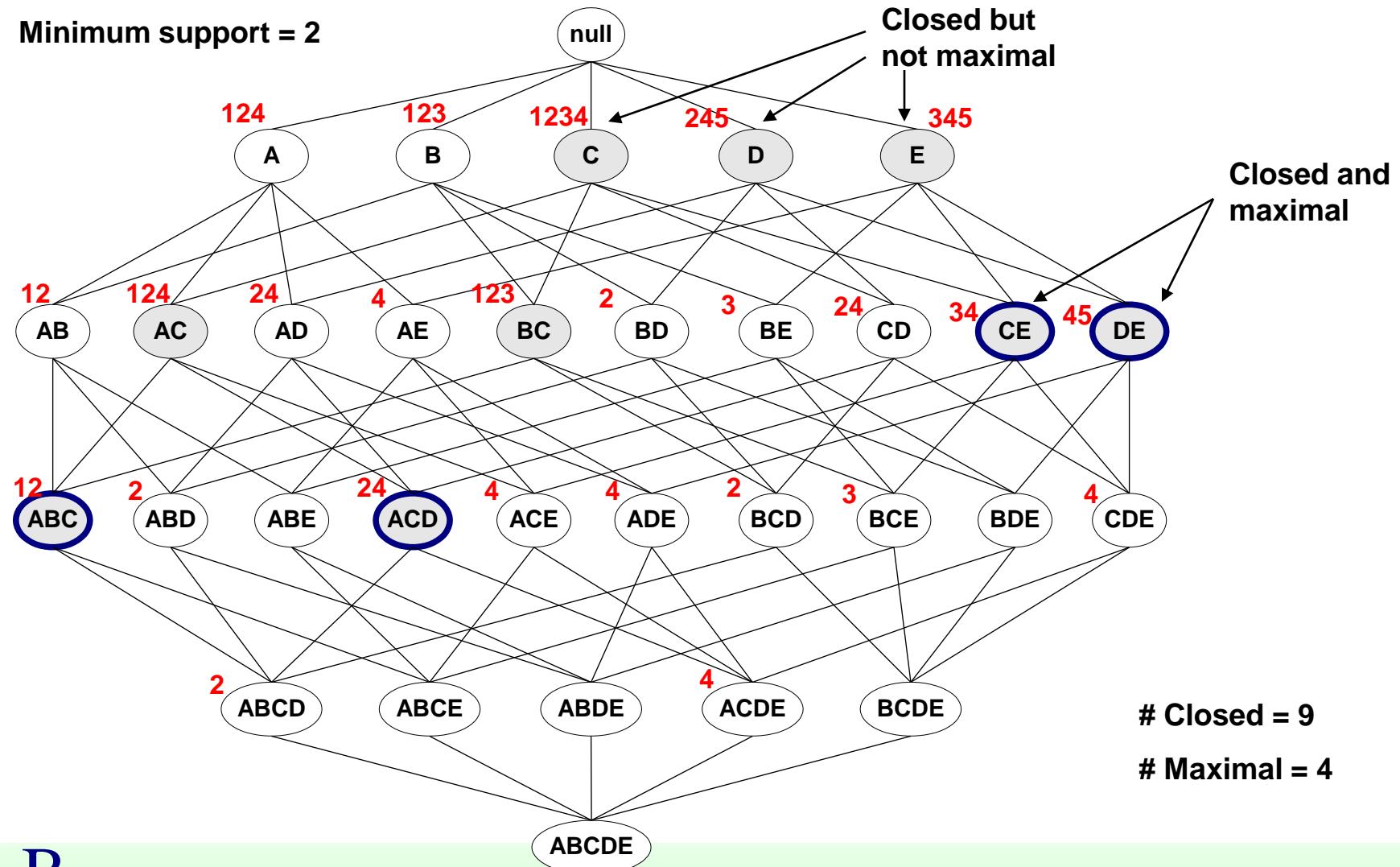


From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



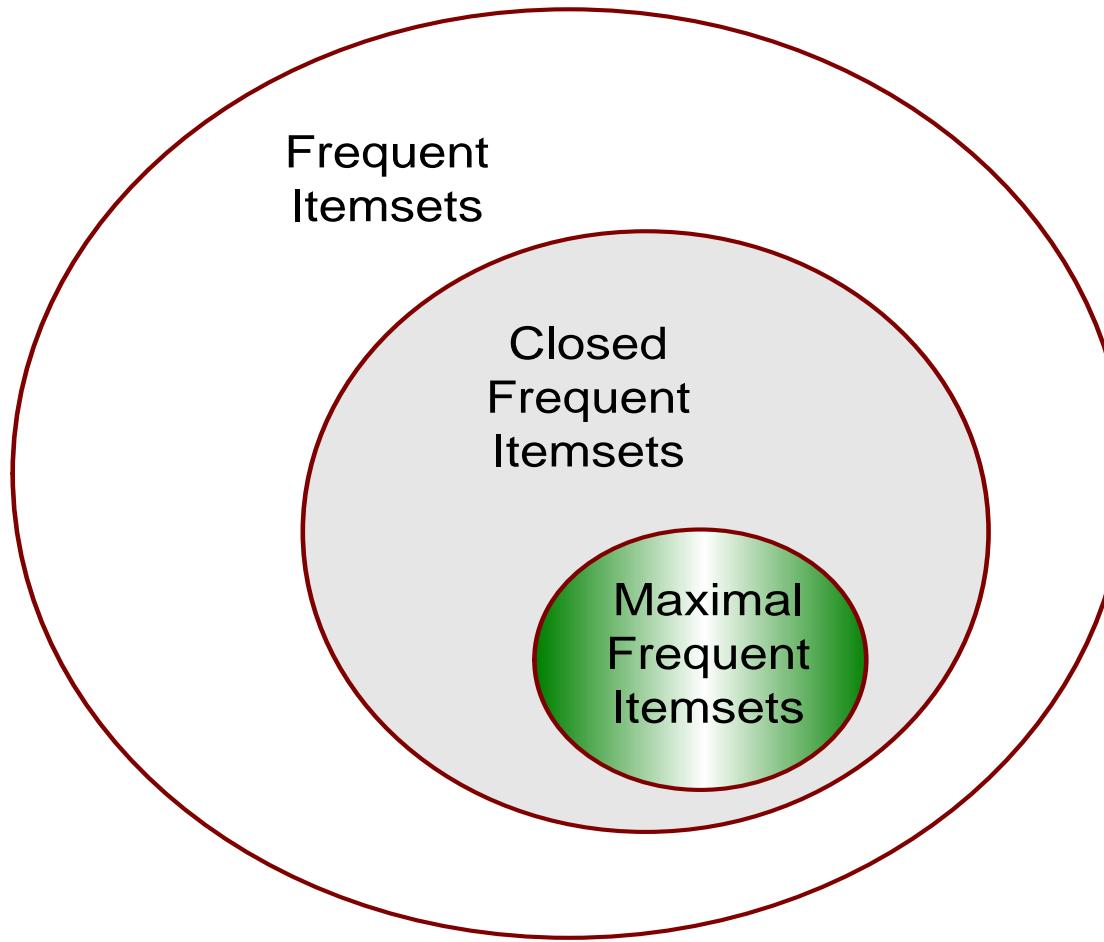
Maximal vs Closed Frequent Itemsets

Minimum support = 2





Maximal vs Closed Itemsets



From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



Effect of Support Threshold

- Selection of the appropriate *minsup* threshold is not obvious
 - If *minsup* is too high
 - itemsets including rare but interesting items may be lost
 - example: pieces of jewellery (or other expensive products)
 - If *minsup* is too low
 - it may become computationally *very expensive*
 - the number of frequent itemsets becomes *very large*



Interestingness Measures

- A large number of patterns may be extracted
 - rank patterns by their interestingness
- Objective measures
 - rank patterns based on statistics computed from data
 - initial framework [Agr94] only considered support and confidence
 - other statistical measures available
- Subjective measures
 - rank patterns according to user interpretation [Silb98]
 - interesting if it contradicts the expectation of a user
 - interesting if it is actionable



Confidence measure: always reliable?

- 5000 high school students are given
 - 3750 eat cereals
 - 3000 play basket
 - 2000 eat cereals and play basket
- Rule

play basket \Rightarrow eat cereals
sup = 40%, conf = 66,7%

is misleading because eat cereals has sup 75% ($>66,7\%$)

- Problem caused by high frequency of rule head
 - negative correlation

	basket	not basket	total
cereals	2000	1750	3750
not cereals	1000	250	1250
total	3000	2000	5000



Correlation or lift

$r: A \Rightarrow B$

$$\text{Correlation} = \frac{P(A, B)}{P(A)P(B)} = \frac{\text{conf}(r)}{\text{sup}(B)}$$

- Statistical independence
 - Correlation = 1
- Positive correlation
 - Correlation > 1
- Negative correlation
 - Correlation < 1



Example

- Association rule

play basket \Rightarrow eat cereals

has corr = 0.89

- negative correlation

- but rule

play basket \Rightarrow not (eat cereals)

has corr = 1,34



#	Measure	Formula
1	ϕ -coefficient	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}{2 - \max_j P(A_j) - \max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(\bar{A},\bar{B})}$
4	Yule's Q	$\frac{P(A,B)P(\bar{A}\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A}\bar{B}) + P(A,\bar{B})P(\bar{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\bar{A}\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A}\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$ $\frac{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i) \log P(A_i), -\sum_j P(B_j) \log P(B_j))}$
7	Mutual Information (M)	
8	J-Measure (J)	$\max \left(P(A,B) \log \left(\frac{P(B A)}{P(B)} \right) + P(\bar{A}\bar{B}) \log \left(\frac{P(\bar{B} \bar{A})}{P(\bar{B})} \right), P(A,B) \log \left(\frac{P(A B)}{P(A)} \right) + P(\bar{A}\bar{B}) \log \left(\frac{P(\bar{A} B)}{P(\bar{A})} \right) \right)$
9	Gini index (G)	$\max \left(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2, P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2 \right)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B A), P(A B))$
12	Laplace (L)	$\max \left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2} \right)$
13	Conviction (V)	$\max \left(\frac{P(A)P(\bar{B})}{P(A\bar{B})}, \frac{P(B)P(\bar{A})}{P(B\bar{A})} \right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B) - P(A)P(B)$
17	Certainty factor (F)	$\max \left(\frac{P(B A)-P(B)}{1-P(B)}, \frac{P(A B)-P(A)}{1-P(A)} \right)$
18	Added Value (AV)	$\max(P(B A) - P(B), P(A B) - P(A))$
19	Collective strength (S)	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$ $\frac{P(A,B)}{P(A) + P(B) - P(A,B)}$
20	Jaccard (ζ)	
21	Klosgen (K)	$\sqrt{P(A,B)} \max(P(B A) - P(B), P(A B) - P(A))$



Considering weight

- Items may be characterized by different importance within a transaction
 - Example: product quantity or price in transactions
- Transactions may be weighted
 - Example: discount on entire market basket
- Weighted dataset
 - Each item is assigned a weight measuring its relevance in the corresponding transaction



Weighted association rules

- Consider item/transaction weights during association rule extraction
- Extend rule quality measures
 - E.g., weighted support, weighted confidence
- Apply ad-hoc weight aggregation functions
 - E.g., min, max, avg



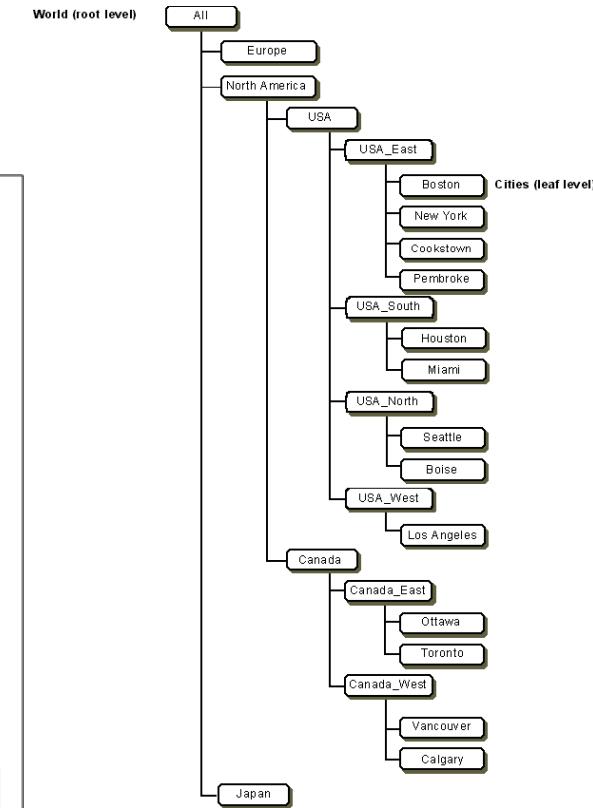
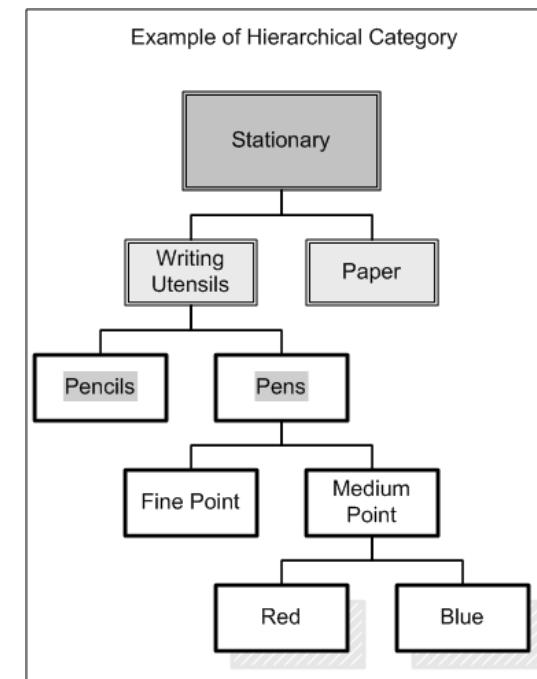
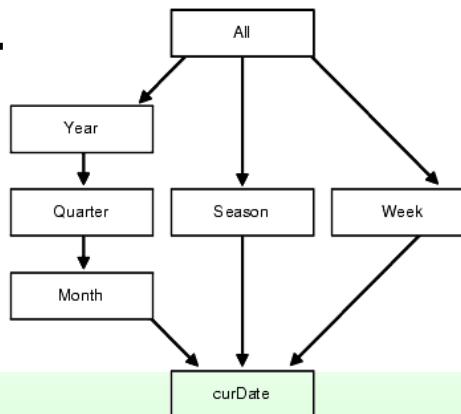
Considering hierarchies

■ Generalization hierarchies

- Aggregation over attributes in a dataset
- Typically user provided

■ Examples

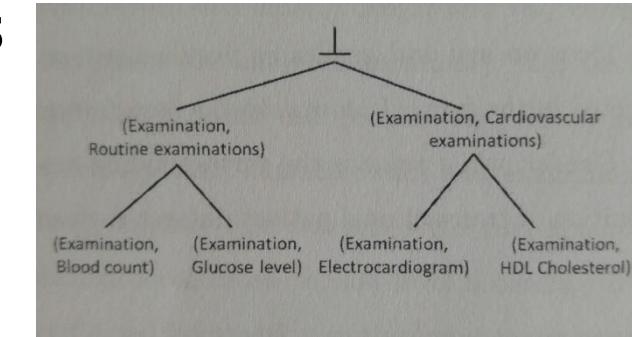
- Time hierarchy
- Product category
- Location hierarchy





Taxonomy

- A taxonomy is a set of is-a hierarchies that aggregate data items into higher-level concepts
- Data item
 - Instance in the (transactional) dataset
 - Represents detailed concepts
- Generalized item
 - Aggregation in higher-level concepts
 - Represents abstractions on instances





Generalized itemsets

- Sets of items at different generalization levels
 - May contain data items together with generalized items defined in the taxonomy
 - Summarize knowledge represented by a set of lower-level descendants
 - Both frequent and infrequent
- A generalized itemset covers a transaction when all
 - its generalized items are ancestors of items included in the transaction
 - its data items are included in the transaction
- Generalized itemset support
 - ratio between number of covered transactions and dataset cardinality



Context-aware data analysis

- Context data provided by different, possibly heterogeneous, sources
 - Mobile devices provide information on
 - the user context (e.g., GPS coordinates)
 - the supplied services
 - temporal information
 - service description
 - duration
 - Additional information available
 - demographics of the user requesting the service



Generalized itemset example

user: John, time: 6.05 p.m., service: Weather
(s = 0.005%)

- A very low support
 - The itemset may be discarded
 - By generalizing
 - the time attribute on a time period
 - the user on a user category
- user: employee, time: 6 p.m. to 7 p.m., service: Weather***
(s = 0.2%)
- May discover interesting properties generalizing *infrequent* items



Generalized association rules

- Extension of “classical” association rules

$$X \rightarrow Y$$

- X and Y are either generalized or not generalized itemsets
 - Extract associations among data items at multiple abstraction levels
 - Support, confidence and lift are defined accordingly



Patient data analysis

- Analysis of multiple level correlations on patient treatment historical data
 - Dataset collected by an Italian Local Health Center
 - Diabetes complications at various severity levels
 - 95K records, 3.5K patients
 - Features
 - Prescribed examinations (26 examinations, 7 categories)
 - Prescribed drugs (200 drugs, 14 categories)
 - Census patient data (gender, age discretized in age groups)
- Sparse dataset
 - Difficult setting of support threshold
 - Low: generates too many rules
 - High: interesting information at lower levels of abstraction may remain hidden



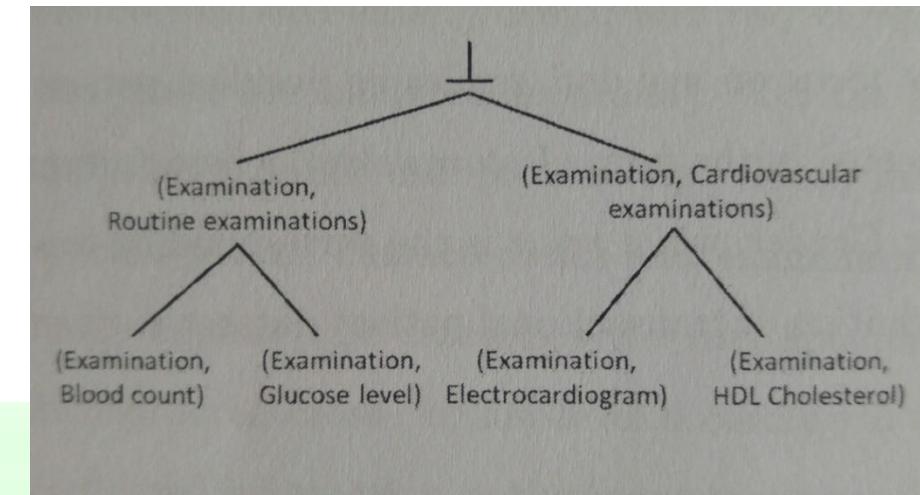
High level rules

- Only generalized itemsets
 - Represent general knowledge
 - May be too high level to perform targeted analyses

- Example

(Examination, Liver) -> (Examination, Kidney)

- Frequently prescribed together
- May be used for examination scheduling





Cross level rules

- Different abstraction levels (generalized items and data items)
 - Combine detailed and general information
- Example
 - (Examination, Liver) -> (Examination, Uric acid)
 - Insight into specific kidney examinations correlated with liver examinations
 - Confidence: 74.8%



Low-level rules

- Only not generalized itemsets (only data items)
 - Very detailed knowledge
 - Covered by high and cross-level rules
 - Large rule set
 - Challenging exploration task
 - Drill down exploration based on formerly extracted high and cross-level rules

Clustering fundamentals

DB
M
G



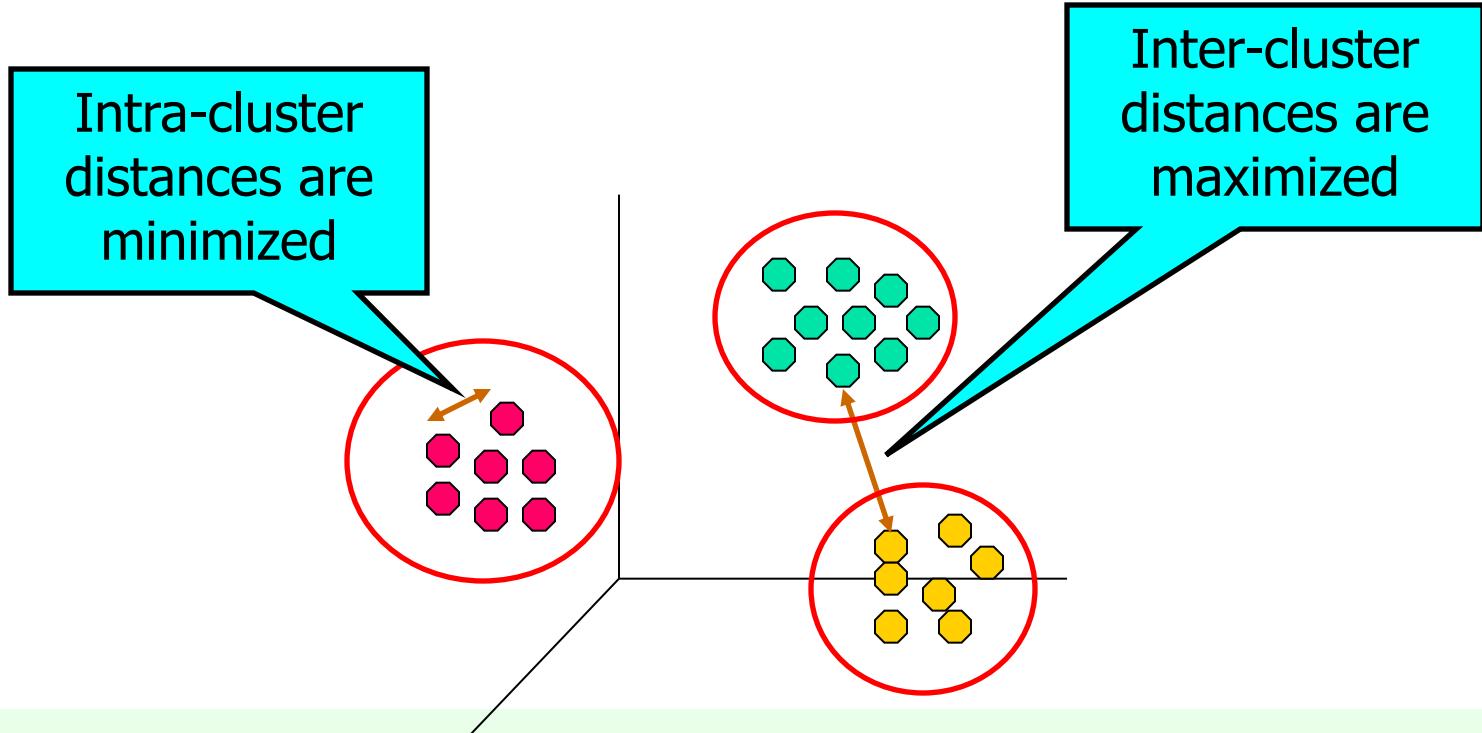
Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Tania Cerquitelli
Politecnico di Torino



What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups





Applications of Cluster Analysis

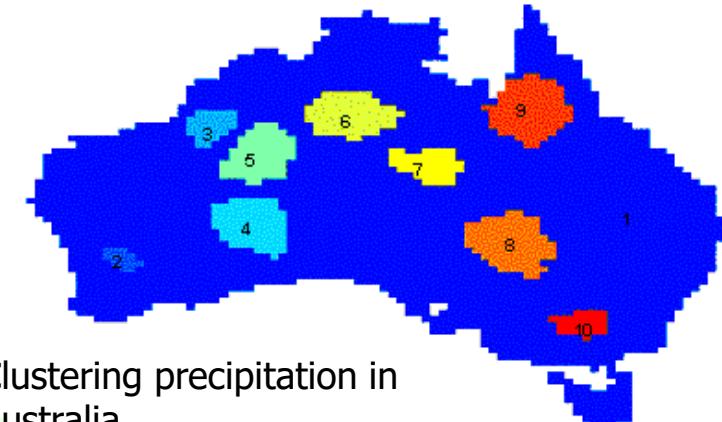
■ Understanding

- Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations

	<i>Discovered Clusters</i>	<i>Industry Group</i>
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP

■ Summarization

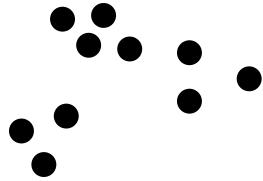
- Reduce the size of large data sets



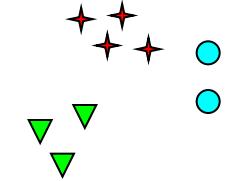
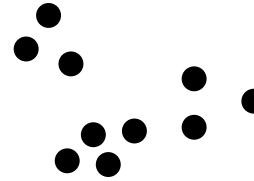
Clustering precipitation in Australia



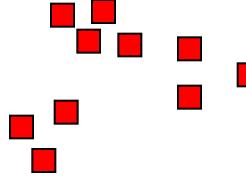
Notion of a Cluster can be Ambiguous



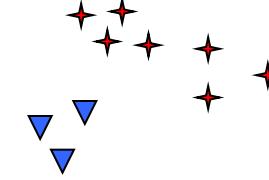
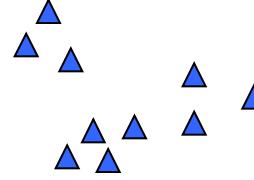
How many clusters?



Six Clusters



Two Clusters



Four Clusters

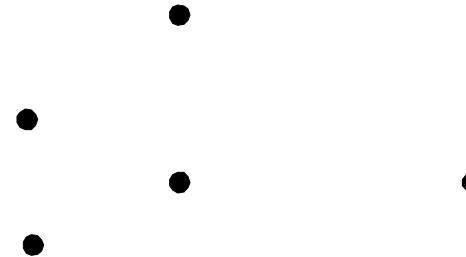
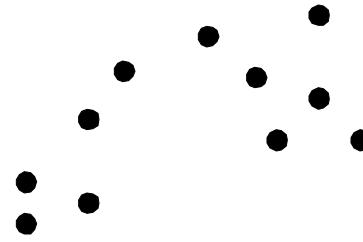


Types of Clusterings

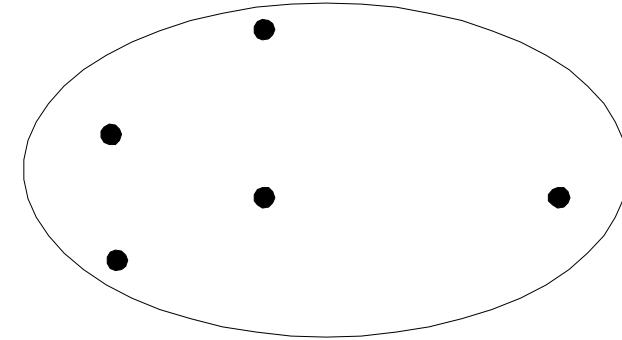
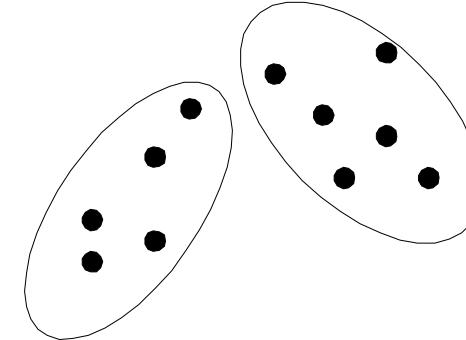
- A **clustering** is a set of clusters
- Important distinction between **hierarchical** and **partitional** sets of clusters
- **Partitional Clustering**
 - Divides data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical clustering**
 - A set of nested clusters organized as a hierarchical tree



Partitional Clustering



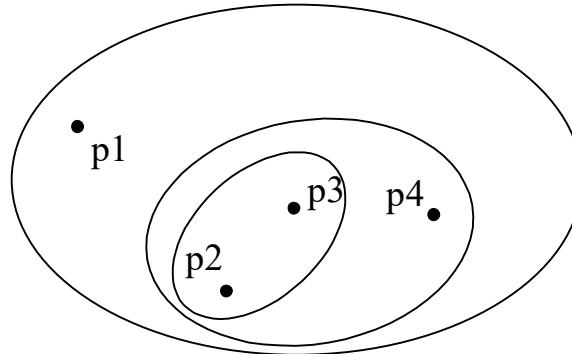
Original Points



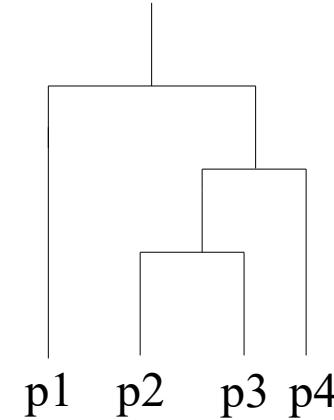
A Partitional Clustering



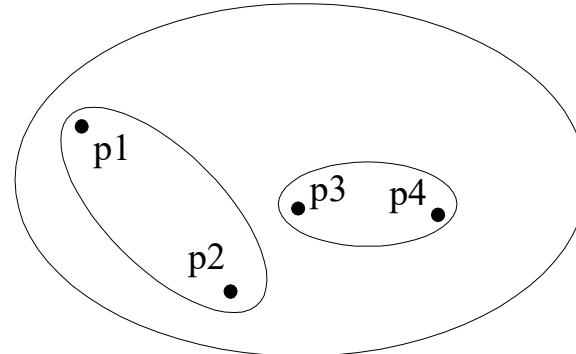
Hierarchical Clustering



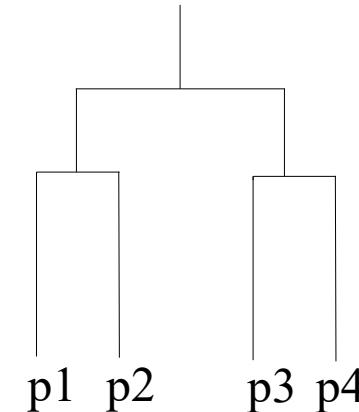
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram



Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
 - In non-exclusive clustering, points may belong to multiple clusters.
- Fuzzy versus non-fuzzy
 - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
 - Weights must sum to 1
 - Probabilistic clustering has similar characteristics
- Partial versus complete
 - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
 - Clusters of widely different sizes, shapes, and densities



Types of Clusters

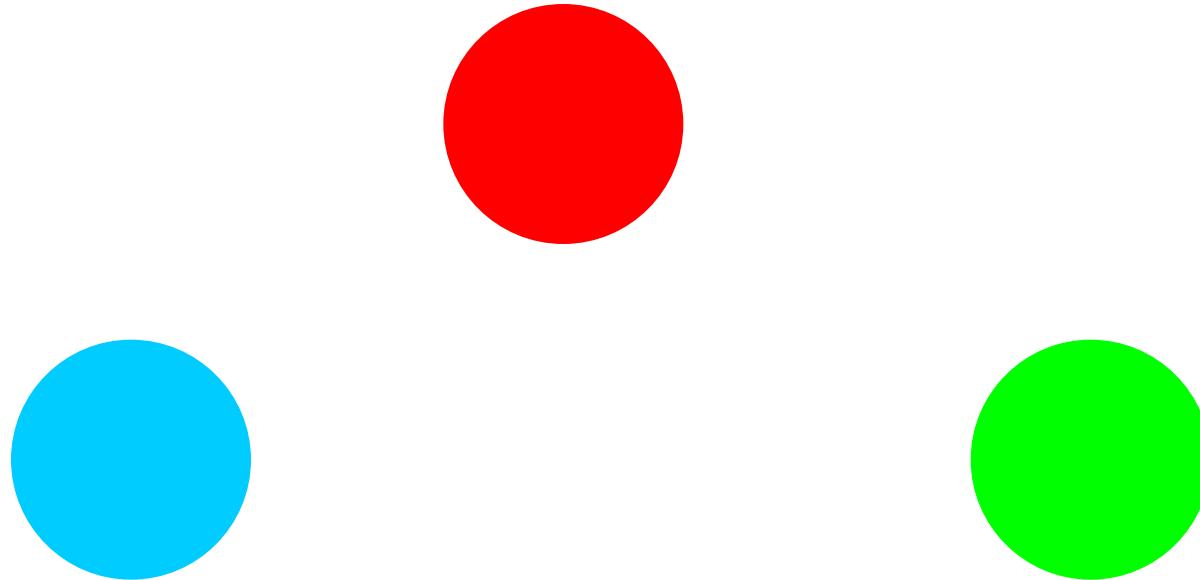
- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function



Types of Clusters: Well Separated

■ Well-Separated Clusters:

- A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters



Types of Clusters: Center-Based

■ Center-based

- A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster
- The center of a cluster is often a **centroid**, the average of all the points in the cluster, or a **medoid**, the most “representative” point of a cluster



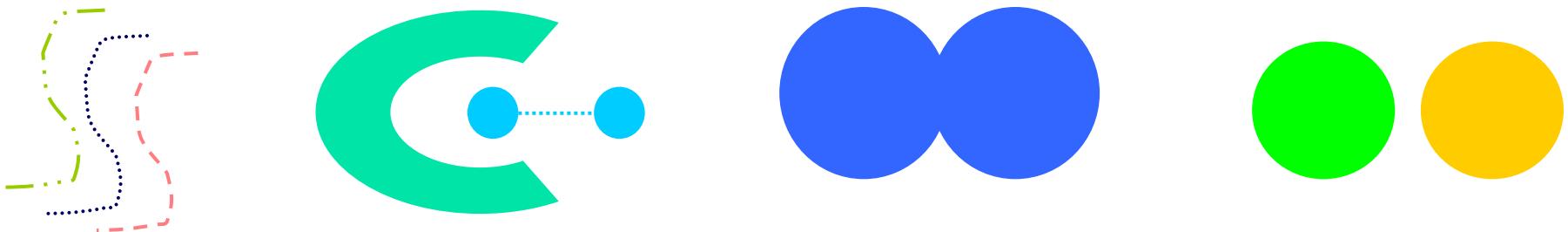
4 center-based clusters



Types of Clusters: Contiguity-Based

■ Contiguous Cluster (Nearest neighbor or Transitive)

- A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



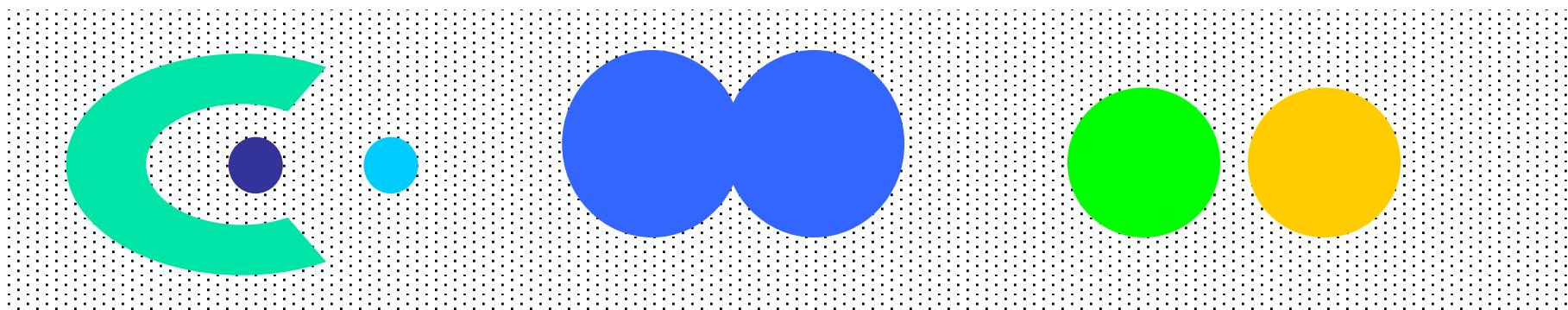
8 contiguous clusters



Types of Clusters: Density-Based

■ Density-based

- A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
- Used when the clusters are irregular or intertwined, and when noise and outliers are present.

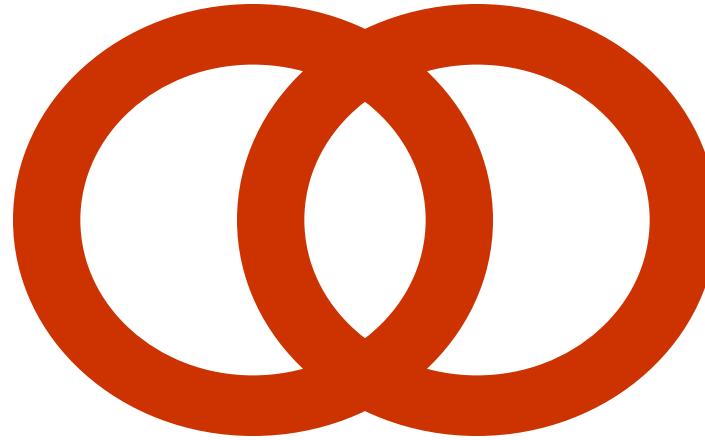


6 density-based clusters



Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
 - Finds clusters that share some common property or represent a particular concept.



2 Overlapping Circles



Clustering Algorithms

- K-means and its variants
- Hierarchical clustering
- Density-based clustering



K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K , must be specified
- The basic algorithm is very simple

-
- 1: Select K points as the initial centroids.
 - 2: **repeat**
 - 3: Form K clusters by assigning all points to the closest centroid.
 - 4: Recompute the centroid of each cluster.
 - 5: **until** The centroids don't change
-

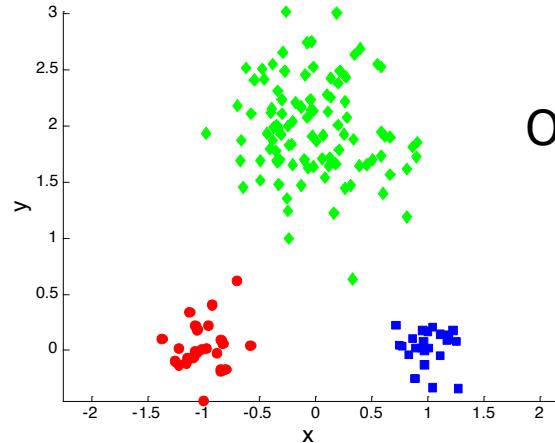


K-means Clustering – Details

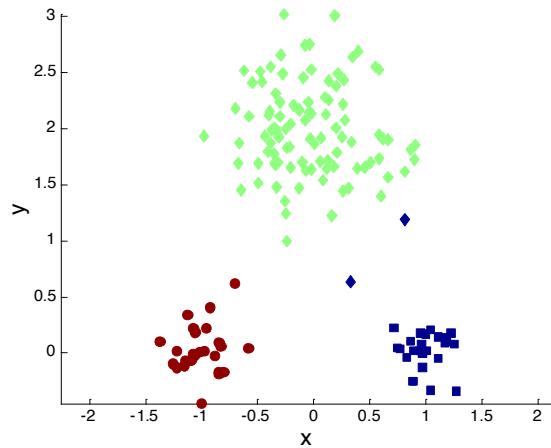
- Initial centroids are often chosen randomly.
 - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is generally measured with the Euclidean distance
 - The centroid (mean point) minimizes the mean squared Euclidean distance to all points in the cluster
- Most of the convergence happens in the first few iterations.
 - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is $O(n * K * I * d)$
 - n = number of points, K = number of clusters,
 I = number of iterations, d = number of attributes



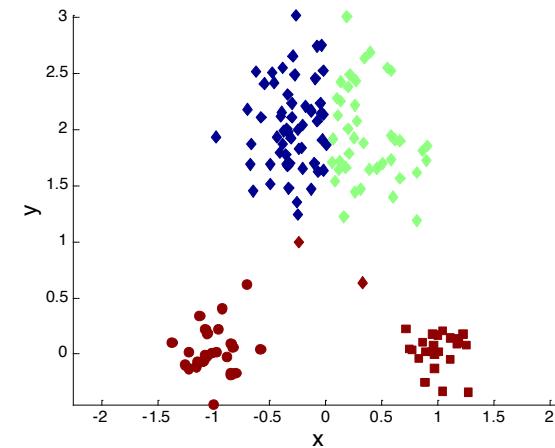
Two different K-means Clusterings



Original Points



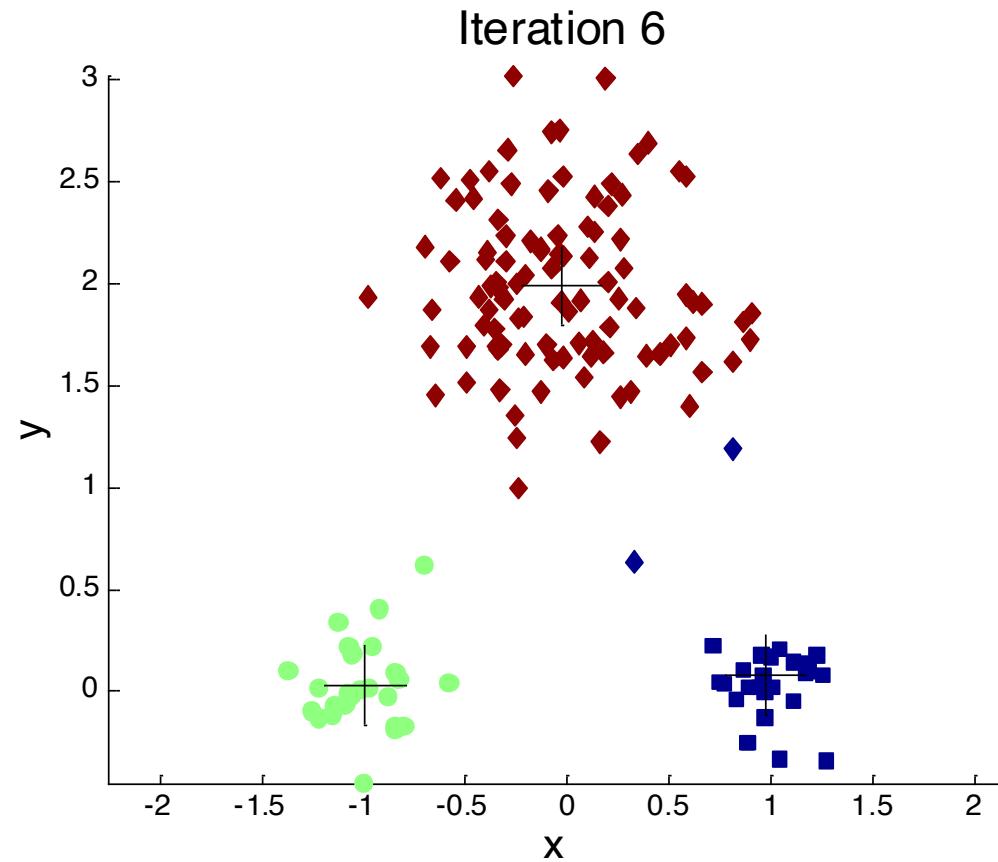
Optimal Clustering



Sub-optimal Clustering

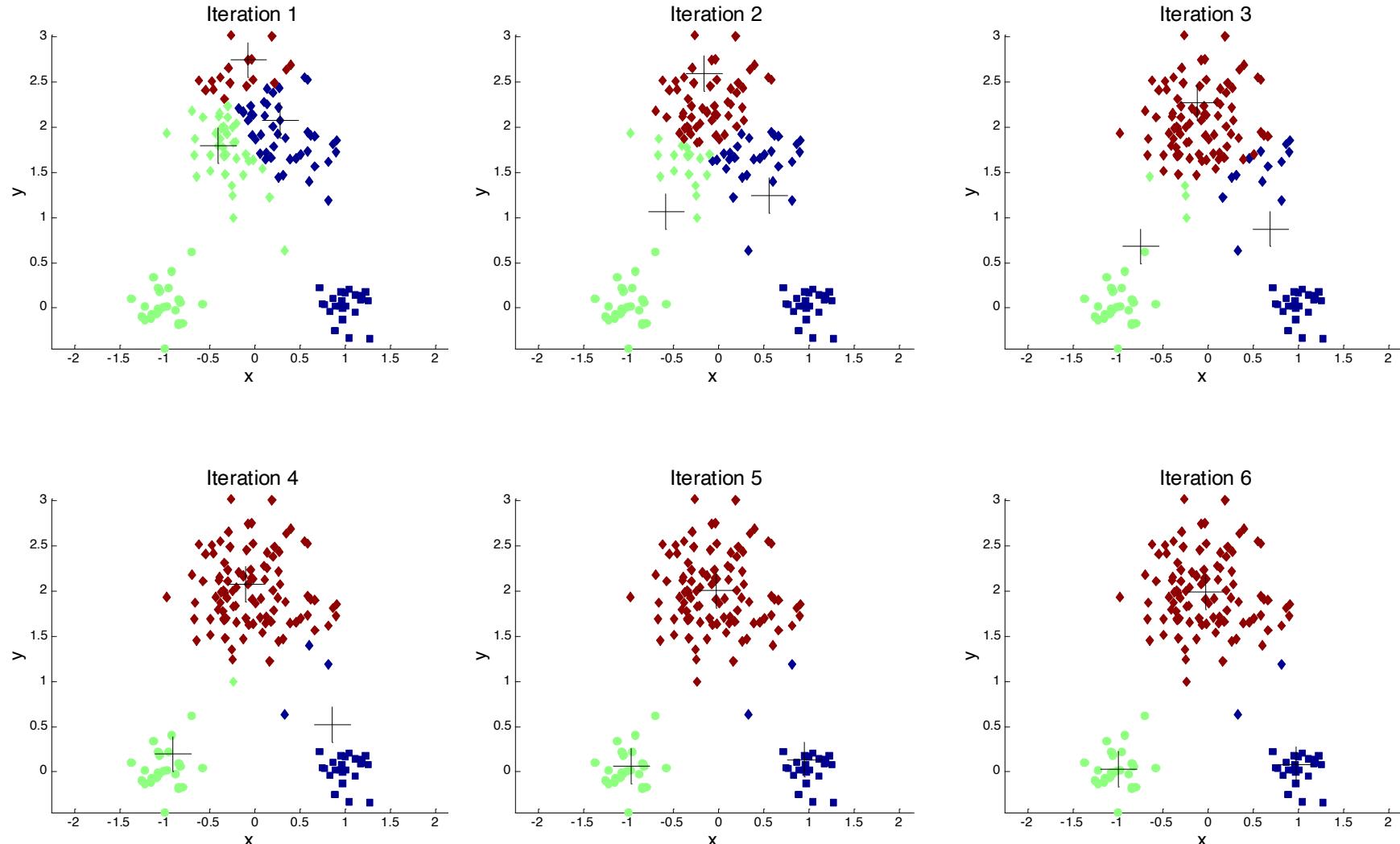


Importance of Choosing Initial Centroids



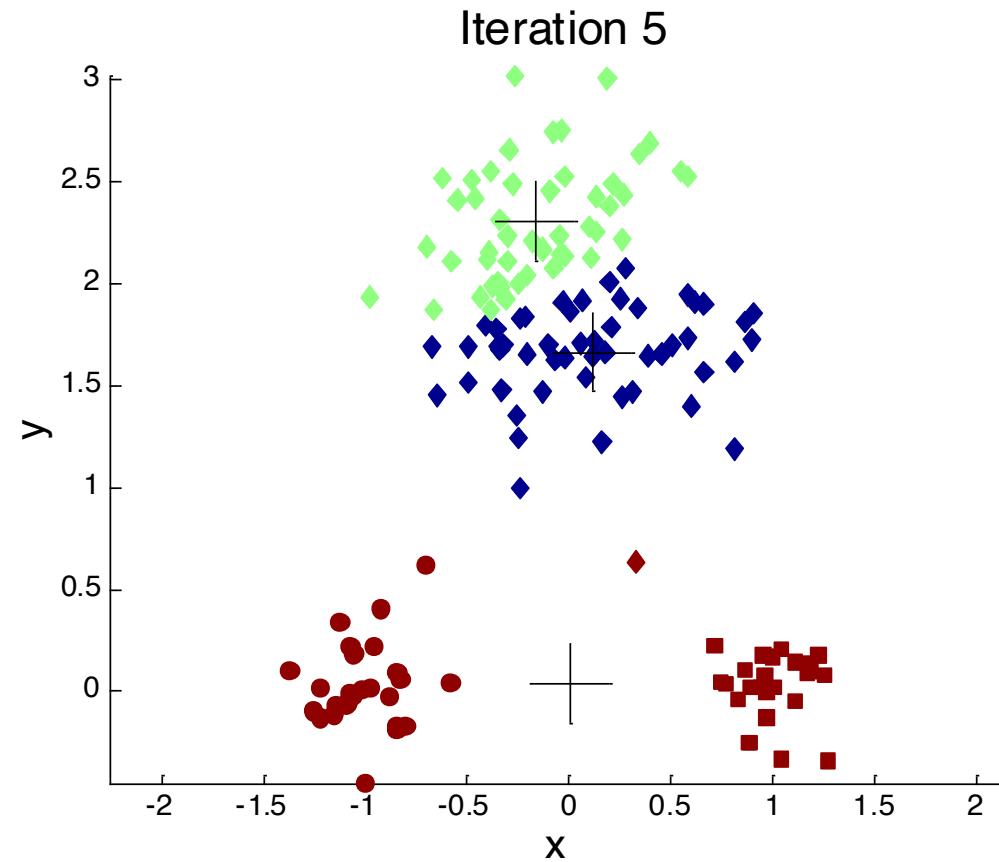


Importance of Choosing Initial Centroids



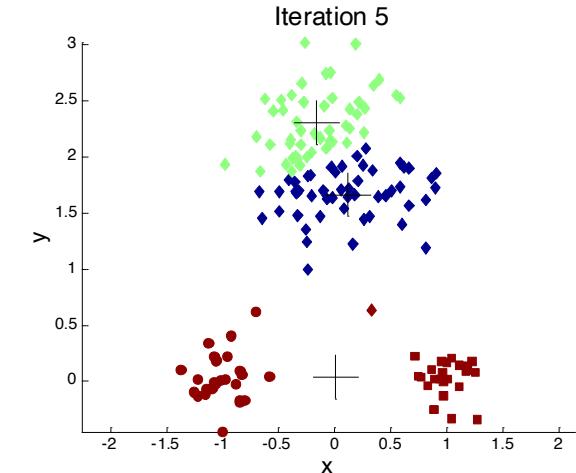
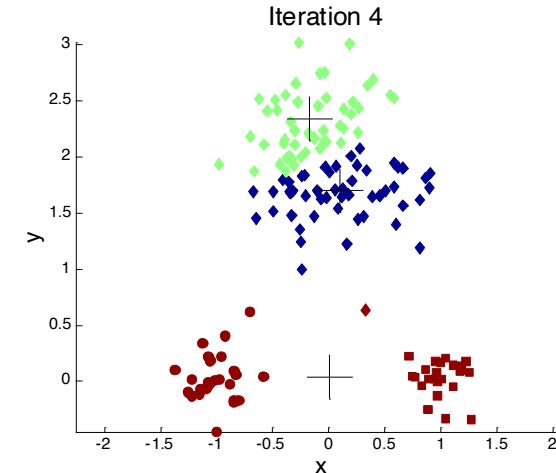
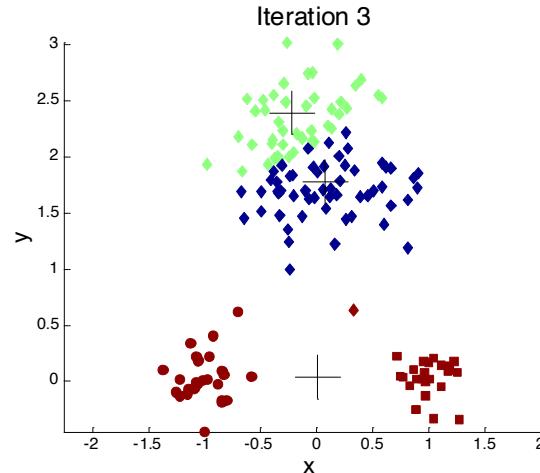
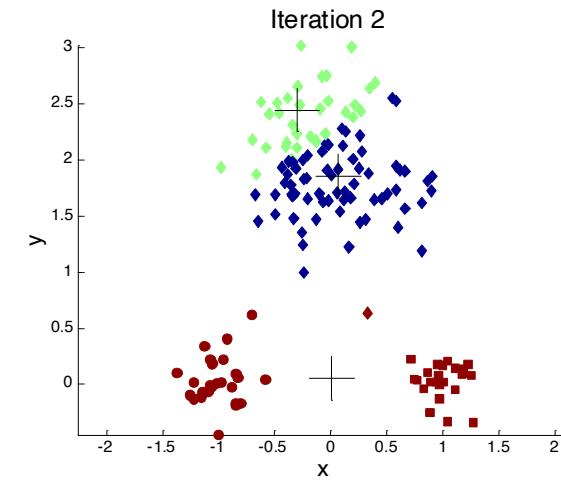
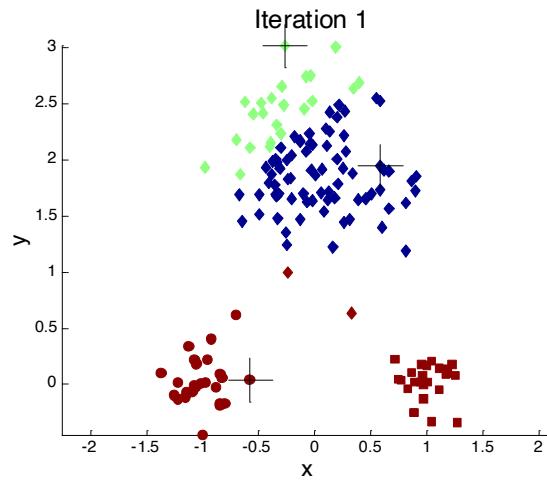


Importance of Choosing Initial Centroids





Importance of Choosing Initial Centroids





Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster
 - To get SSE, we square these errors and sum them.

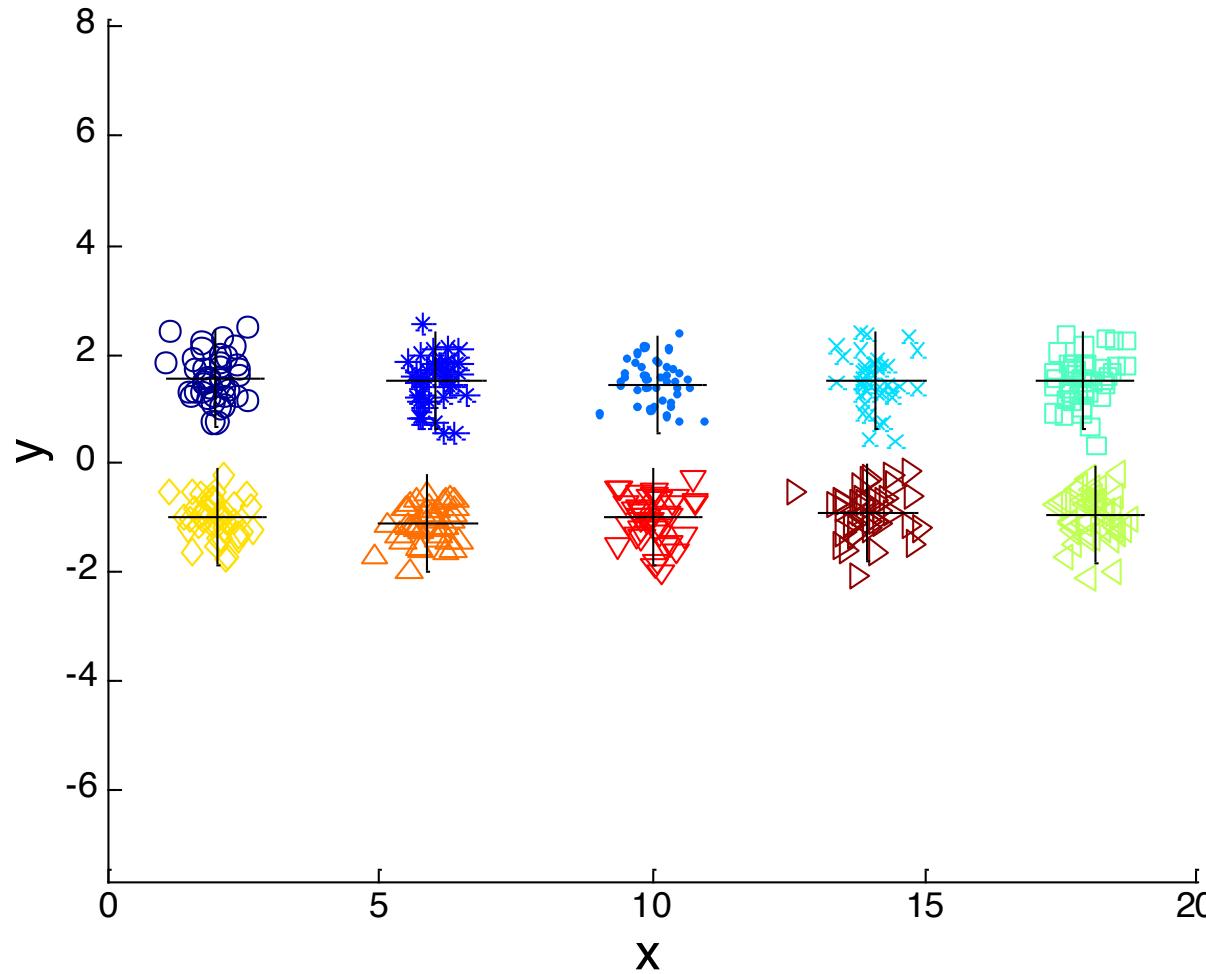
$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - can show that m_i corresponds to the center (mean) of the cluster
- Given two clusters, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K, the number of clusters
 - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K



10 Clusters Example

Iteration 4

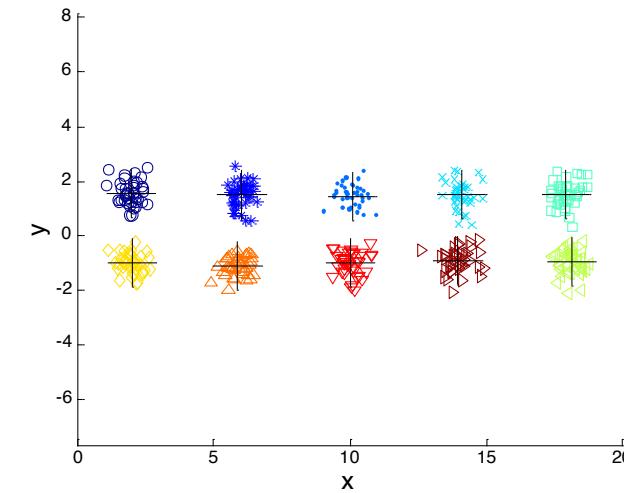
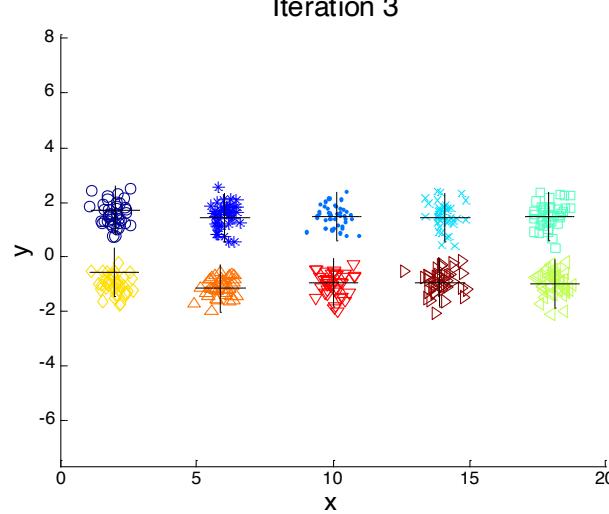
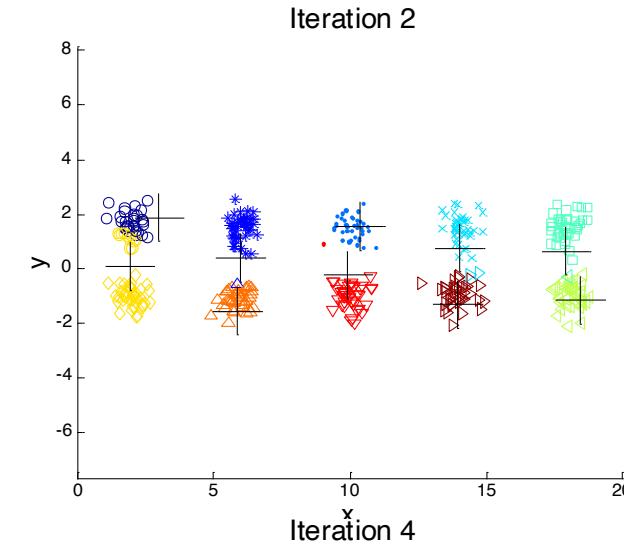
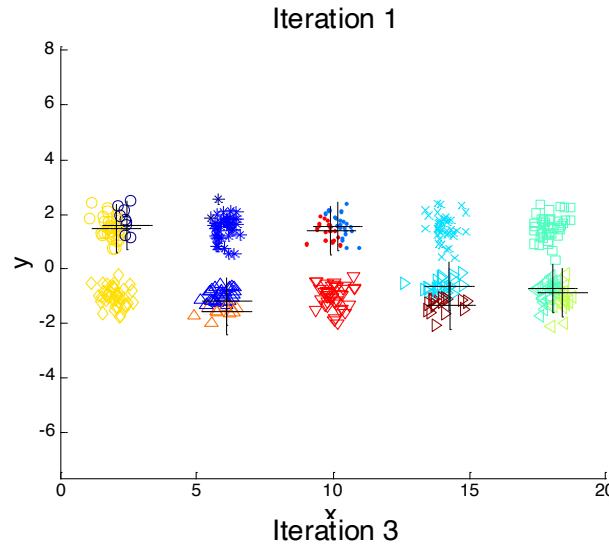


Starting with two initial centroids in one cluster of each pair of clusters

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



10 Clusters Example



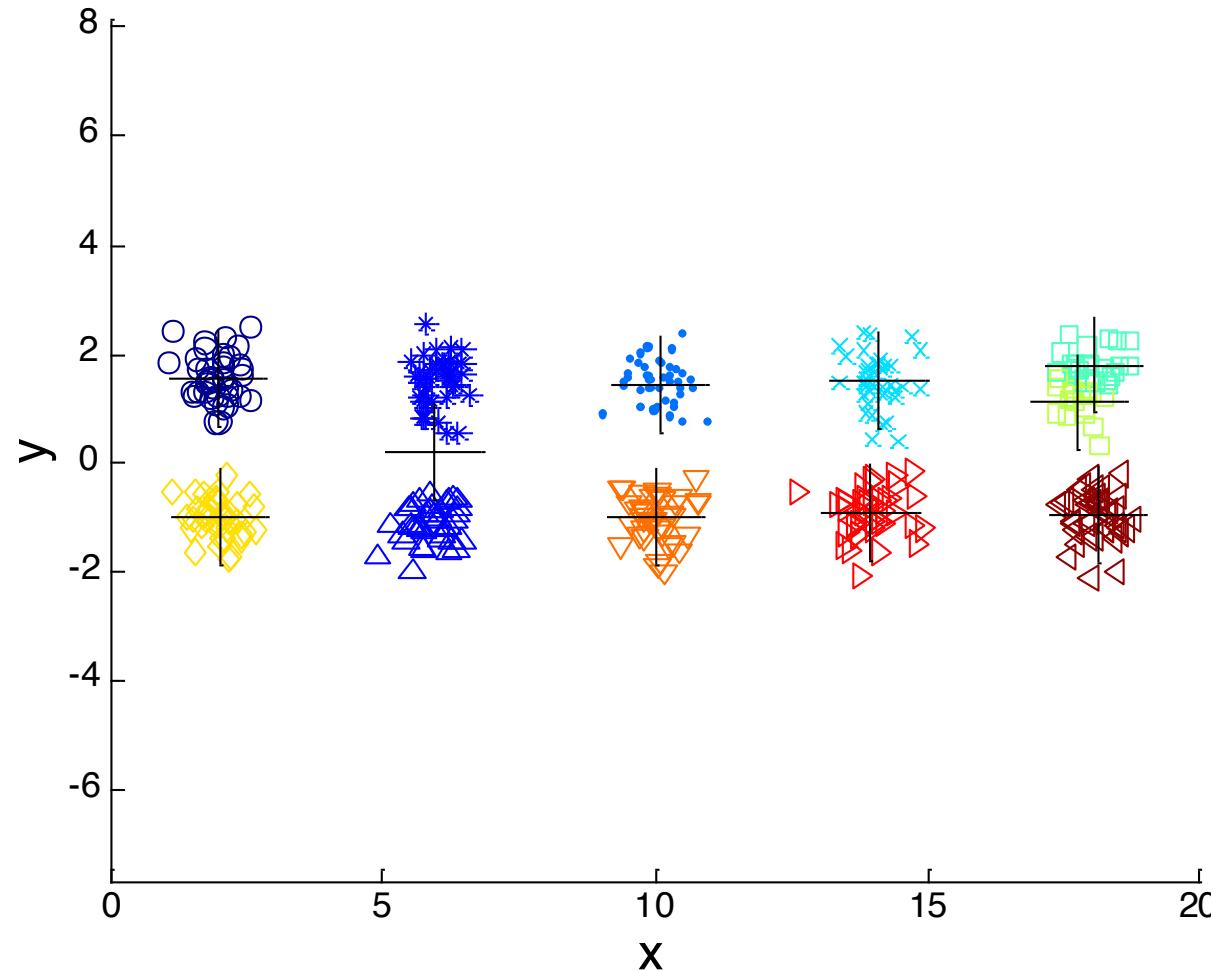
Starting with two initial centroids in one cluster of each pair of clusters

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



10 Clusters Example

Iteration 4



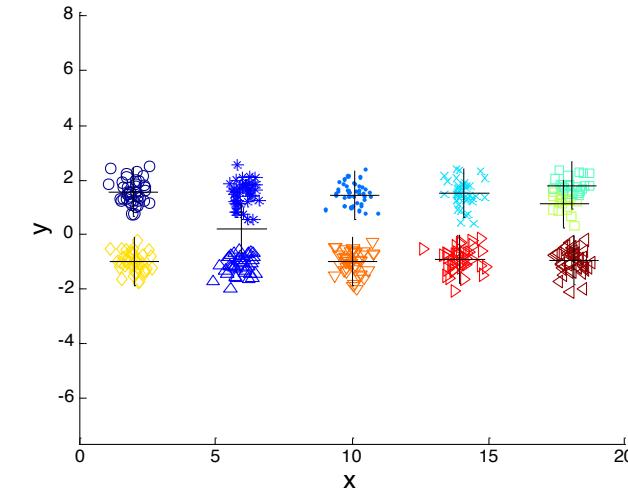
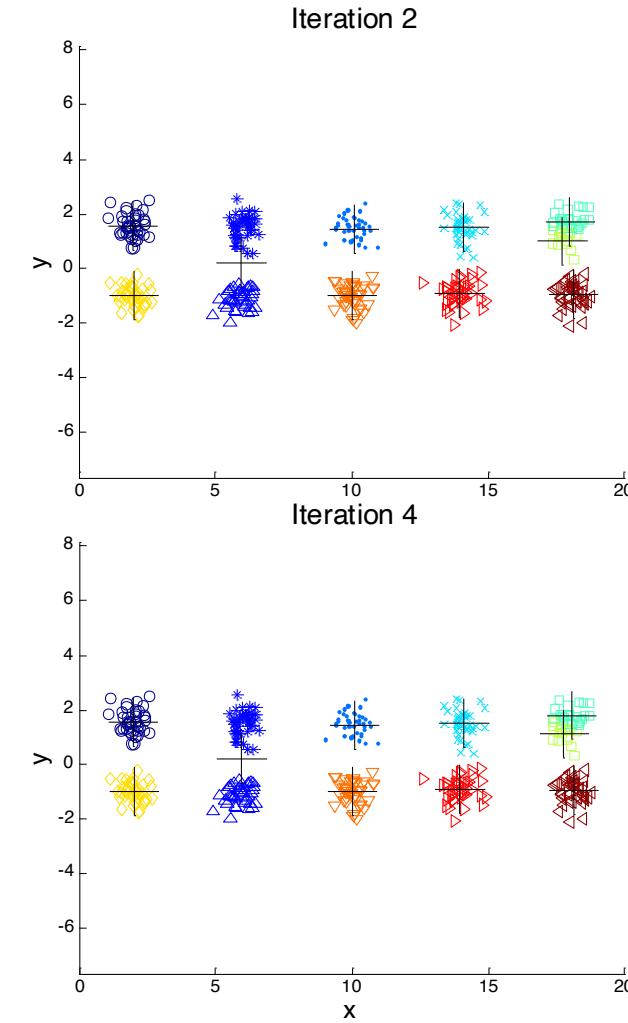
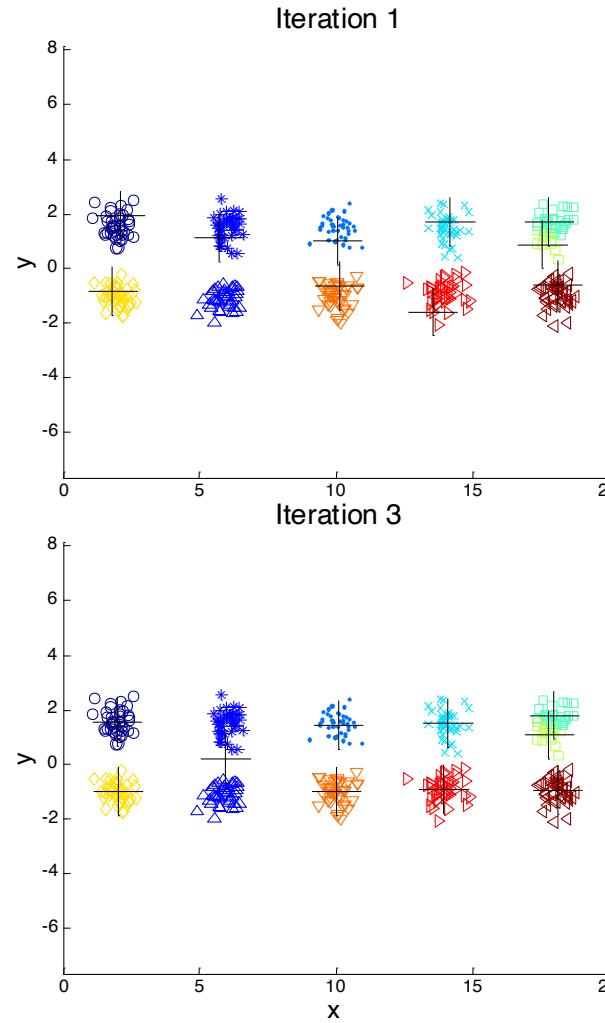
**DB
M**

Starting with some pairs of clusters having three initial centroids, while other have only one.

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



10 Clusters Example





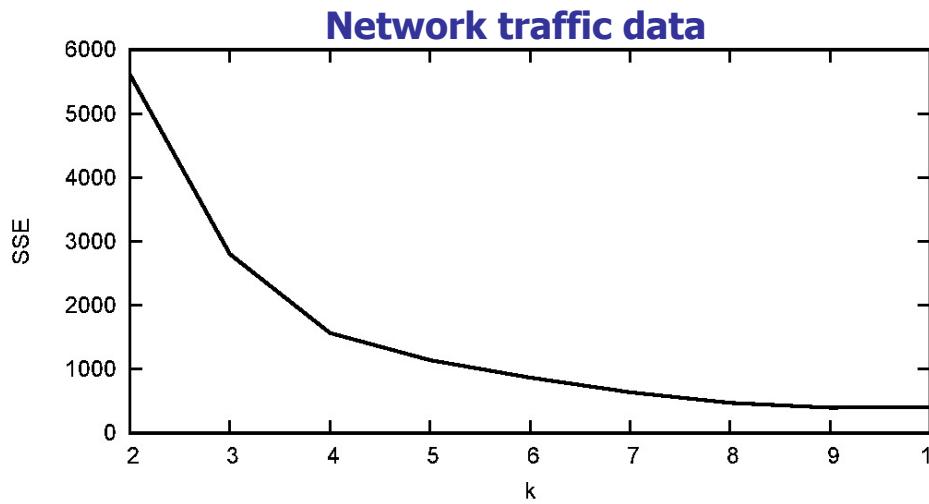
Solutions to Initial Centroids Problem

- Multiple runs
 - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select spread out centroids
 - Pick 1st one randomly, then choose the others to be “distant” from the previous ones (K-means++)
- Postprocessing
- Bisecting K-means
 - Not as susceptible to initialization issues



K-means parameter setting

- Elbow graph (Knee approach)
 - Plotting the quality measure trend (e.g., SSE) against K
 - Choosing the value of K
 - the gain from adding a centroid is negligible
 - The reduction of the quality measure is not interesting anymore





Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters
- Several strategies
 - Choose the point that contributes most to SSE
 - Choose a point from the cluster with the highest SSE
 - If there are several empty clusters, the above can be repeated several times.



Pre-processing and Post-processing

- Pre-processing
 - Normalize the data
 - Eliminate outliers
- Post-processing
 - Eliminate small clusters that may represent outliers
 - Split 'loose' clusters, i.e., clusters with relatively high SSE
 - Merge clusters that are 'close' and that have relatively low SSE
 - Can use these steps during the clustering process



Bisecting K-means

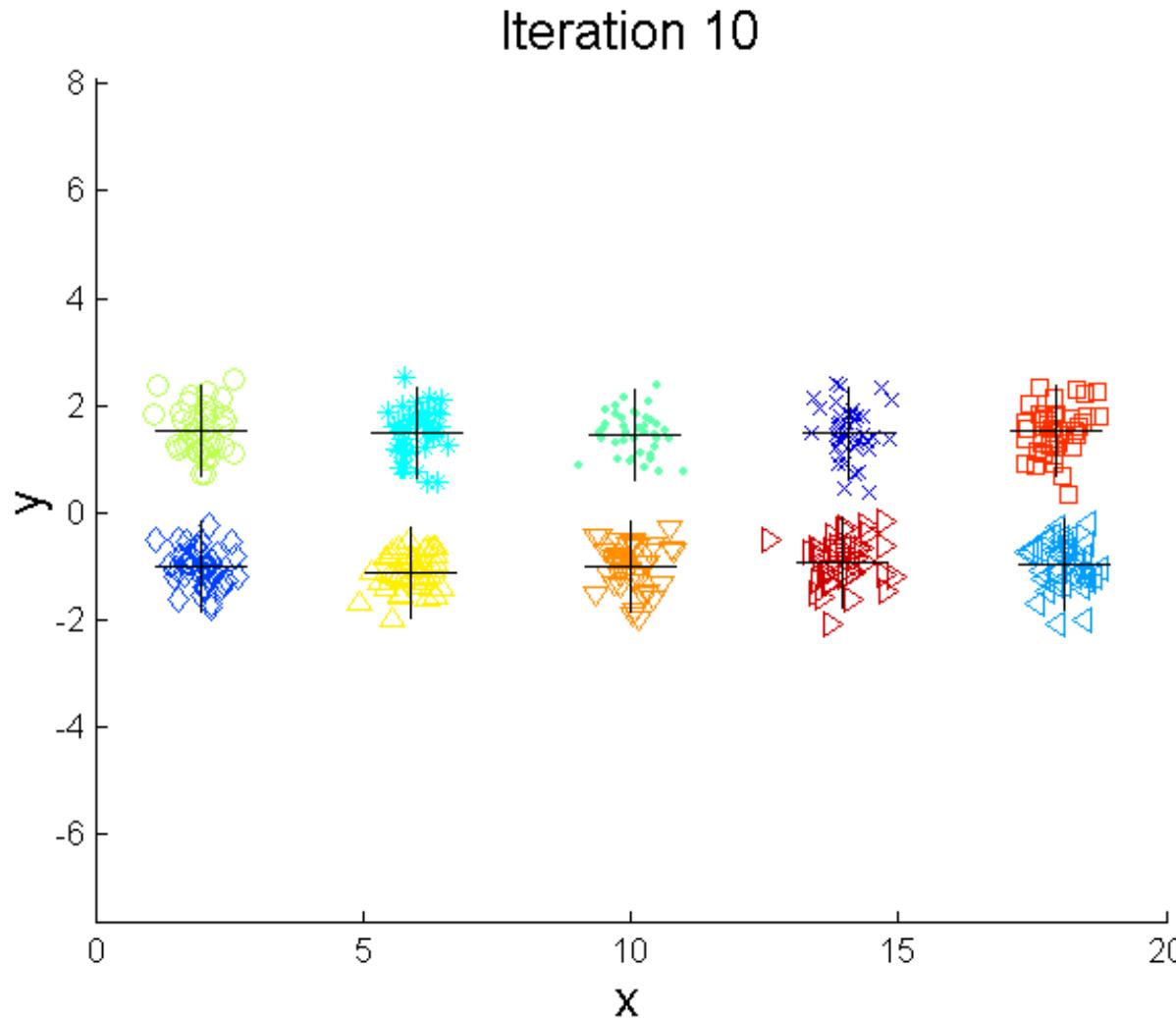
■ Bisecting K-means algorithm

- Variant of K-means that can produce a partitional or a hierarchical clustering

```
1: Initialize the list of clusters to contain the cluster containing all points.  
2: repeat  
3:   Select a cluster from the list of clusters  
4:   for  $i = 1$  to number_of_iterations do  
5:     Bisect the selected cluster using basic K-means  
6:   end for  
7:   Add the two clusters from the bisection with the lowest SSE to the list of clusters.  
8: until Until the list of clusters contains  $K$  clusters
```



Bisecting K-means Example



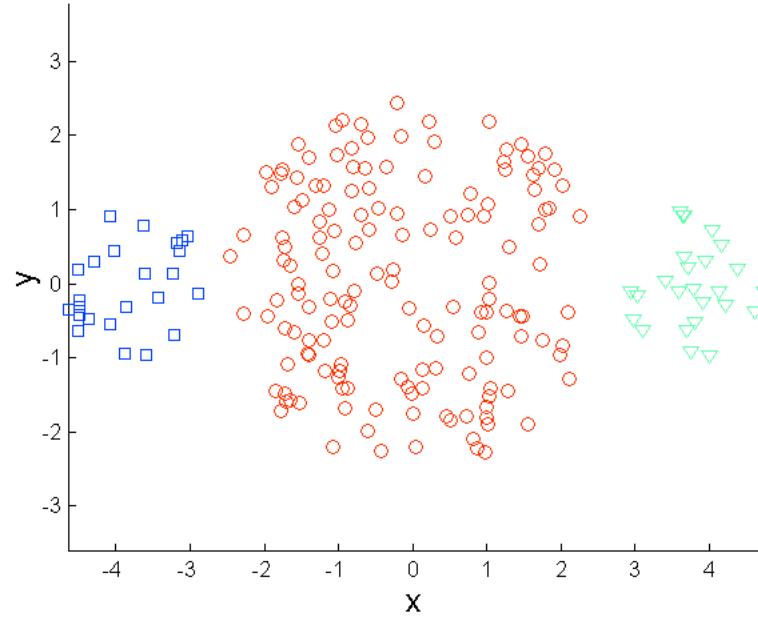


Limitations of K-means

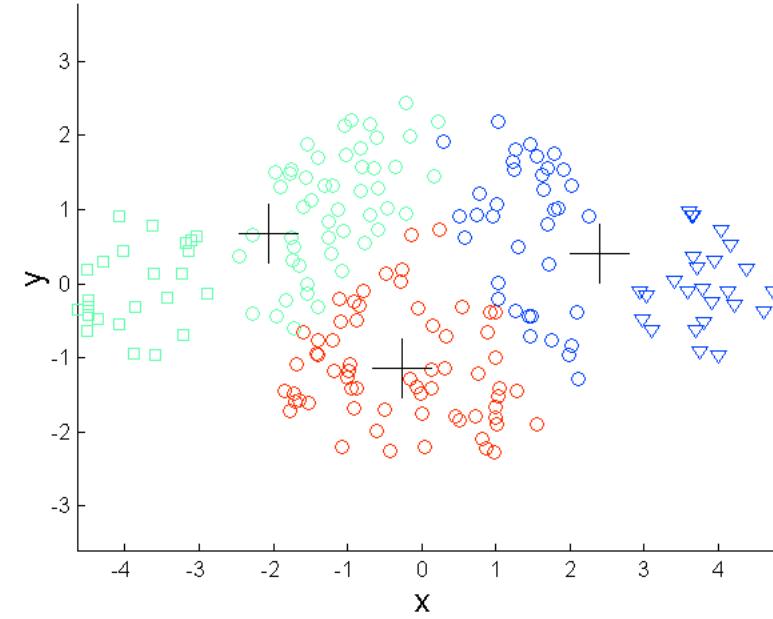
- K-means has problems when clusters are of differing
 - Sizes
 - Densities
 - Non-globular shapes
- K-means has problems when the data contains outliers.



Limitations of K-means: Differing Sizes



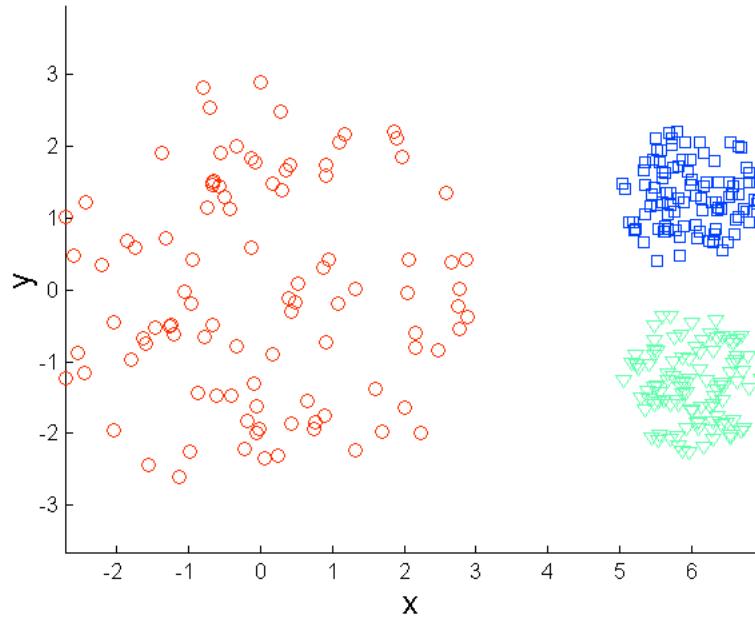
Original Points



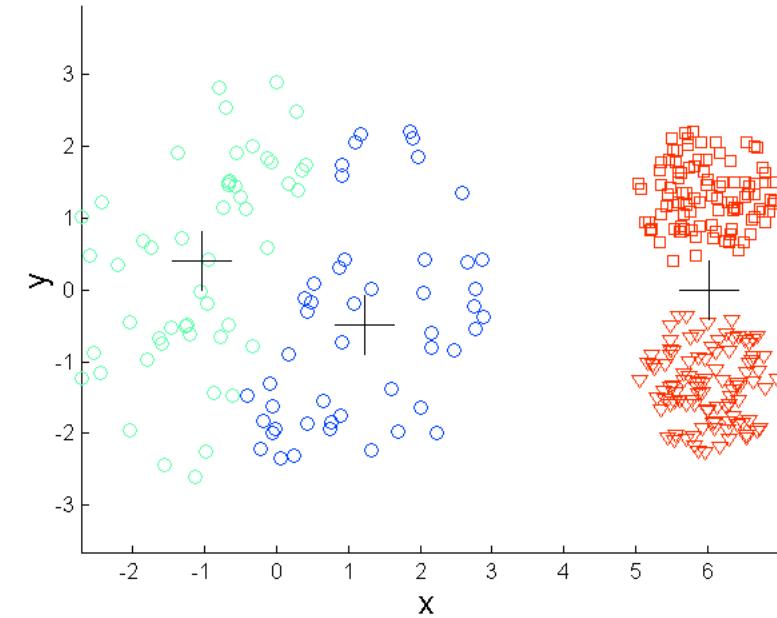
K-means (3 Clusters)



Limitations of K-means: Differing Density



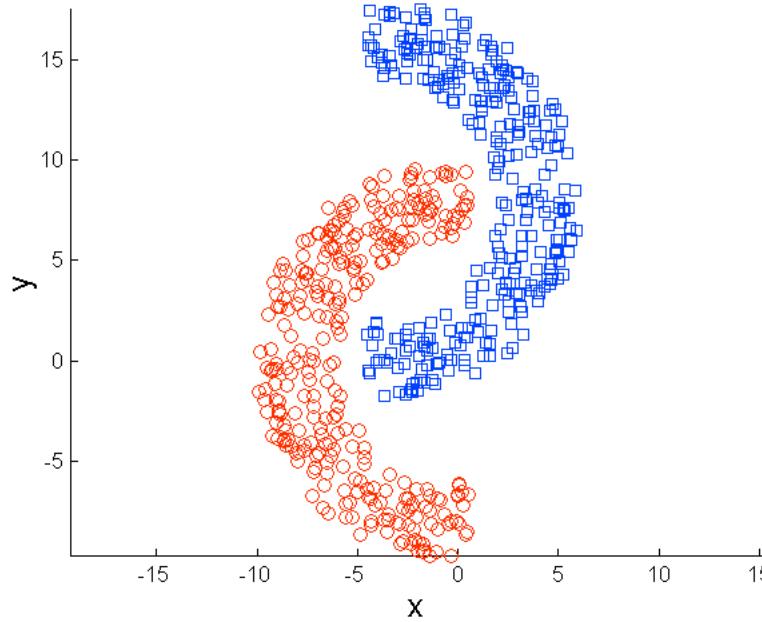
Original Points



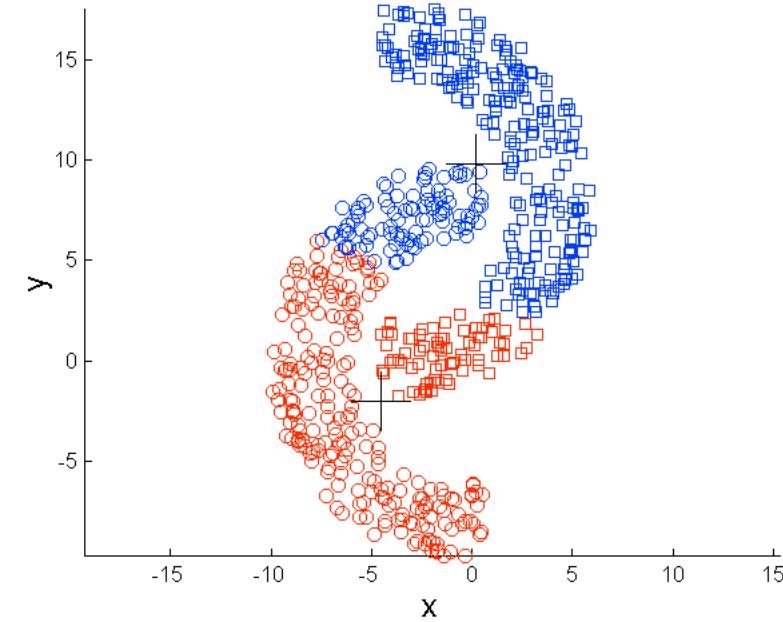
K-means (3 Clusters)



Limitations of K-means: Non-globular Shapes



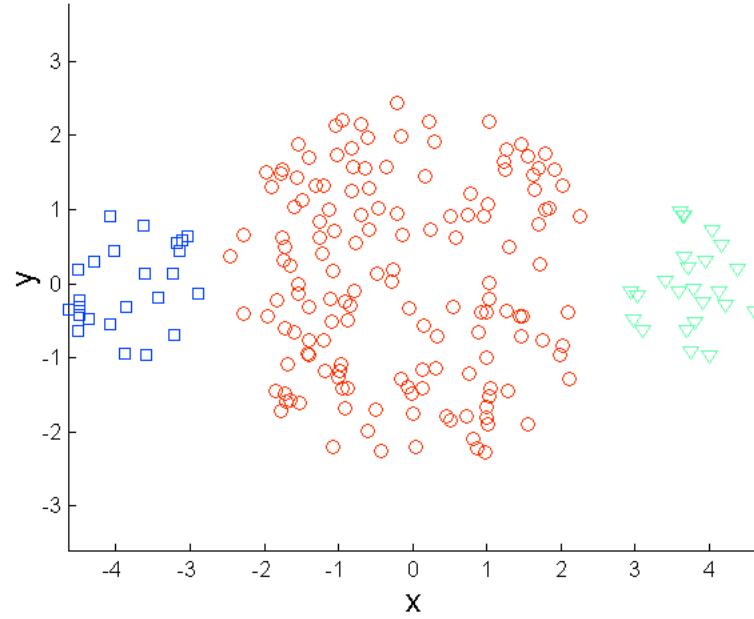
Original Points



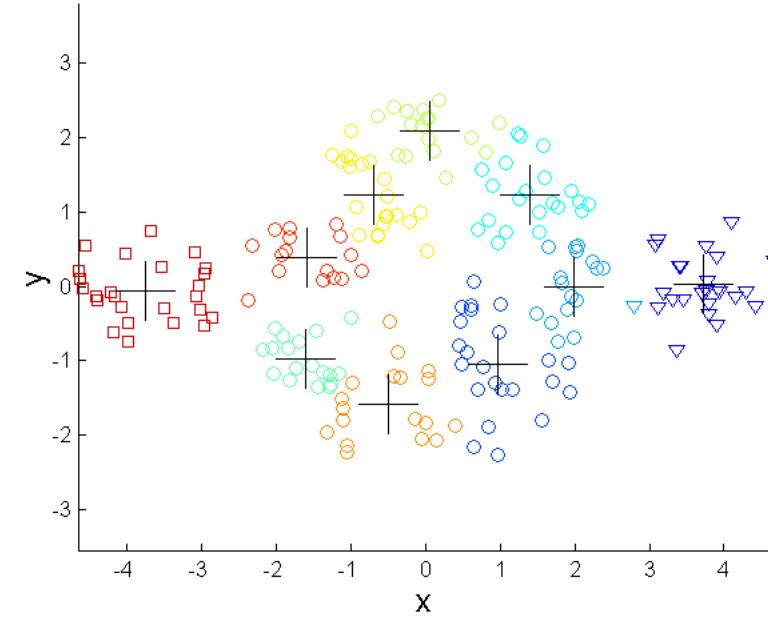
K-means (2 Clusters)



Overcoming K-means Limitations



Original Points



K-means Clusters

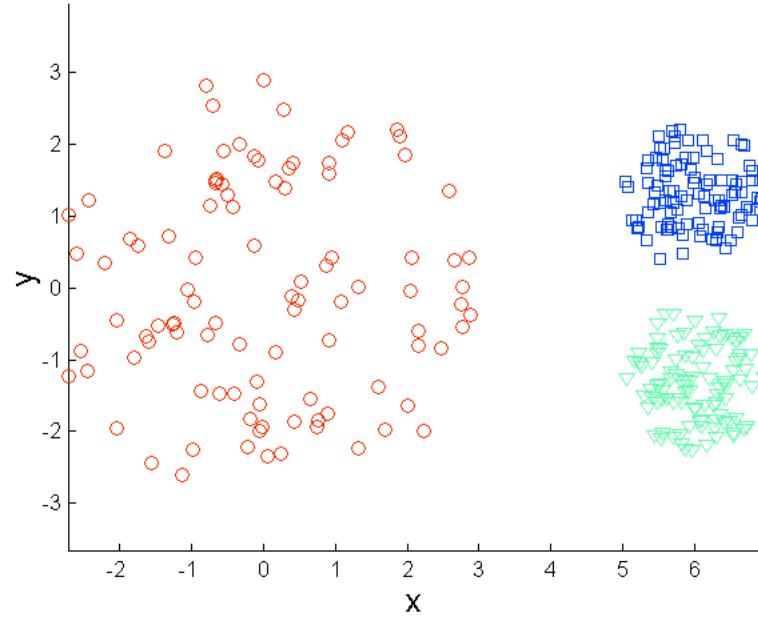
One solution is to use many clusters.

Find parts of clusters, but need to put together.

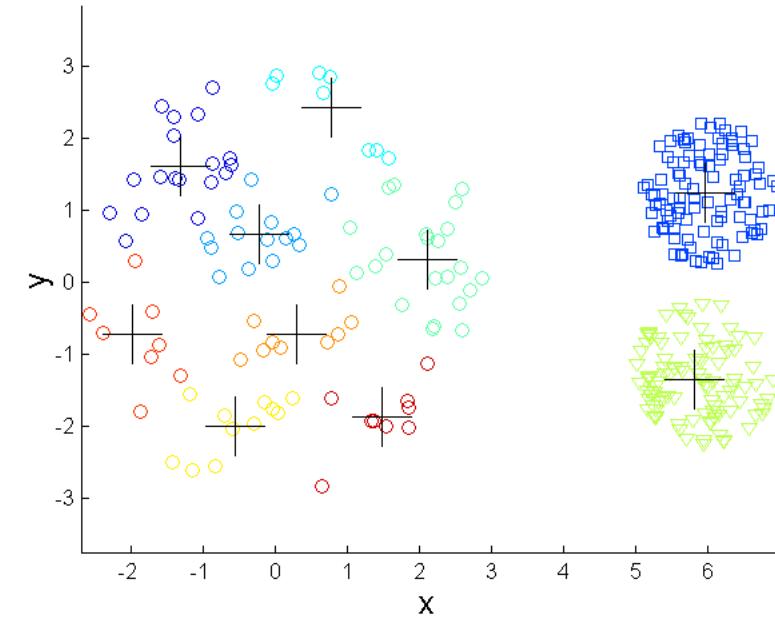
From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



Overcoming K-means Limitations



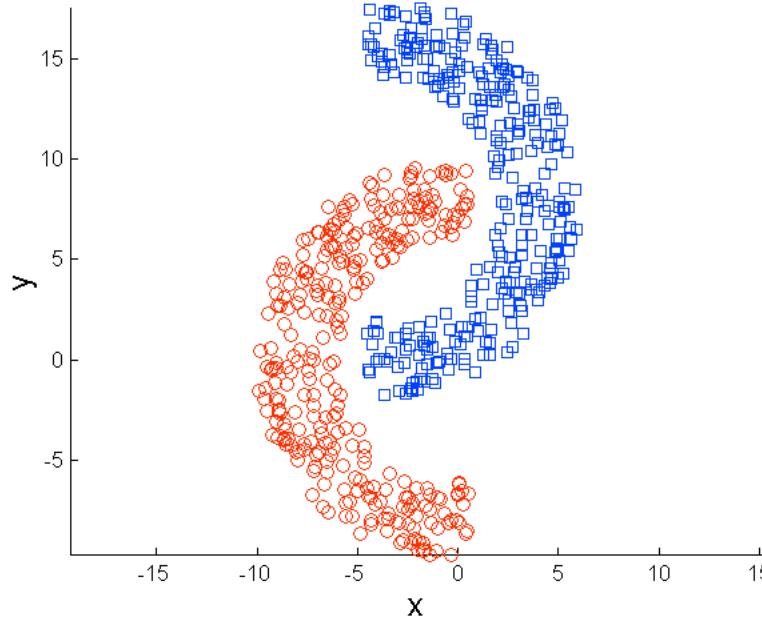
Original Points



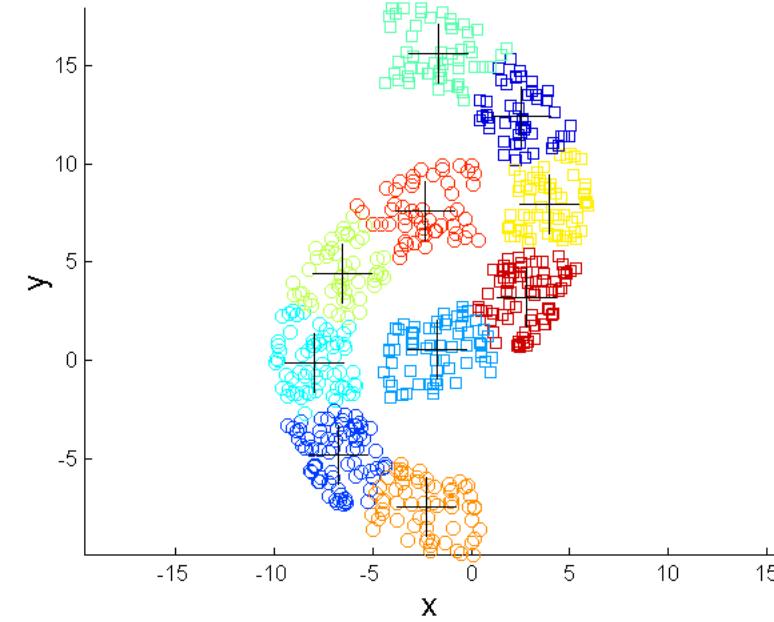
K-means Clusters



Overcoming K-means Limitations



Original Points

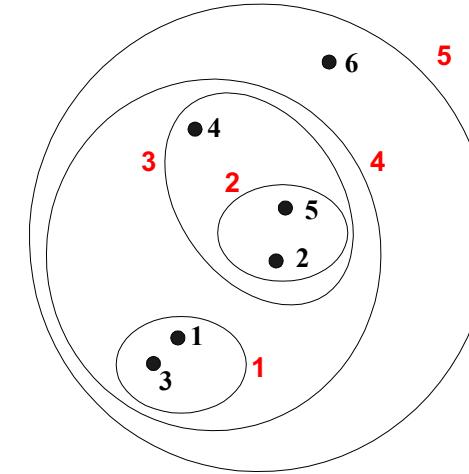
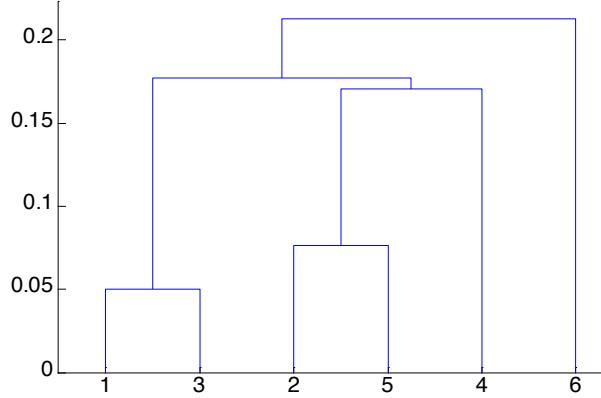


K-means Clusters



Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
 - A tree like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
 - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- They may correspond to meaningful taxonomies
 - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)



Hierarchical Clustering

- Two main types of hierarchical clustering
 - Agglomerative:
 - Start with the points as individual clusters
 - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
 - Divisive:
 - Start with one, all-inclusive cluster
 - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
 - Merge or split one cluster at a time



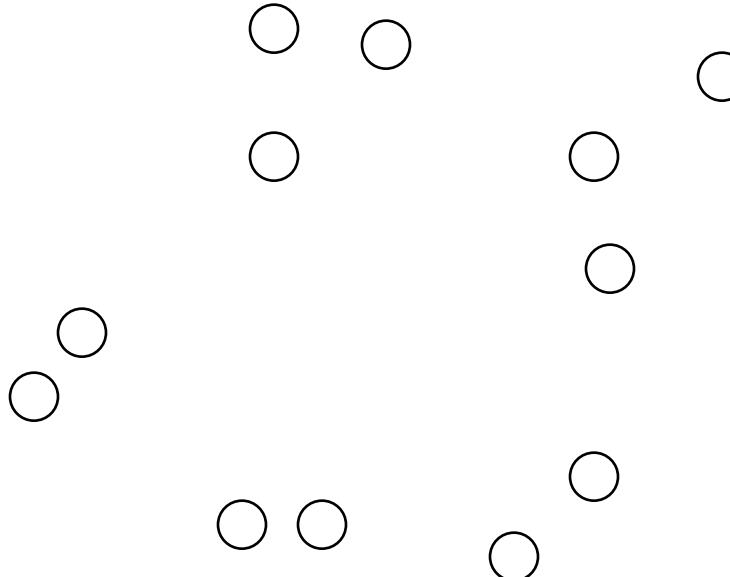
Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
 1. Compute the proximity matrix
 2. Let each data point be a cluster
 3. **Repeat**
 4. Merge the two closest clusters
 5. Update the proximity matrix
 6. **Until** only a single cluster remains
 - Key operation is the computation of the proximity of two clusters
 - Different approaches to defining the distance between clusters distinguish the different algorithms



Starting Situation

- Start with clusters of individual points and a proximity matrix



	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						

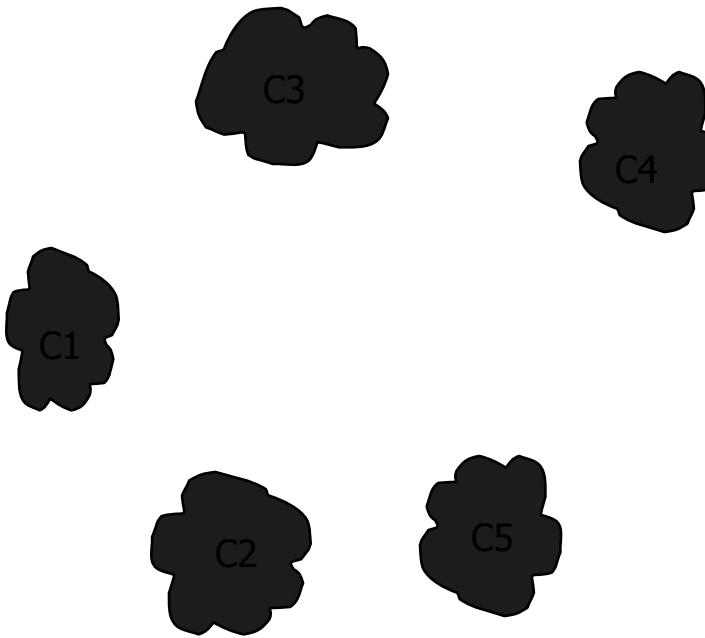
Proximity Matrix

p1 p2 p3 p4 ... p9 p10 p11 p12



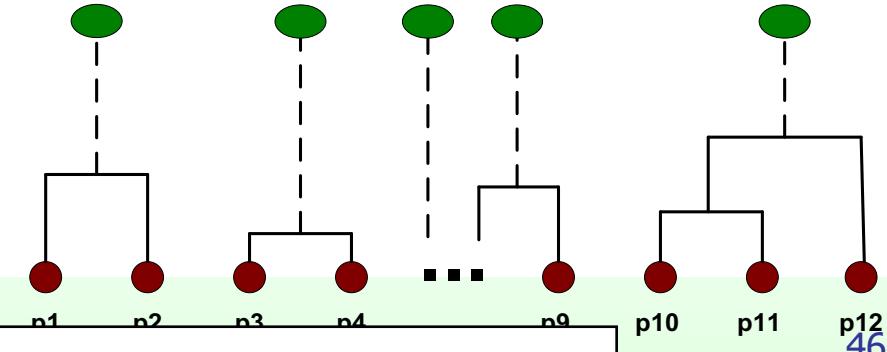
Intermediate Situation

- After some merging steps, we have some clusters



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

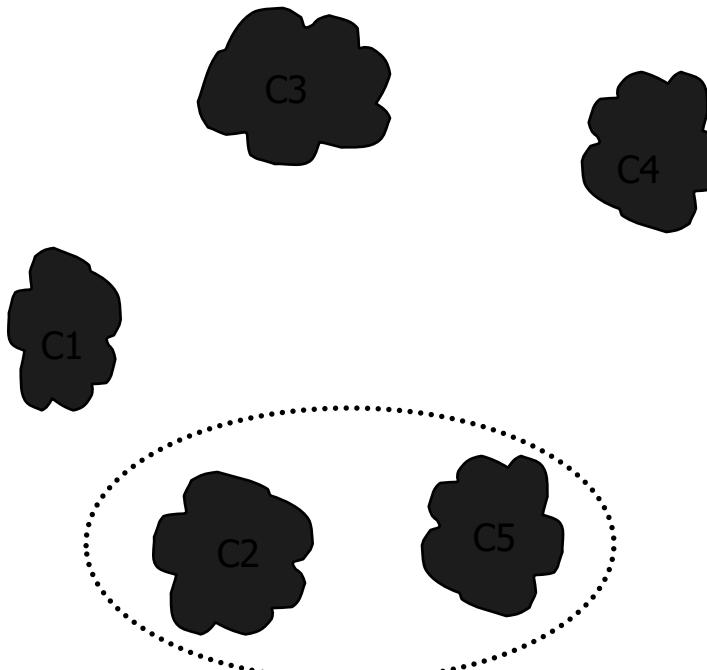
Proximity Matrix





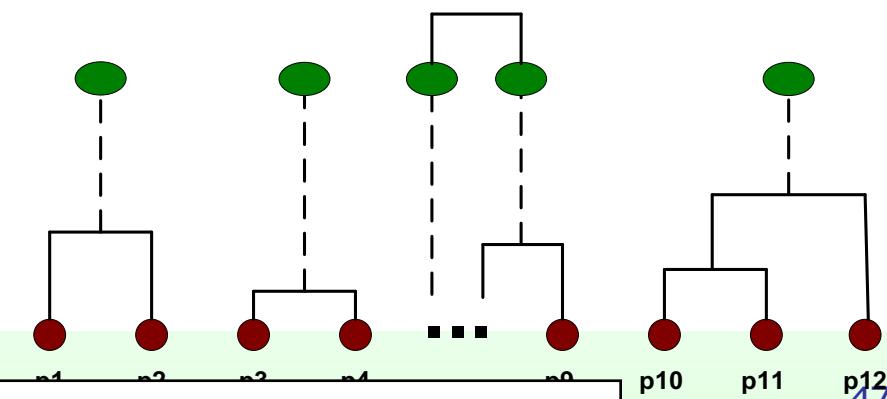
Intermediate Situation

- We want to merge the two closest clusters (C_2 and C_5) and update the proximity matrix.



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

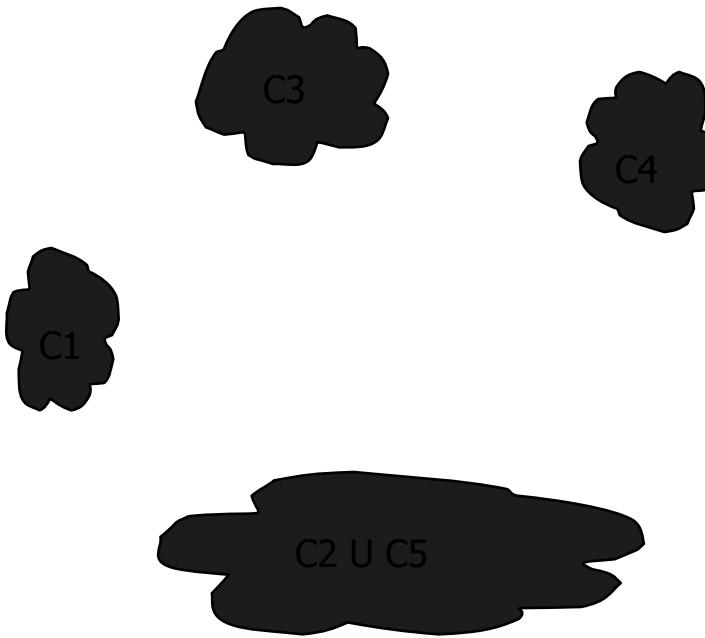
Proximity Matrix





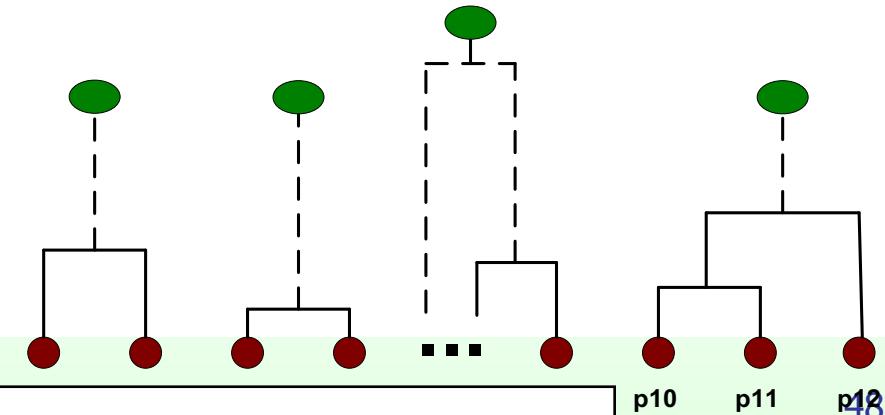
After Merging

- The question is “How do we update the proximity matrix?”



	C2 U C5	C1	C5	C3	C4
C1	?				
C2 U C5	?	?	?	?	?
C3		?			
C4		?			

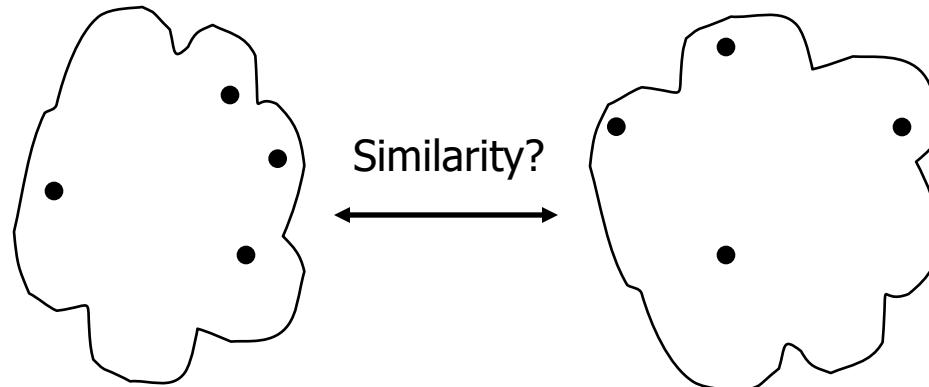
Proximity Matrix



From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006



How to Define Inter-Cluster Similarity



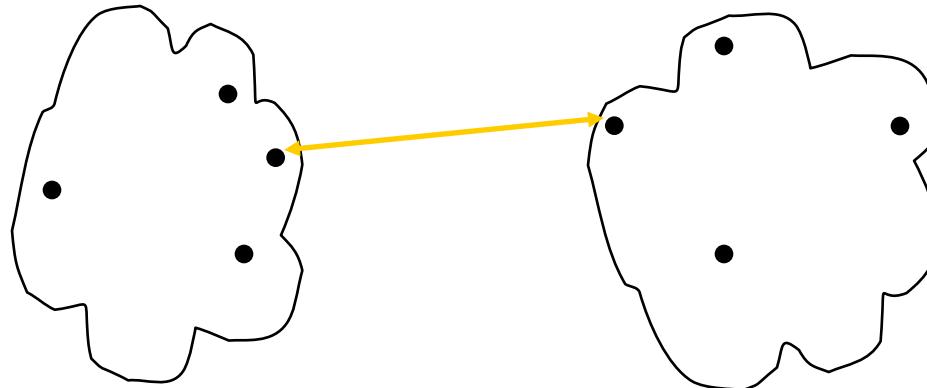
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix



How to Define Inter-Cluster Similarity



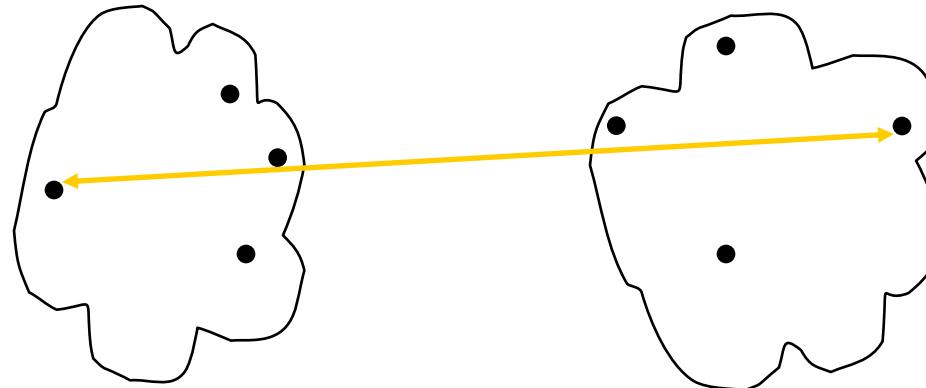
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix



How to Define Inter-Cluster Similarity



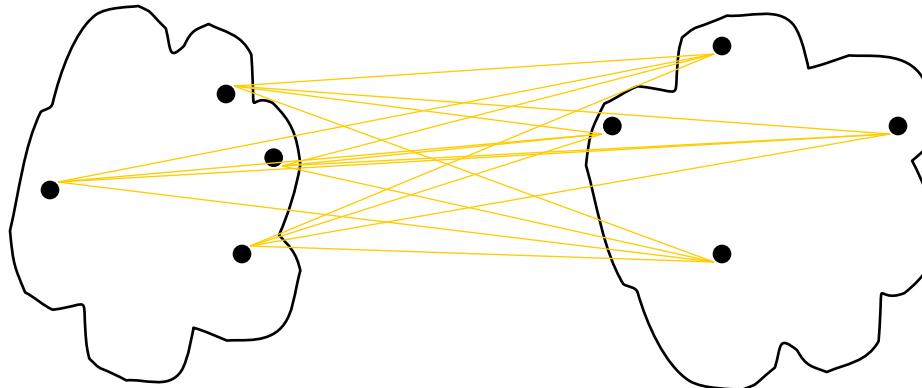
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

Proximity Matrix



How to Define Inter-Cluster Similarity



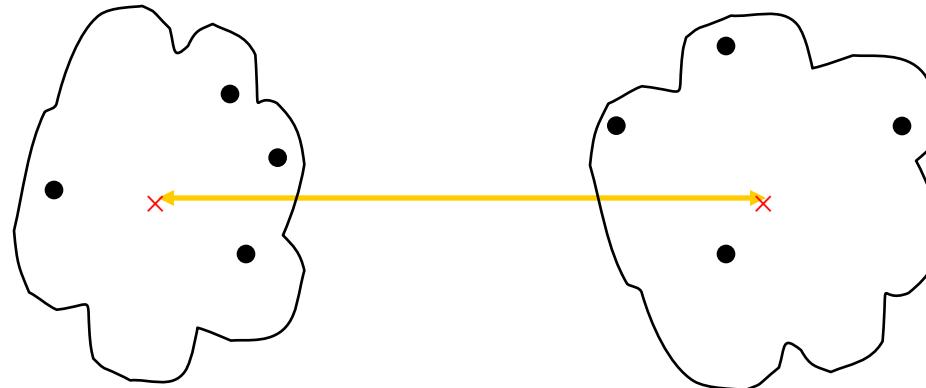
- MIN
- MAX
- **Group Average**
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix



How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

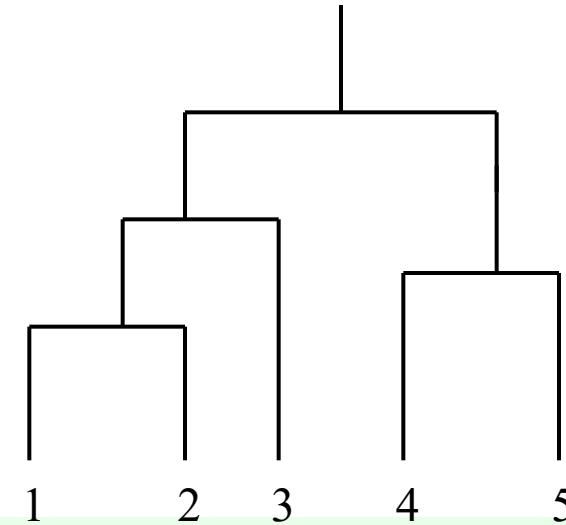
Proximity Matrix



Cluster Similarity: MIN or Single Link

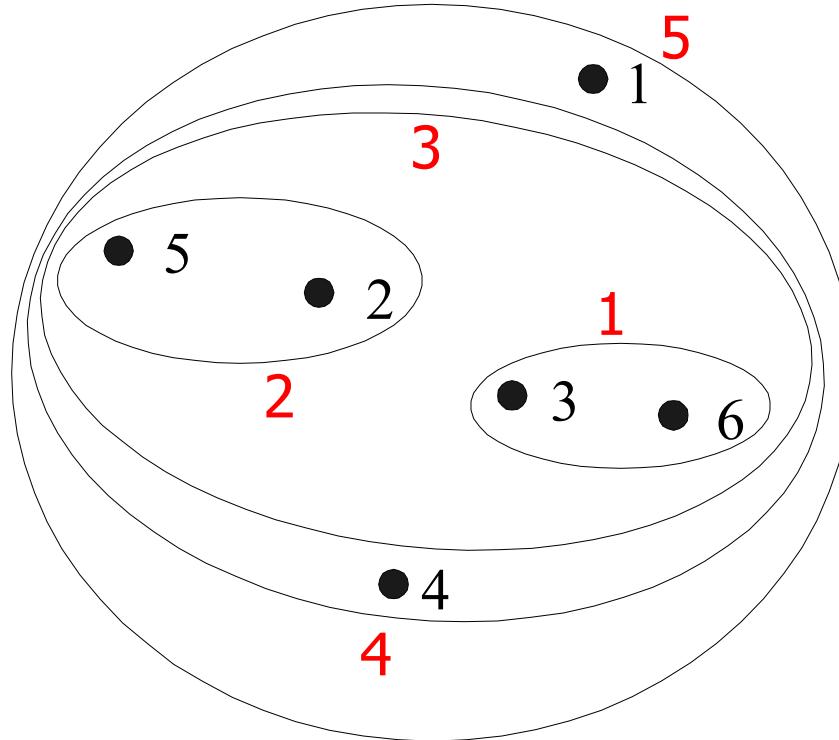
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
 - Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

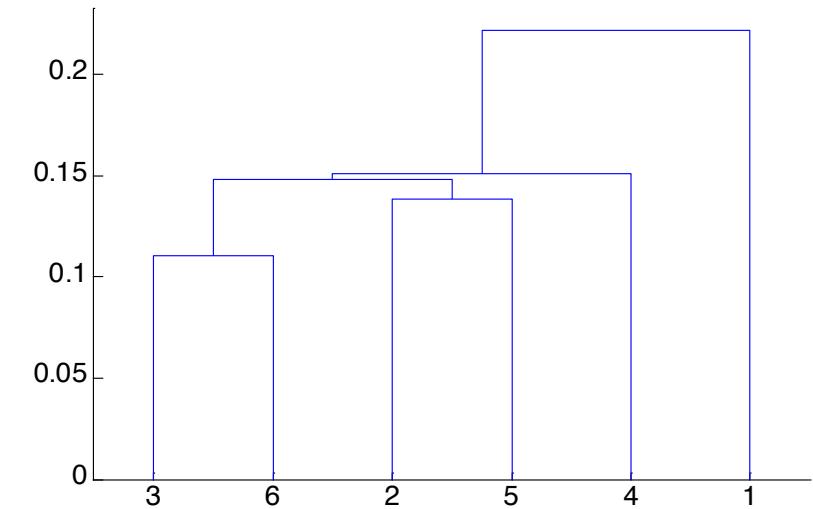




Hierarchical Clustering: MIN



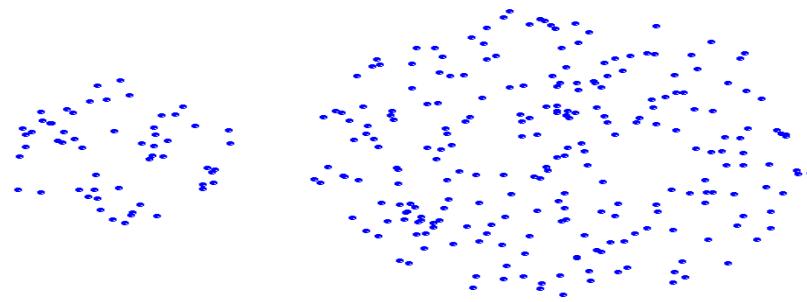
Nested Clusters



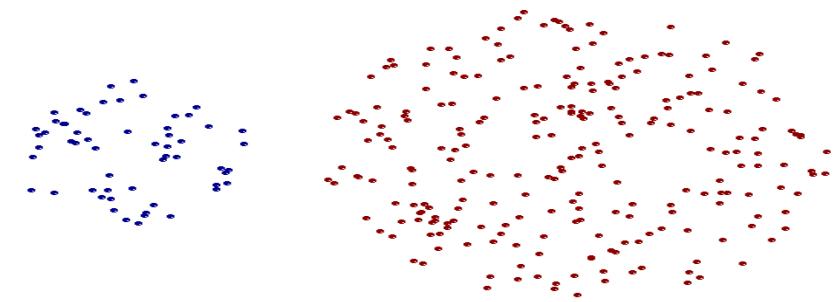
Dendrogram



Strength of MIN



Original Points

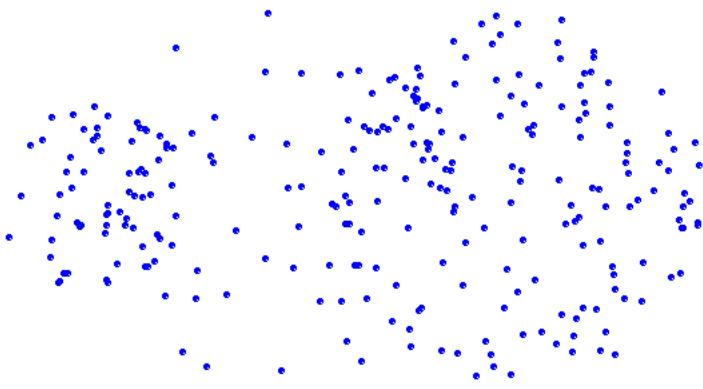


Two Clusters

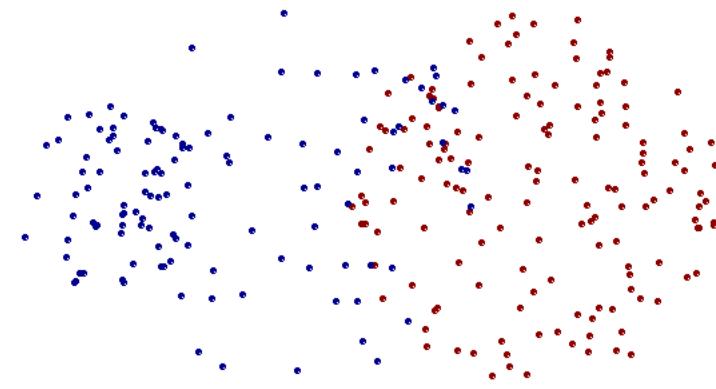
- Can handle non-elliptical shapes



Limitations of MIN



Original Points



Two Clusters

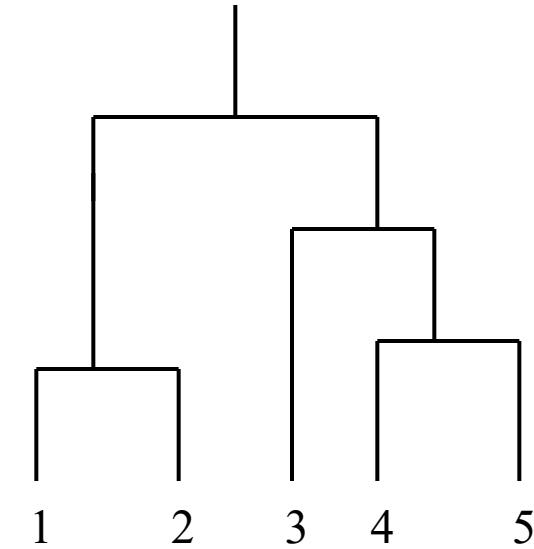
- Sensitive to noise and outliers



Cluster Similarity: MAX or Complete Linkage

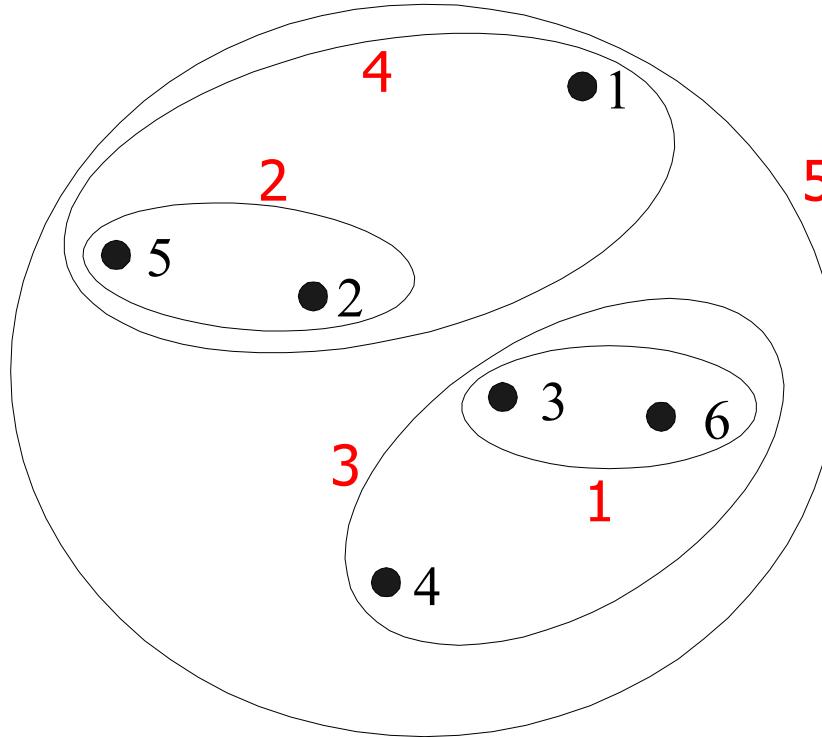
- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
 - Determined by all pairs of points in the two clusters

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

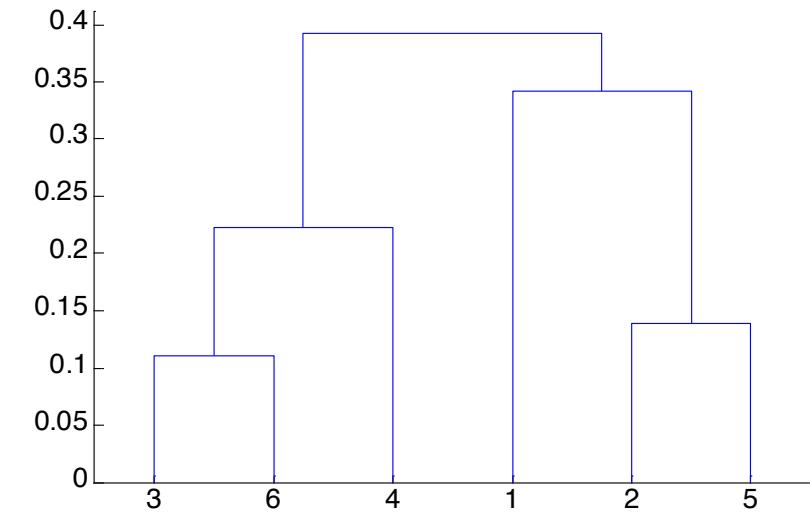




Hierarchical Clustering: MAX



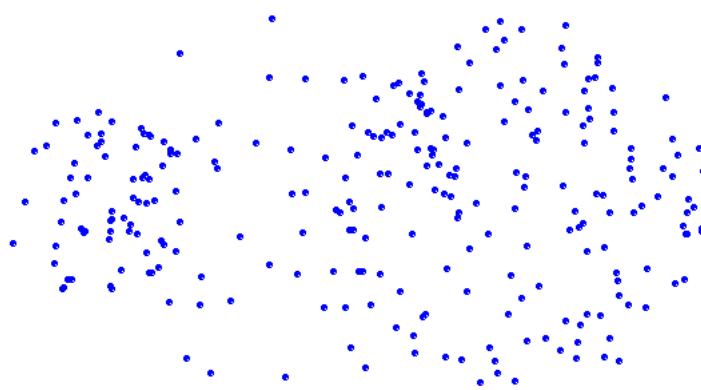
Nested Clusters



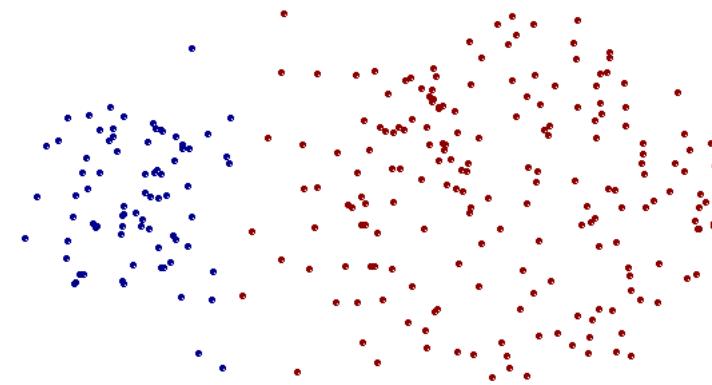
Dendrogram



Strength of MAX



Original Points

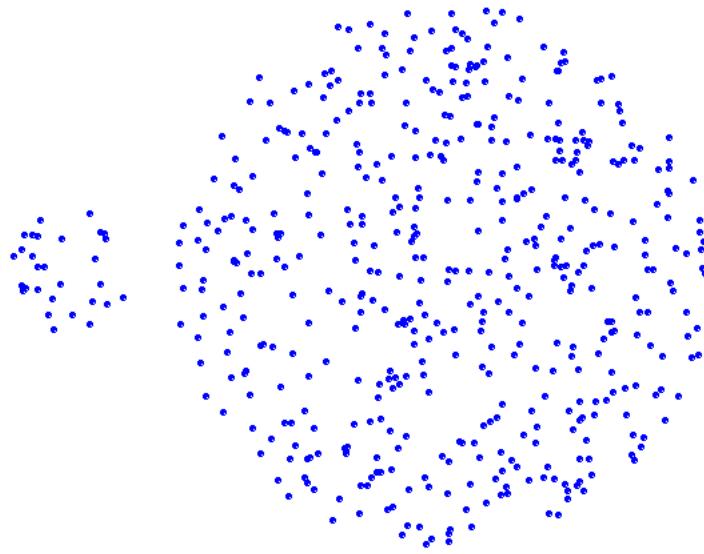


Two Clusters

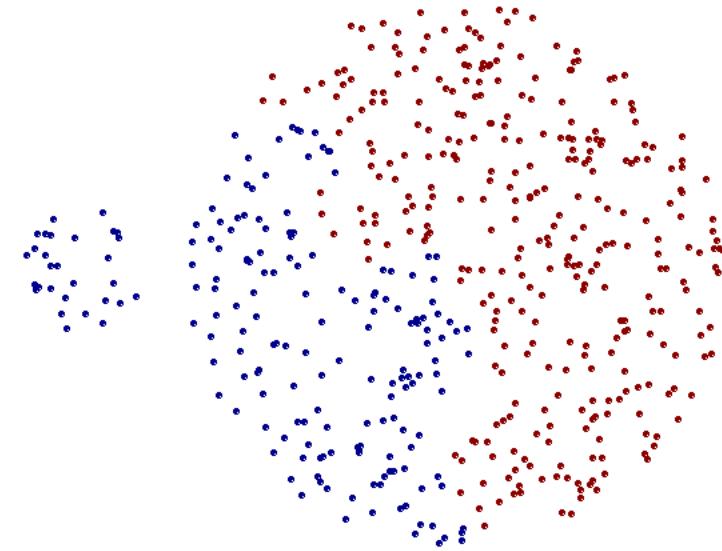
- Less susceptible to noise and outliers



Limitations of MAX



Original Points



Two Clusters

- Tends to break large clusters
- Biased towards globular clusters



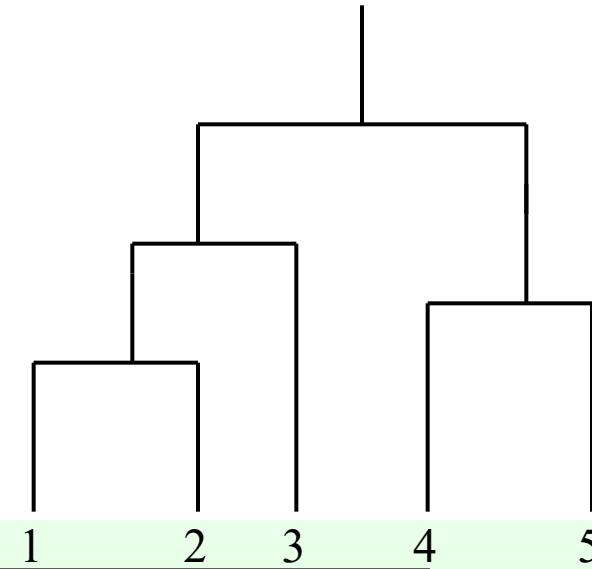
Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

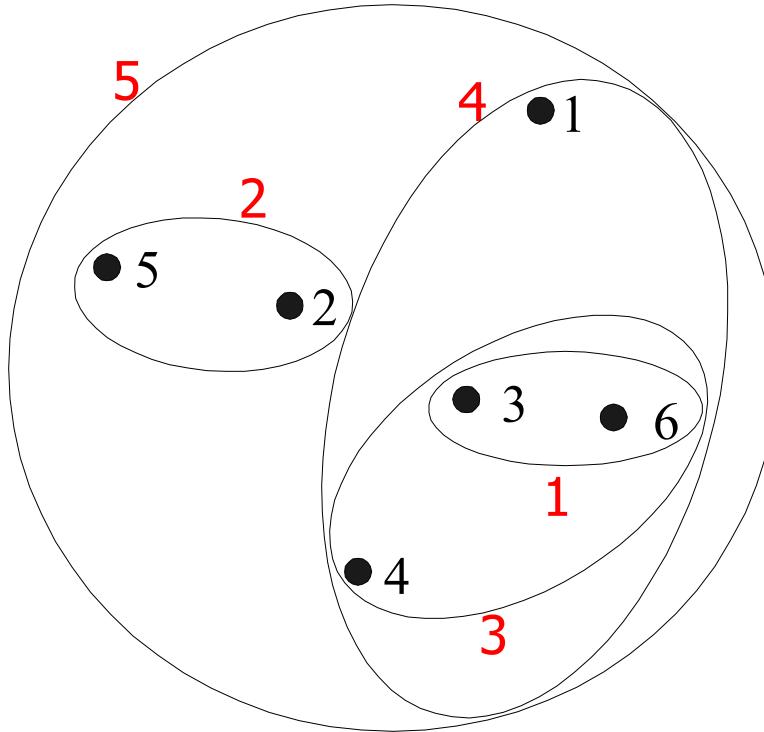
- Need to use average connectivity for scalability since total proximity favors large clusters

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

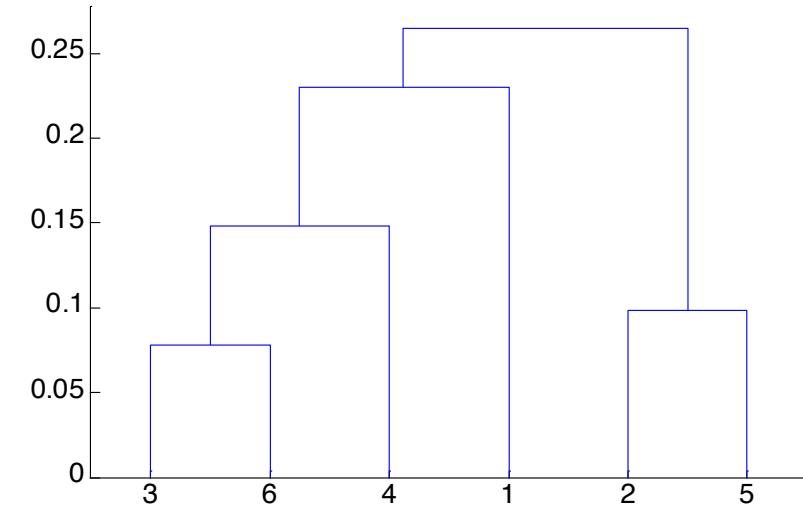




Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram



Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
 - Less susceptible to noise and outliers
- Limitations
 - Biased towards globular clusters

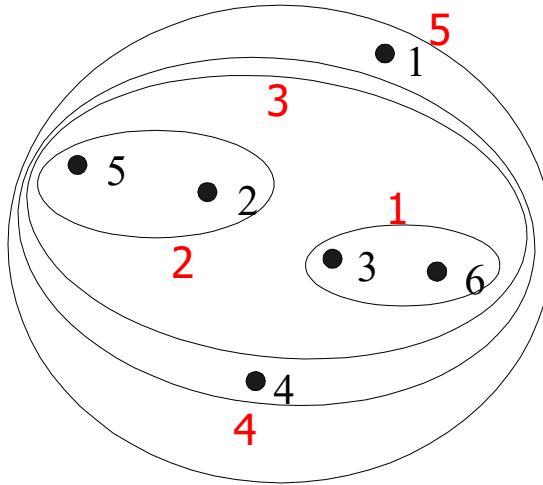


Cluster Similarity: Ward's Method

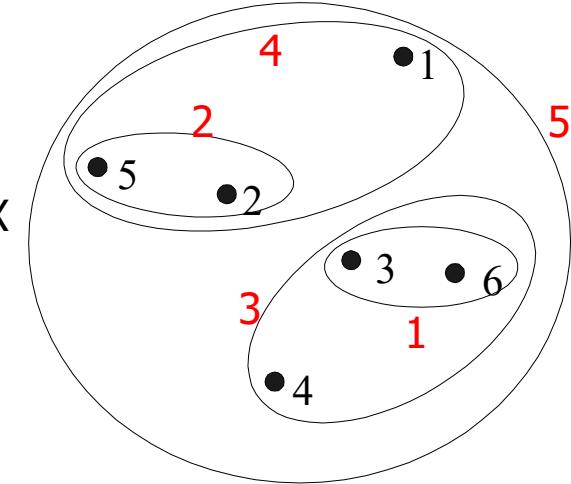
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
 - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
 - Can be used to initialize K-means



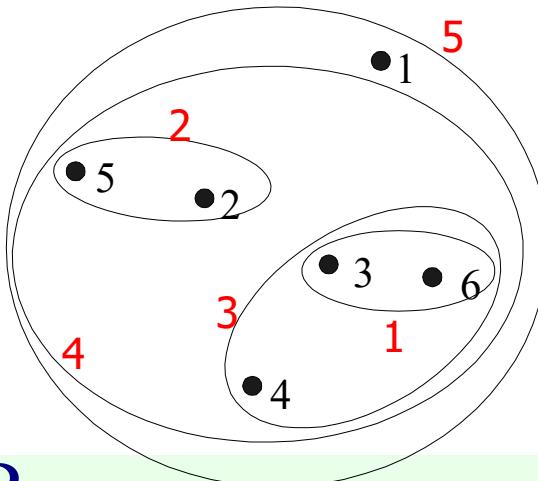
Hierarchical Clustering: Comparison



MIN

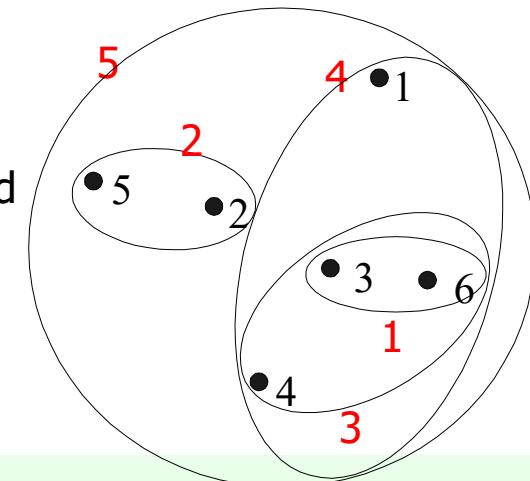


MAX



Group Average

Ward's Method





Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
 - N is the number of points.
- $O(N^3)$ time in many cases
 - There are N steps and at each step the size, N^2 , proximity matrix must be updated and searched
 - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

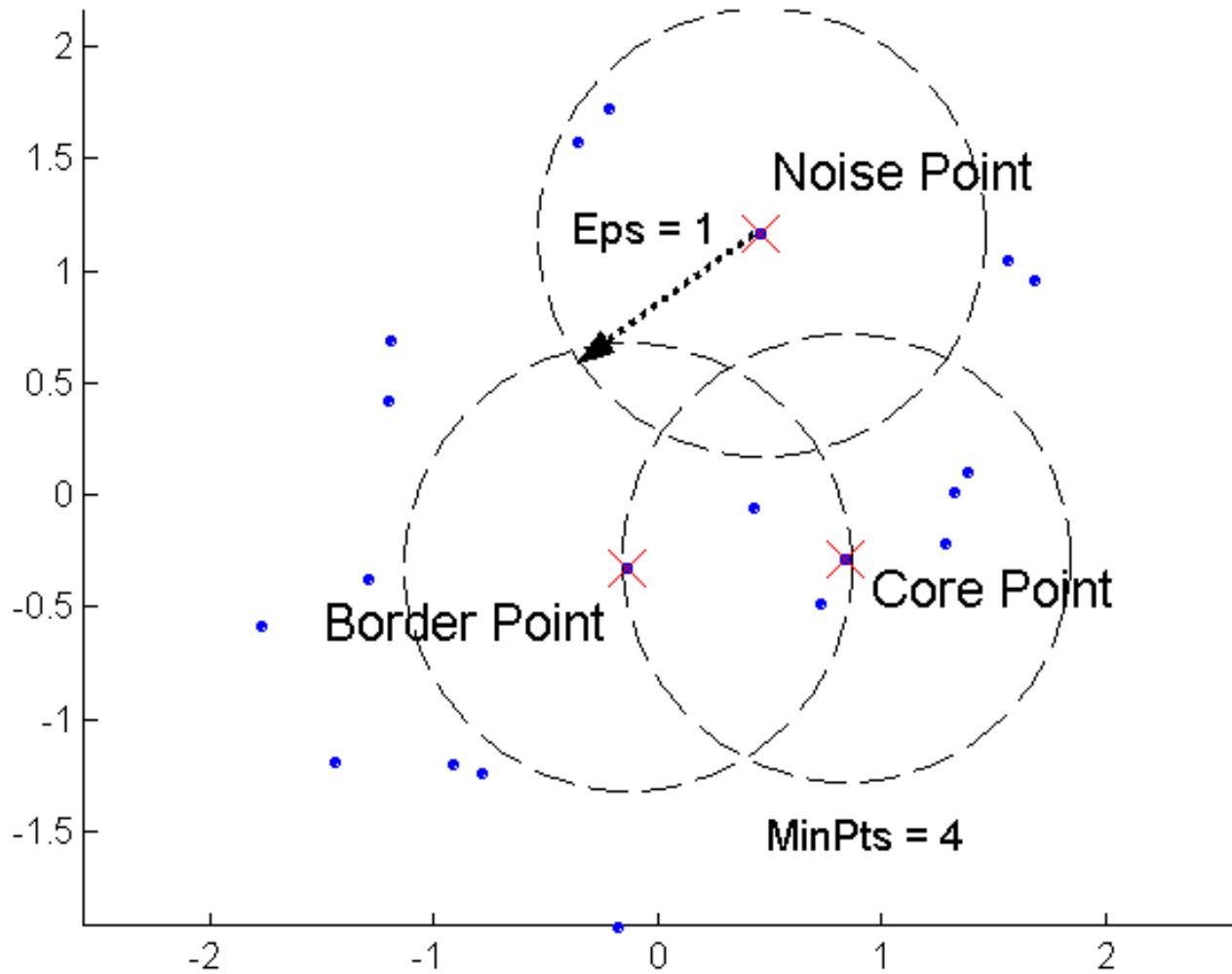


DBSCAN

- DBSCAN is a density-based algorithm
 - Density = number of points within a specified radius (Eps)
 - A point is a **core point** if it has more than a specified number of points (MinPts) within Eps
 - These are points that are at the interior of a cluster
 - A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
 - A **noise point** is any point that is not a core point or a border point.



DBSCAN: Core, Border, and Noise Points





DBSCAN Algorithm

- Eliminate noise points
- Perform clustering on the remaining points

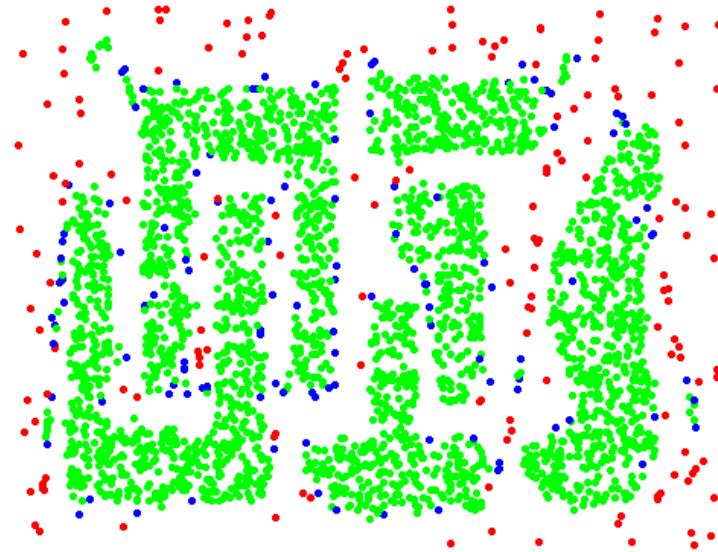
```
current_cluster_label ← 1
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    end if
    for all points in the  $Eps$ -neighborhood, except  $i^{th}$  the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
        end if
    end for
end for
```



DBSCAN: Core, Border, and Noise Points



Original Points



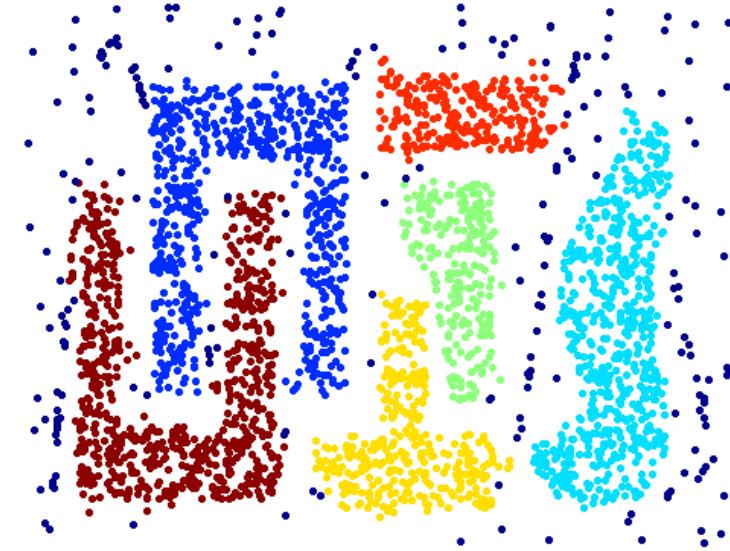
Point types: **core**,
border and **noise**



When DBSCAN Works Well



Original Points

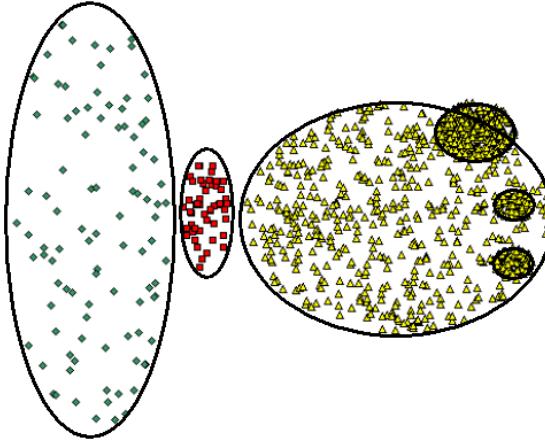


Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

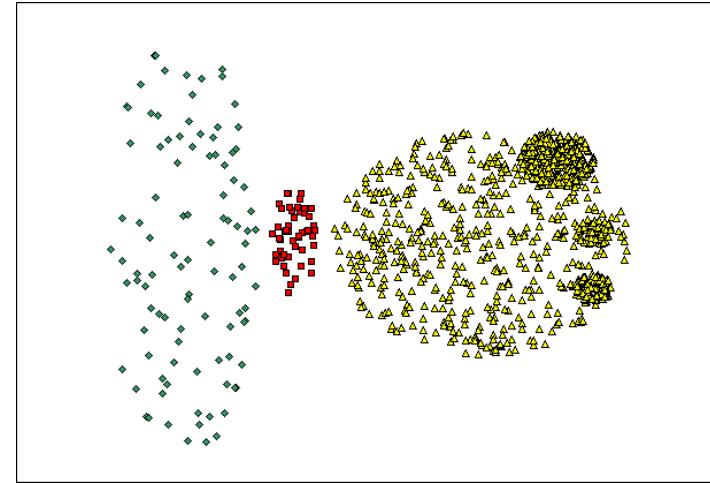


When DBSCAN Does NOT Work Well

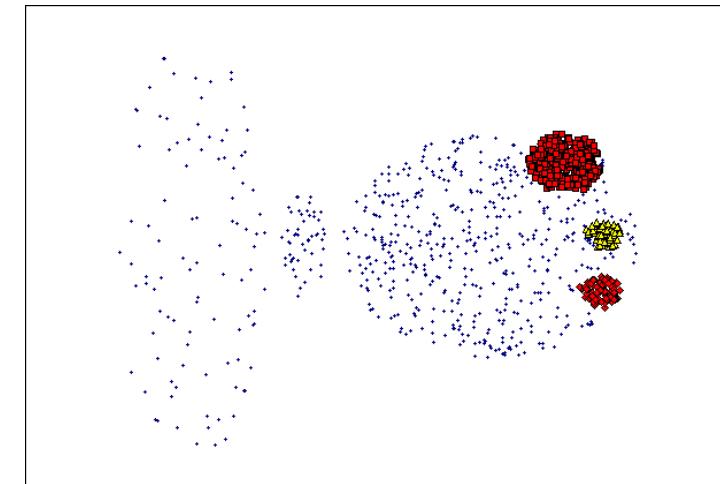


Original Points

- Varying densities
- High-dimensional data



(MinPts=4, Eps=9.75).

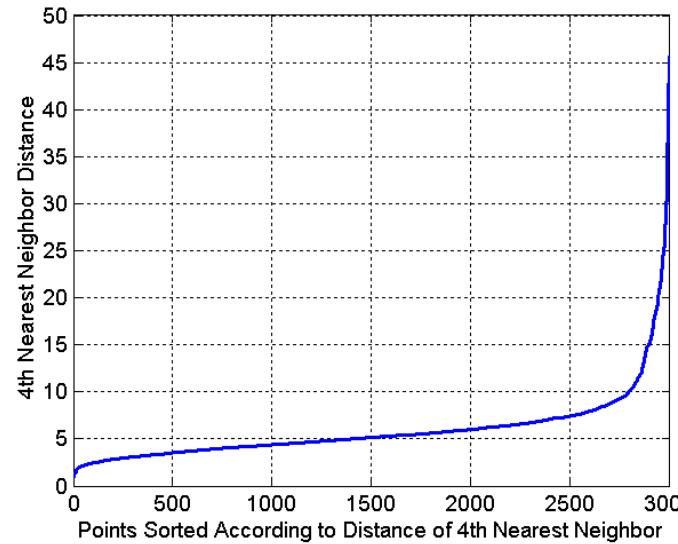


(MinPts=4, Eps=9.62)



DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor



Cluster Validity



Data Base and Data Mining Group of Politecnico di Torino

Elena Baralis, Tania Cerquitelli
Politecnico di Torino



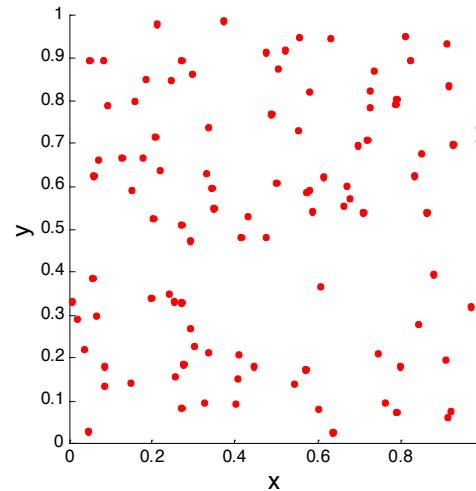
Cluster Validity

- For supervised classification we have a variety of measures to evaluate how good our model is
 - Accuracy, precision, recall
- For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?
- But “clusters are in the eye of the beholder”!
- Then why do we want to evaluate them?
 - To avoid finding patterns in noise
 - To compare clustering algorithms
 - To compare two sets of clusters
 - To compare two clusters

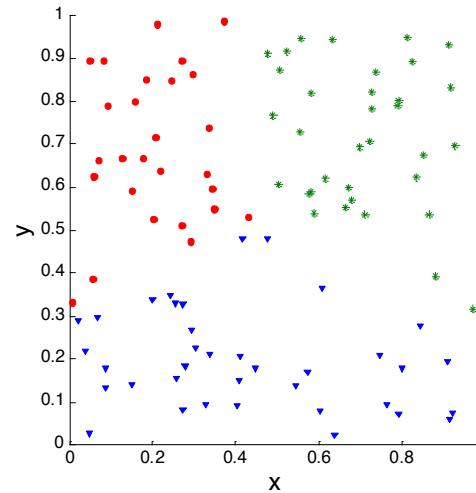


Clusters found in Random Data

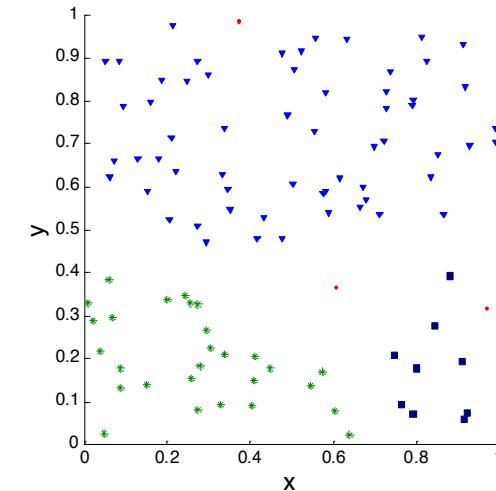
Random Points



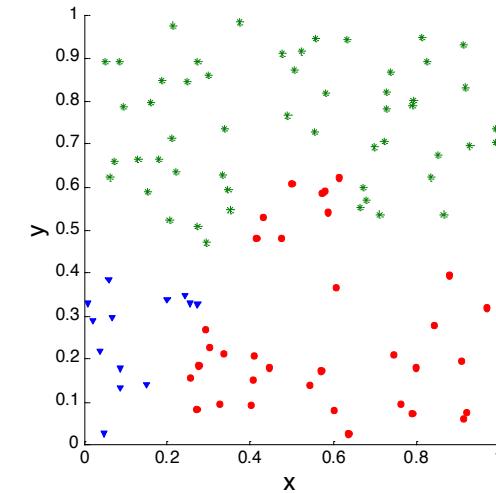
K-means



DBSCAN



Complete Link





Different Aspects of Cluster Validation

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
 - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the 'correct' number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.



Measures of Cluster Validity

- Numerical measures are applied to judge various aspects of cluster validity
- Numerical measures can be classified into three classes
 - **External Index:** Used to measure the extent to which cluster labels match externally supplied class labels.
 - e.g., entropy, purity
 - **Internal Index:** Used to measure the goodness of a clustering structure *without* respect to external information.
 - e.g., Sum of Squared Error (SSE), cluster cohesion, cluster separation, Silhouette index
 - **Relative Index:** Used to compare two different clusterings or clusters.
 - Rand-Index, adjusted rand-index
 - Often an external or internal index is used for this function, e.g., SSE or entropy



Internal Measures: Cohesion and Separation

- **Cluster Cohesion:** Measures how closely related are objects in a cluster
 - Cohesion is measured by the within cluster sum of squares (SSE)
$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$
- **Cluster Separation:** Measure how distinct or well-separated a cluster is from other clusters
 - Separation is measured by the between cluster sum of squares

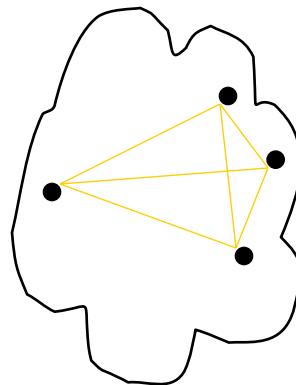
$$BSS = \sum_i |C_i| (m - m_i)^2$$

■ Where $|C_i|$ is the size of cluster i

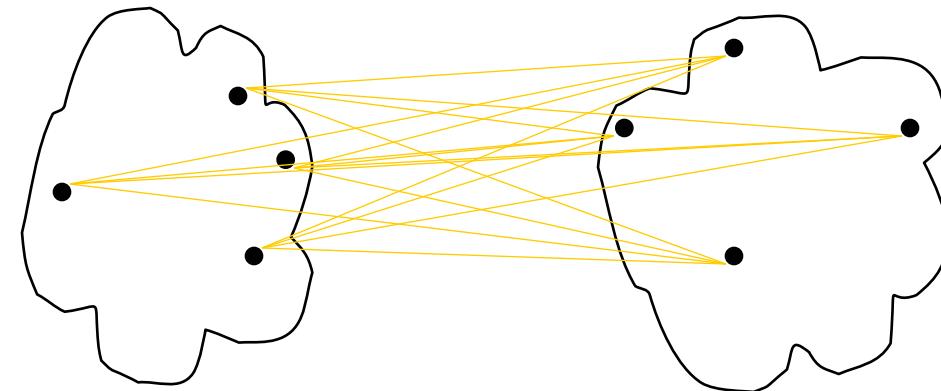


Internal Measures: Cohesion and Separation

- A proximity graph based approach can also be used for cohesion and separation.
 - Cluster cohesion is the sum of the weight of all links within a cluster.
 - Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion



separation



Internal measures: Silhouette

- A succinct measure to evaluate how well each object lies within its cluster
- It is defined for single points
- It considers both cohesion and separation
- Can be computed for
 - Individual points
 - Individual clusters
 - Clustering result



Internal measures: Silhouette

- For each object i
 - $a(i)$: the average dissimilarity of i with all other objects within the same cluster (the smaller the value, the better the assignment)
 - $b(i)$: $\min(\text{average dissimilarity of } i \text{ to any other cluster, of which } i \text{ is not a member})$
- Ranges between -1 and +1
 - Typically between 0 and 1
 - The closer to 1, the better
- Silhouette for clusters and clusterings
 - The average $s(i)$ over all data of a *cluster* measures how tightly grouped all the data in the cluster are
 - The average $s(i)$ over all data of the *dataset* measures how appropriately the data has been clustered



External Measures of Cluster Validity: Entropy and Purity

Table 5.9. K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij} , the ‘probability’ that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j . Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^L p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^K \frac{m_i}{m} e_j$, where m_j is the size of cluster j , K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j , is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^K \frac{m_i}{m} purity_j$.



Rand Index

- Idea
 - Any two objects that are in the same cluster should be in the same class and vice versa
- Given
 - f_{00} = number of pairs of objects having a different class and a different cluster
 - f_{01} = number of pairs of objects having a different class and the same cluster
 - f_{10} = number of pairs of objects having the same class and a different cluster
 - f_{11} = number of pairs of objects having the same class and the same cluster
- Rand Index

$$Rand\ Index = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$



Final Comment on Cluster Validity

“The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.”

Algorithms for Clustering Data, Jain and Dubes

Data Science & Machine Learning Lab

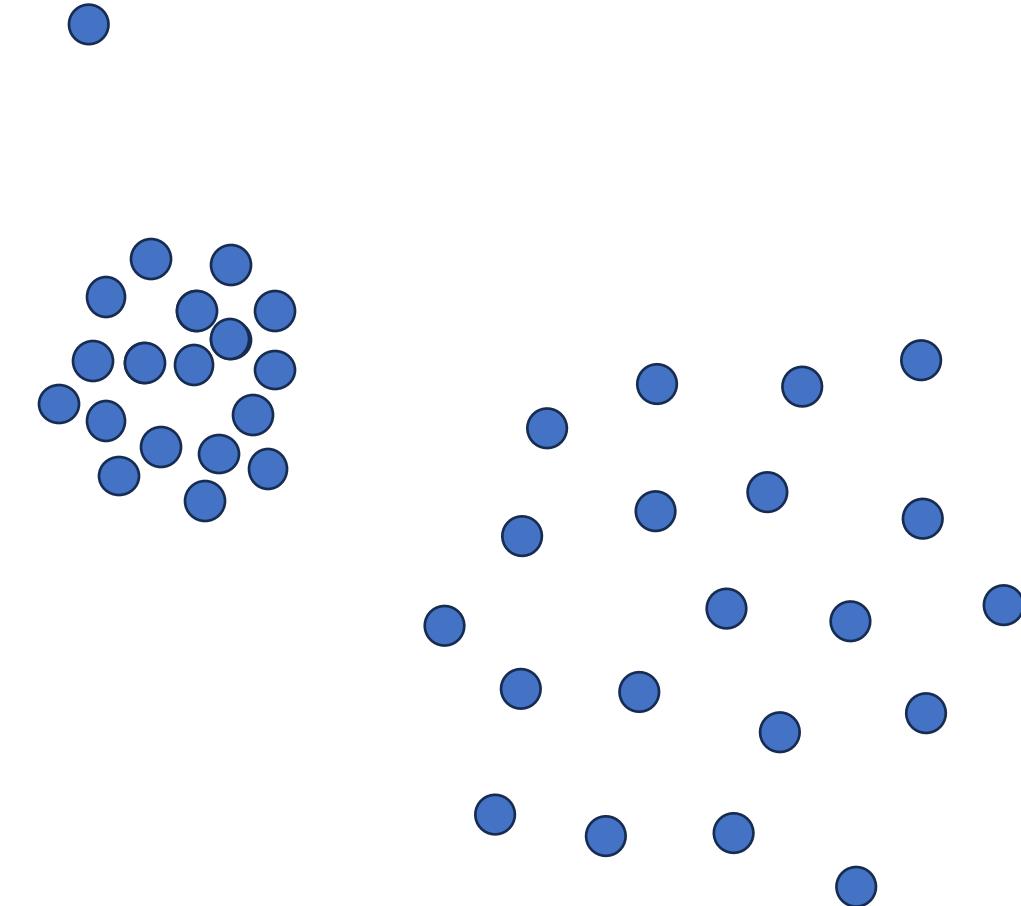
Anomaly Detection

Flavio Giobergia



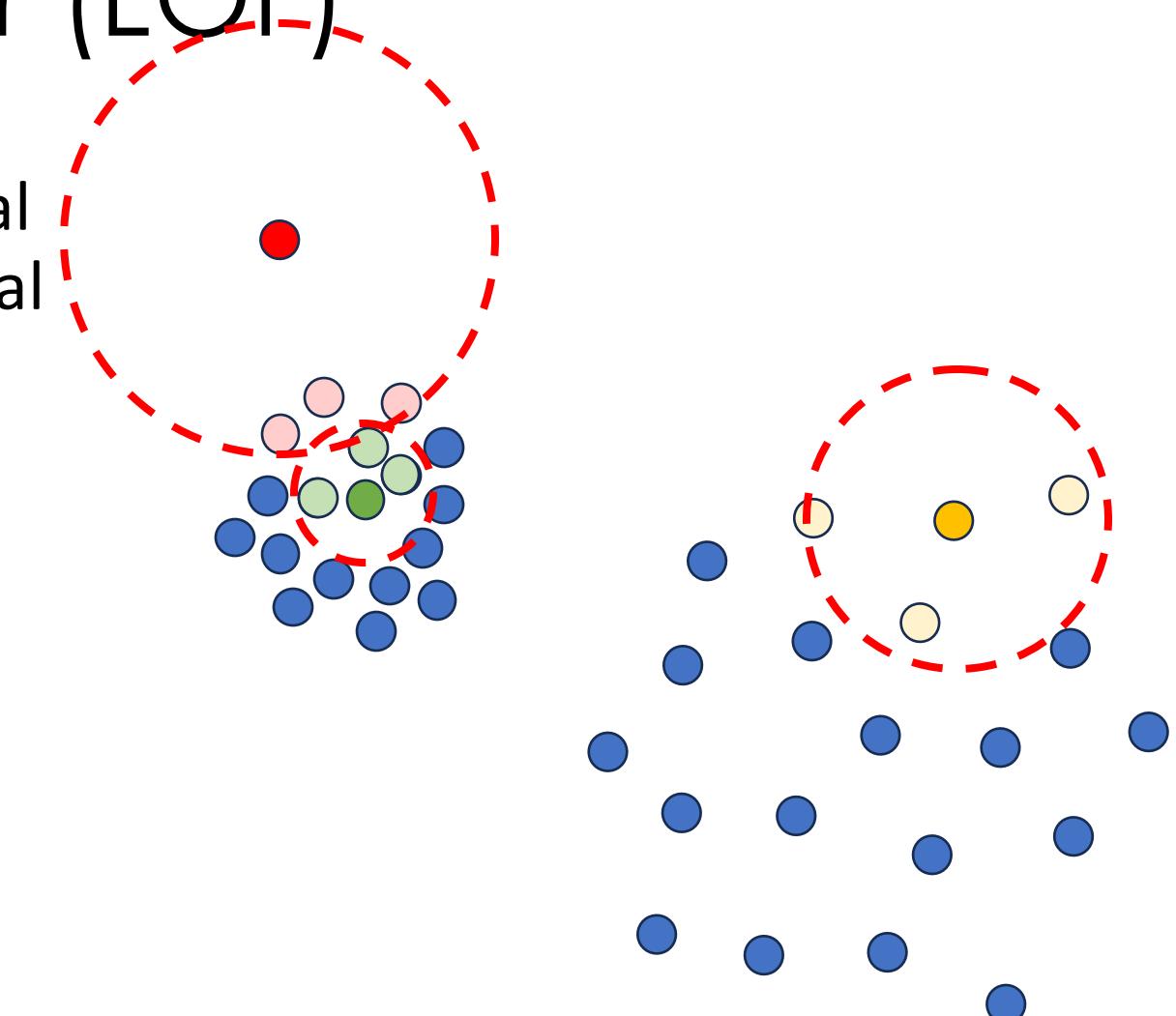
Local Outlier Factor (LOF)

- LOF considers points to be anomalous if they are found in non-dense regions of space.
- Since different regions of space can have different densities, LOF estimates the "local" density of each region of space
- If a point has a smaller density than its (k) nearest neighbors, it is considered an anomaly.



Local Outlier Factor (LOF)

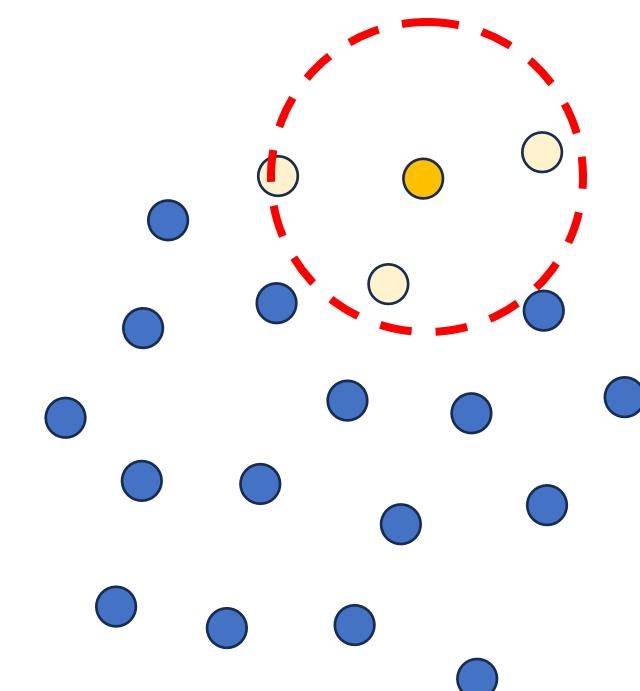
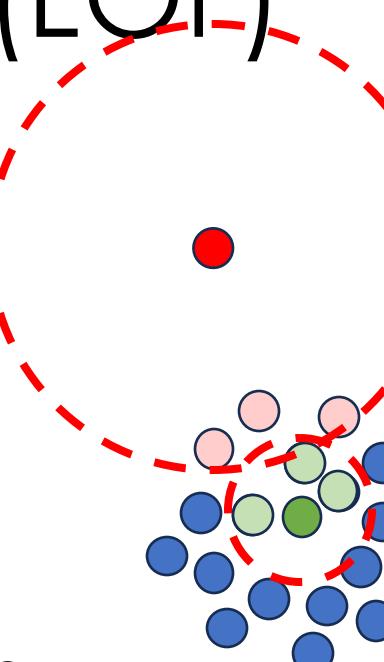
- For a given point, we compute its “local density”, which is inversely proportional to the distance to its k-th nearest neighbor*
- Large distance: neighbors are far away
→ low-density region
- Small distance: neighbors are close by
→ high-density region



* Note: this is a slight simplification w.r.t. actual LOF – where the “Local Reachability Distance” (LRD) is computed as an estimate of the local density

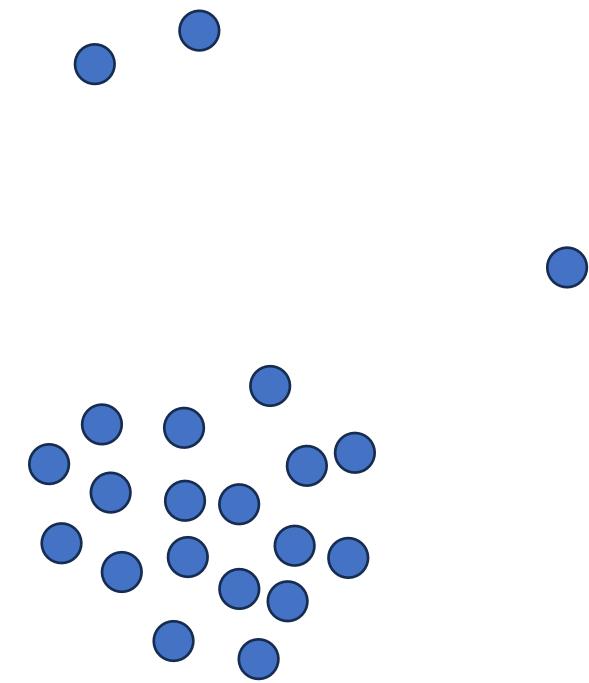
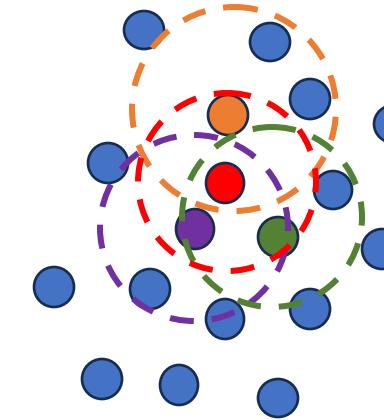
Local Outlier Factor (LOF)

- If all high-density regions of space had the same density, we could simply define a threshold distance τ
- All points having k-th NN more distant than τ are anomalies
- However, the space may have regions of space of varying distance
- So, we need to “estimate” a τ separately for each point



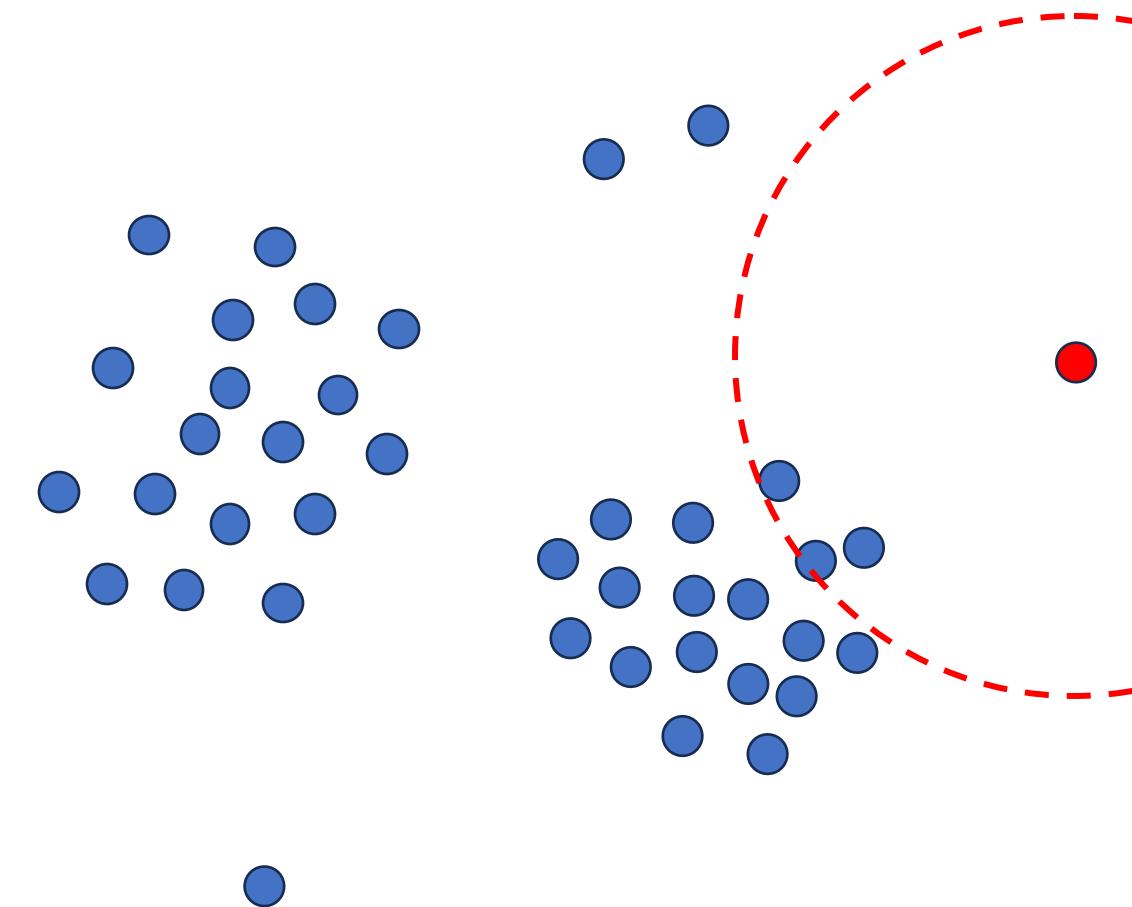
Local Outlier Factor (LOF)

- We estimate the “local density” of a sample ● with the average distance of the k-th NN of the neighbors of ●

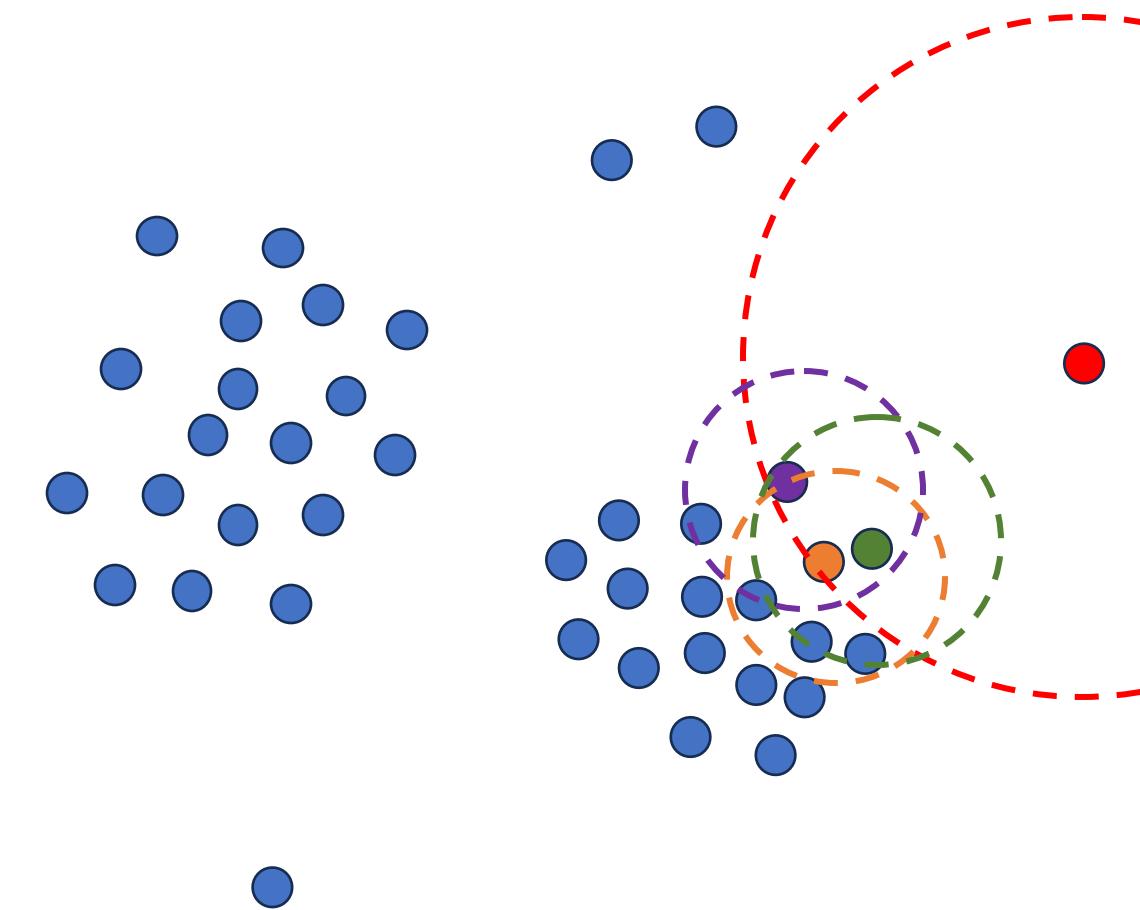
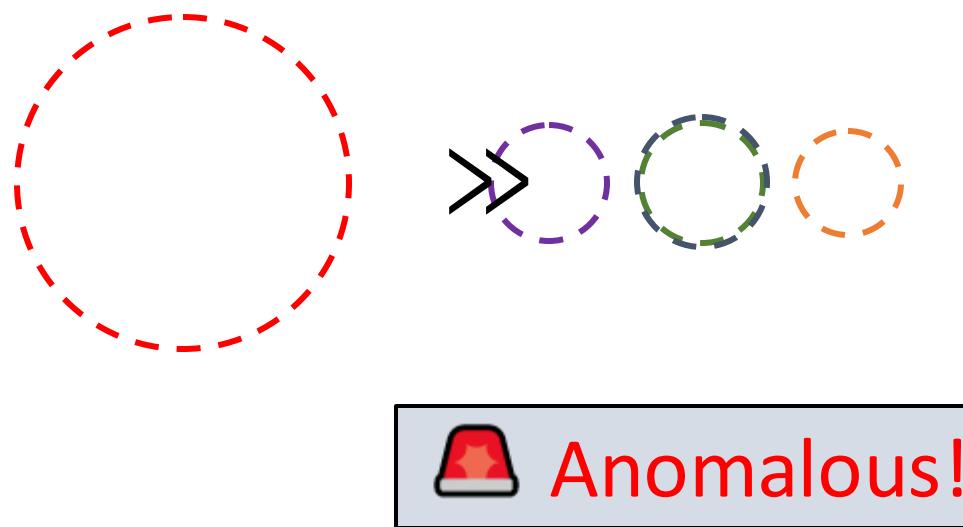


Local Outlier Factor (LOF)

- For an anomalous point, the k-th NN will be quite far
- But, the nearest neighbors themselves (if not anomalous) will have much closer neighbors!

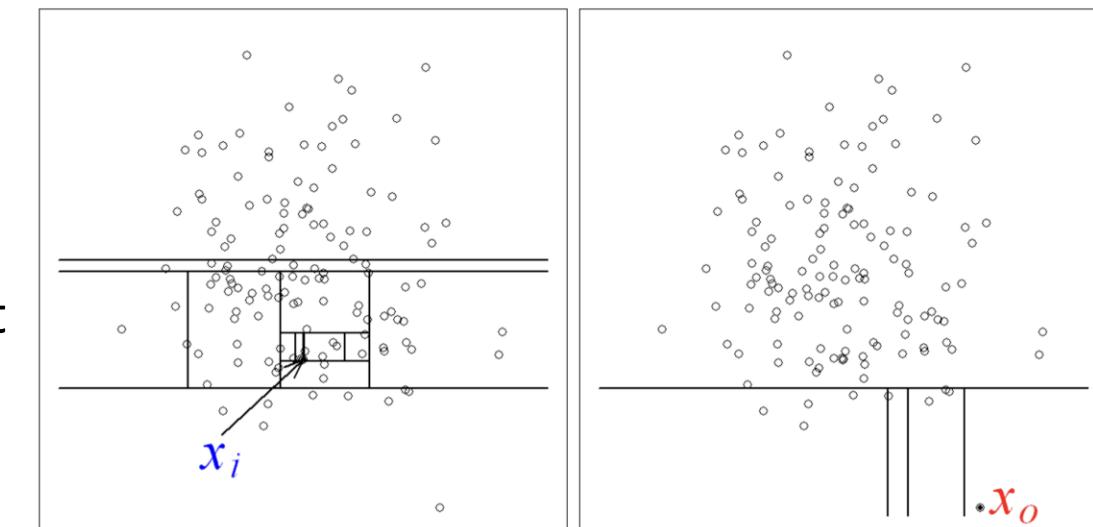


Local Outlier Factor (LOF)



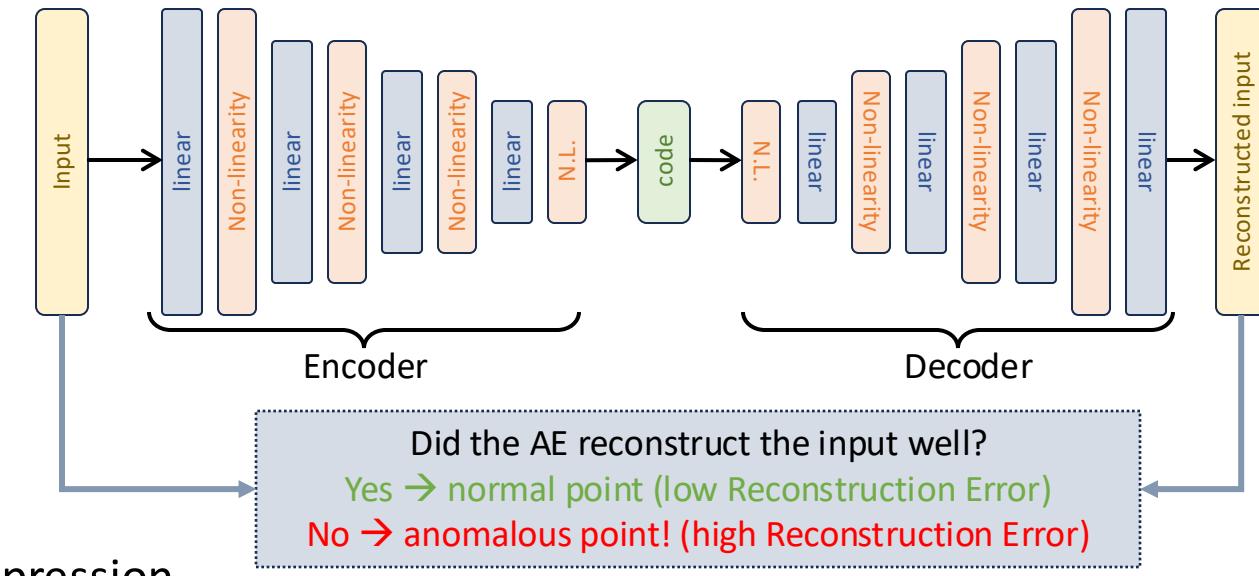
Isolation Forest

- An Isolation Forest is a collection of *randomly generated trees*
 - Randomly generated = for each node, randomly select a feature and split value, and split on “feature < value”
 - Each tree will split the input space into random partitions
- **Anomalies**: points isolated in fewer splits (short root-leaf path)
- **Normal points**: more difficult to be isolated (long root-leaf path)
- Repeat for multiple (random) trees, to get multiple paths
- *Anomaly score*(x) = $2^{\frac{-\mathbb{E}(h(x))}{c(n)}}$
 - $\mathbb{E}(h(x)) \rightarrow$ average path length to isolate x
 - $c(n) \rightarrow$ normalization factor based on dataset size, since more points = deeper trees on average = longer paths to isolate each point



Autoencoders (AE)

- Autoencoders are Neural Networks with an encoder and a decoder.
 - The encoder compresses the **N-dimensional input** into a **D-dimensional vector ("code")**
 - Note that, if $D < N$ (as is often the case), the compression will be lossy, if the data is full rank
 - The decoder takes the **D-dimensional "compressed"** vector and tries to reconstruct the **original input**
- During training, the AE “learns” to compress typical data and to reconstruct it properly.
- After training, if a data point is reconstructed well, it means that the AE knew how to compress/decompress the point (so, it was an “expected” point)
- Anomalous points are those that the AE cannot reconstruct well (because the AE hasn’t seen “many” such points)



Anomaly Detection

Data Science and Machine Learning Lab

Flavio Giobergia

0.1 Anomaly Detection (overview)

Anomaly Detection (AD) is the process of identifying data points, events, or patterns that deviate significantly from expected behavior.

In **unsupervised AD**, the goal is to detect anomalies without labeled data – the model learns the normal patterns and flags deviations (==> We will mainly focus on this!)

In **supervised AD**, instances of anomalies are available, models learn to separate normal patterns from anomalies.

AD algorithms typically produce, for each sample, an **anomaly score** indicating how likely it is to be an anomaly. A threshold is then applied to classify samples as normal or anomalous.

0.2 Types of Anomalies

0.2.1 Point Anomalies

Individual data points that significantly differ from the rest of the population.

Example: a person withdrawing \$10,000 in cash, at 3 AM from an ATM.

```
[1]: import numpy as np
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

[2]: # add arrow to an anomalous point
def show_anomaly(ax, point):
    # white background, black border for the text box
    ax.annotate('Anomaly', xy=point, xytext=(0.5, 0.5), arrowprops=dict(facecolor='red', shrink=0.05), fontsize=12, color='red', bbox=dict(boxstyle="round,pad=0.3", edgecolor='black', facecolor='white'))

[3]: # generate dataset (sampled from normal distribution, with 1 anomaly)
X_normal = np.random.normal(loc=0.0, scale=0.25, size=(300, 2))
X_anomaly = np.array([[2,2]])
X = np.vstack([X_normal, X_anomaly])

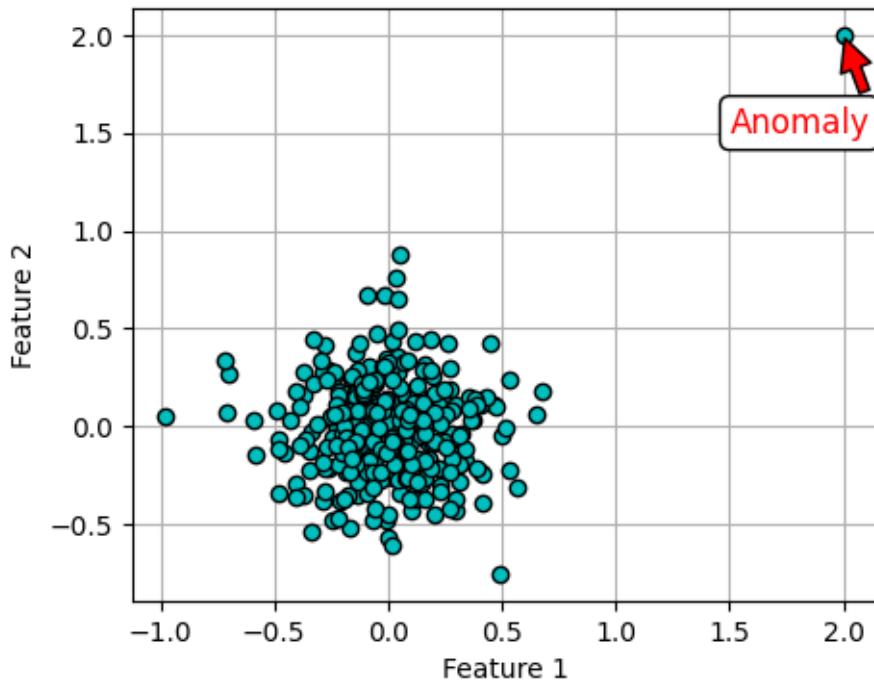
# plot dataset
fig, ax = plt.subplots(figsize=(5, 4))
```

```

ax.scatter(X[:, 0], X[:, 1], c='c', edgecolors='k')
show_anomaly(ax, X_anomaly[0])

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.grid()
ax.set_axisbelow(True)

```



0.2.2 Contextual Anomalies

For contextual anomalies, a data point that is anomalous only within a certain context i.e., it would be “normal” in other circumstances.

Example: a person doing a wire transfer for \$500,000. If it was a company, the operation would be less anomalous!

```

[4]: X, y = make_blobs(n_samples=100, centers=3, cluster_std=0.5, random_state=1)
X *= .1
anomaly_id = (y == 0).nonzero()[0][-1]
y[anomaly_id] = 1

markers = ["P", "o", "^"]
group_labels = ["Cats", "Dogs", "Parrots"]

```

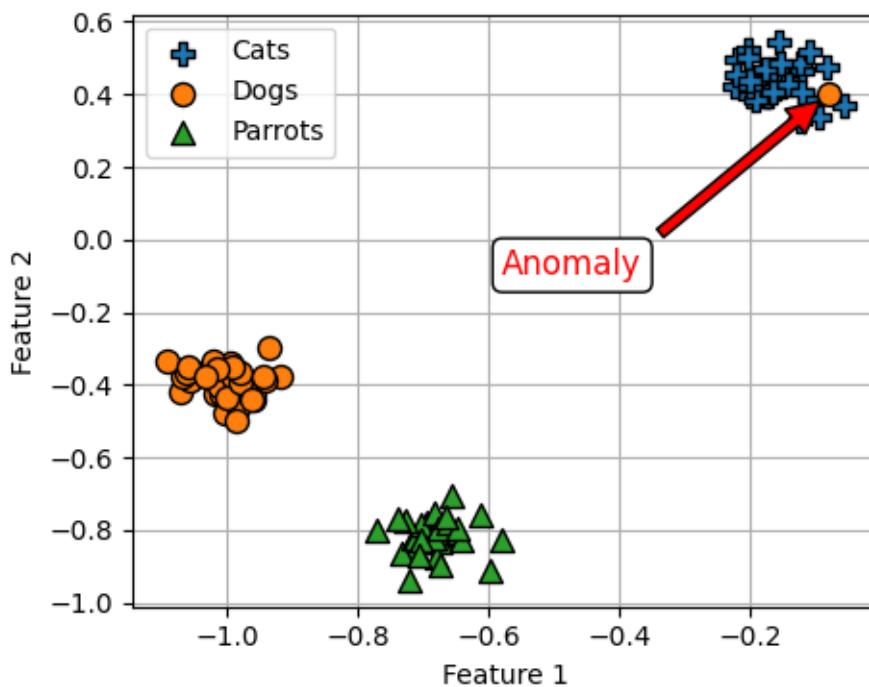
```

fig, ax = plt.subplots(figsize=(5, 4))
for c_id in np.unique(y):
    ax.scatter(X[y==c_id, 0], X[y==c_id, 1], label=group_labels[c_id], c=group_labels[c_id], edgecolors='k', marker=markers[c_id], s=75)

show_anomaly(ax, X[anomaly_id])

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.legend()
ax.grid()
ax.set_axisbelow(True)

```



0.2.3 Collective Anomalies

A group of otherwise normal points that collectively form an anomaly.

Example: multiple people depositing cash for \$9,000 and then wire-transferring the same amount to a foreign account. A single deposit of \$9,000 is not anomalous, but the collective behavior is suspicious.

```
[5]: from sklearn.datasets import make_moons
```

```

X, y = make_moons(n_samples=900, noise=0.1, random_state=1)
X_noise = np.random.normal(loc=(1.5, 0.75), scale=0.075, size=(20, 2))

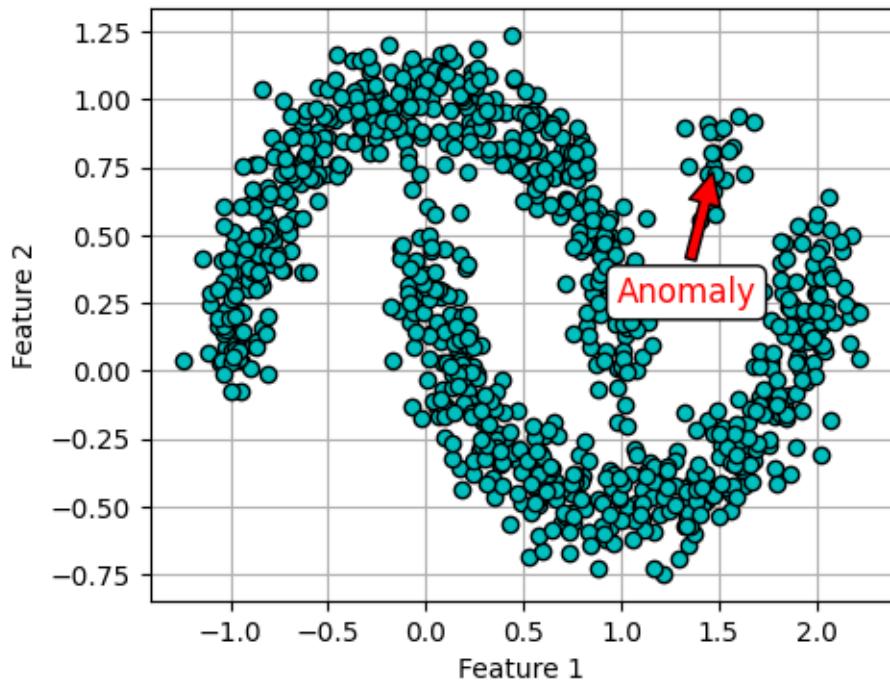
X = np.vstack([X, X_noise])

fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X[:, 0], X[:, 1], c='c', edgecolors='k')

show_anomaly(ax, X_noise[0])

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.grid()
ax.set_axisbelow(True)

```



0.3 Anomaly Detection techniques

- **Statistical methods:**
 - Based on statistical models to identify data points that deviate significantly from the expected distribution
 - Example: Z-score method
- **Clustering-based methods:**
 - Use clustering algorithms to group similar data points together and identify those that do not belong to any cluster as anomalies

- Example: DBSCAN, K-means
 - **Proximity-based methods:**
 - Use distance or density measures to identify anomalies based on their proximity to other data points
 - Example: k-NN, Local Outlier Factor (LOF)
 - **Tree-based methods:**
 - Use properties of trees to isolate anomalies in the data
 - Example: Isolation Forest
 - **Boundary-based methods:**
 - Use decision boundaries to separate normal data points from anomalies
 - Example: One-Class SVM
 - **Neural Network-based methods:**
 - Use neural networks to learn data distributions and identify anomalies (e.g., based on reconstruction errors)
 - Example: Autoencoders
-

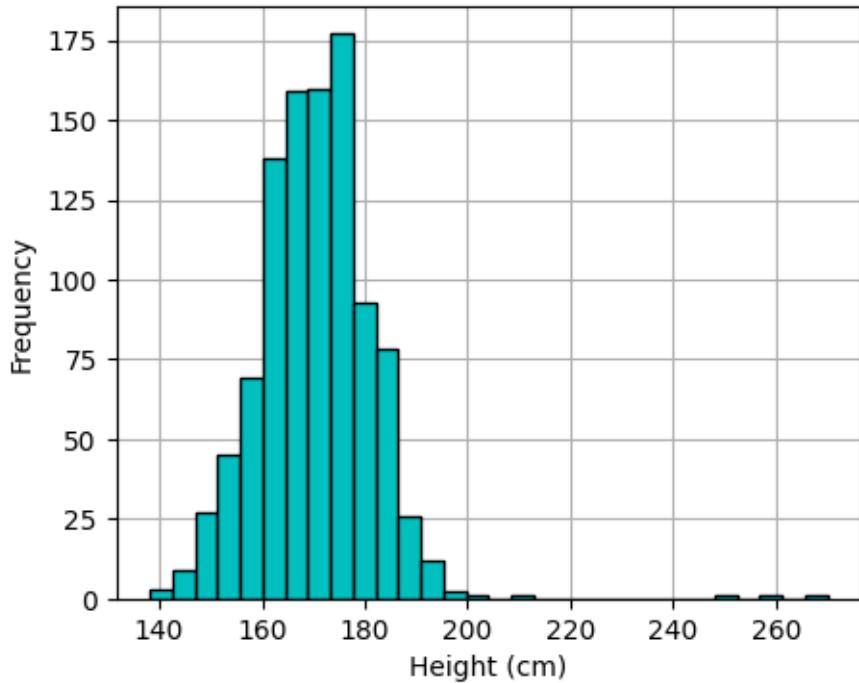
0.3.1 Statistical methods

Quantify the deviation of a data point from the expected distribution using statistical measures. For instance, the Z-score method calculates how many standard deviations a data point is from the mean. Points with high absolute Z-scores are considered anomalies.

Multivariate generalizations can also be used (e.g., Mahalanobis distance).

```
[6]: heights = np.random.normal(loc=170, scale=10, size=1000)
heights_anomalies = np.array([250, 260, 270])
heights = np.hstack([heights, heights_anomalies])
```

```
[7]: fig, ax = plt.subplots(figsize=(5, 4))
ax.hist(heights, bins=30, color='c', edgecolor='k')
ax.set_xlabel('Height (cm)')
ax.set_ylabel('Frequency')
ax.grid()
ax.set_axisbelow(True)
```



```
[8]: z_normalized = (heights - np.mean(heights)) / np.std(heights)

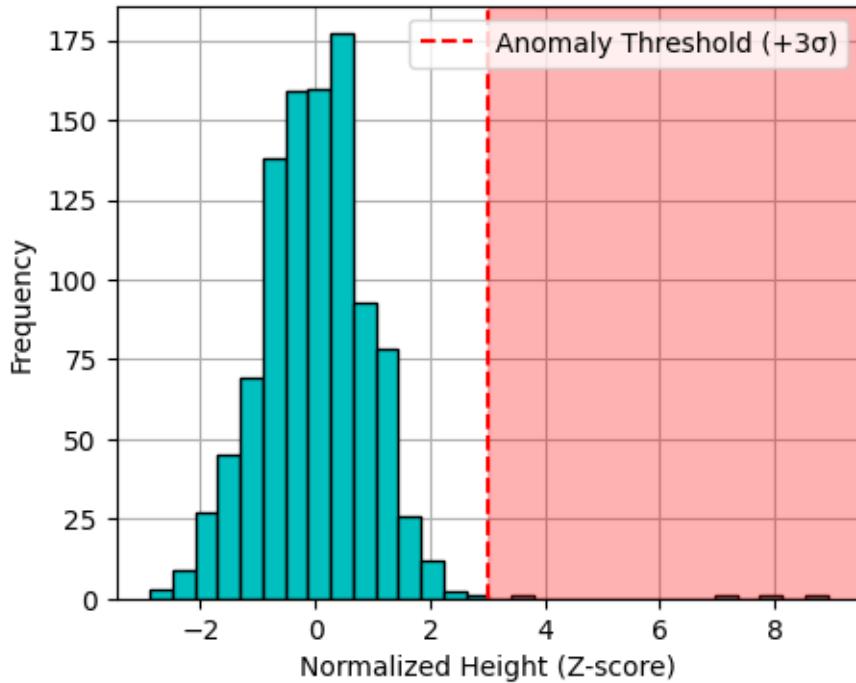
anomaly_threshold = 3

print(f"Anomalies (3σ): {heights[abs(z_normalized) > anomaly_threshold]}")
```

Anomalies (3σ): [208.59256742 250. 260. 270.]

```
[9]: fig, ax = plt.subplots(figsize=(5, 4))
ax.hist(z_normalized, bins=30, color='c', edgecolor='k')
ax.set_xlabel('Normalized Height (Z-score)')
ax.set_ylabel('Frequency')
ax.grid()
ax.set_axisbelow(True)

ax.axvline(x=anomaly_threshold, color='r', linestyle='--', label=f'Anomaly Threshold (+{anomaly_threshold}σ)')
ax.axvspan(anomaly_threshold, xlim[1], color='red', alpha=0.3)
ax.legend()
ax.set_xlim(xlim);
```



0.3.2 Clustering-based methods

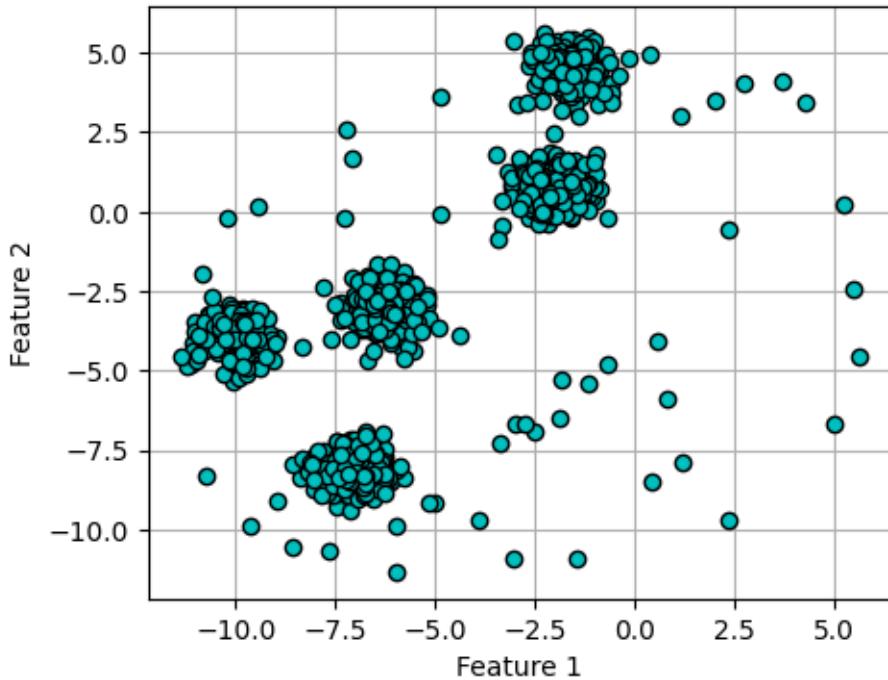
Clustering algorithms group similar data points together. Data points that do not belong to any cluster or are in small clusters are considered anomalies.

K-means With K-means, for example, points that are far from their assigned cluster centroids can be flagged as anomalies.

```
[10]: X_blob, y_blob = make_blobs(n_samples=1_000, centers=5, cluster_std=0.5, random_state=1)
X_anomalies = np.random.uniform(X_blob.min(), X_blob.max(), size=(50, 2))

X_blob = np.vstack([X_blob, X_anomalies])

fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X_blob[:, 0], X_blob[:, 1], c='c', edgecolors='k')
ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.grid()
ax.set_axisbelow(True)
```



```
[11]: from sklearn.cluster import KMeans

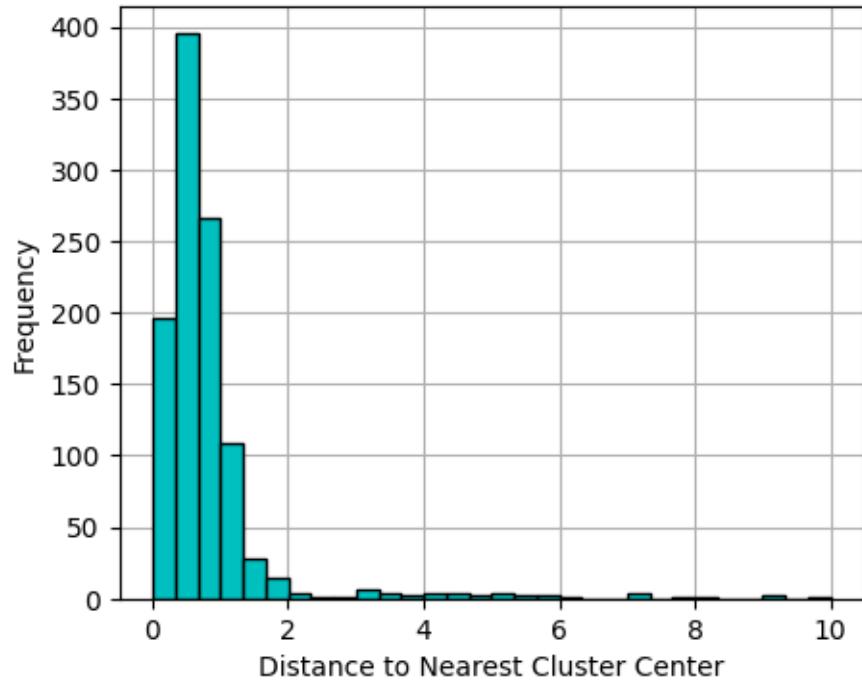
model = KMeans(n_clusters=5)
model.fit(X_blob)

# transform() computes the distance between each point and each cluster center
# (equivalent to euclidean_distances(X, model.cluster_centers_))
# Each point is assigned to the closest cluster. With min(axis=1) we can extract
# that distance
distances = model.transform(X_blob).min(axis=1)

fraction_of_anomalies = 0.05 # we expect ~ 5% of the data to be anomalies

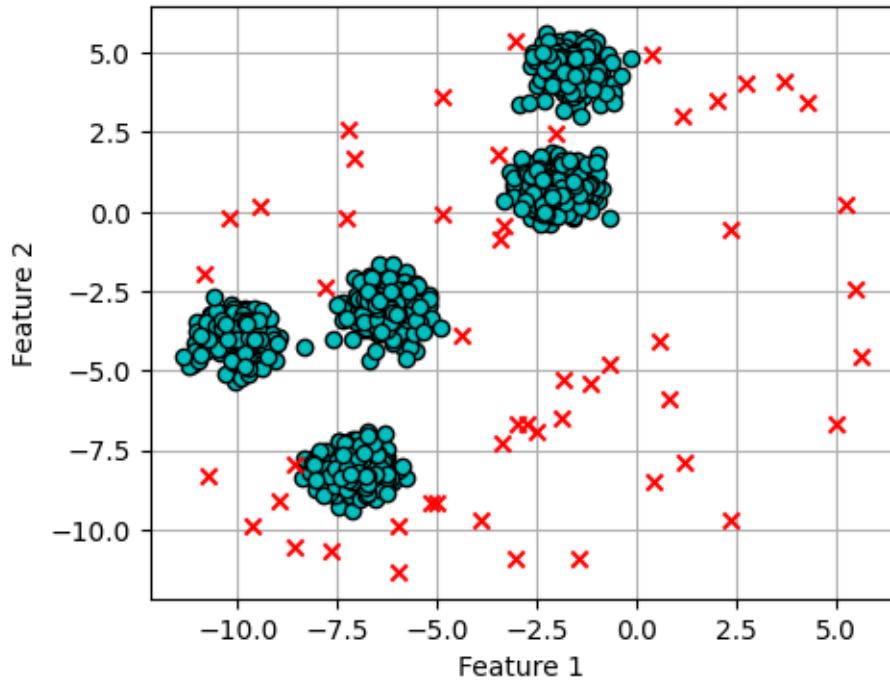
anomalous_indices = distances.argsort()[-int(fraction_of_anomalies * len(X_blob)):]
y_anomaly = np.zeros(len(X_blob), dtype=int)
y_anomaly[anomalous_indices] = 1
```

```
[12]: fig, ax = plt.subplots(figsize=(5, 4))
ax.hist(distances, bins=30, color='c', edgecolor='k')
ax.set_xlabel('Distance to Nearest Cluster Center')
ax.set_ylabel('Frequency')
ax.grid()
ax.set_axisbelow(True)
```



```
[13]: fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X_blob[y_anomaly==0, 0], X_blob[y_anomaly==0, 1], c='c', ▾
           edgecolors='k')
ax.scatter(X_blob[y_anomaly==1, 0], X_blob[y_anomaly==1, 1], c='r', marker='x')

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.grid()
ax.set_axisbelow(True)
```



DBSCAN Some clustering algorithms (e.g., DBSCAN) can directly identify outliers as points that do not belong to any cluster.

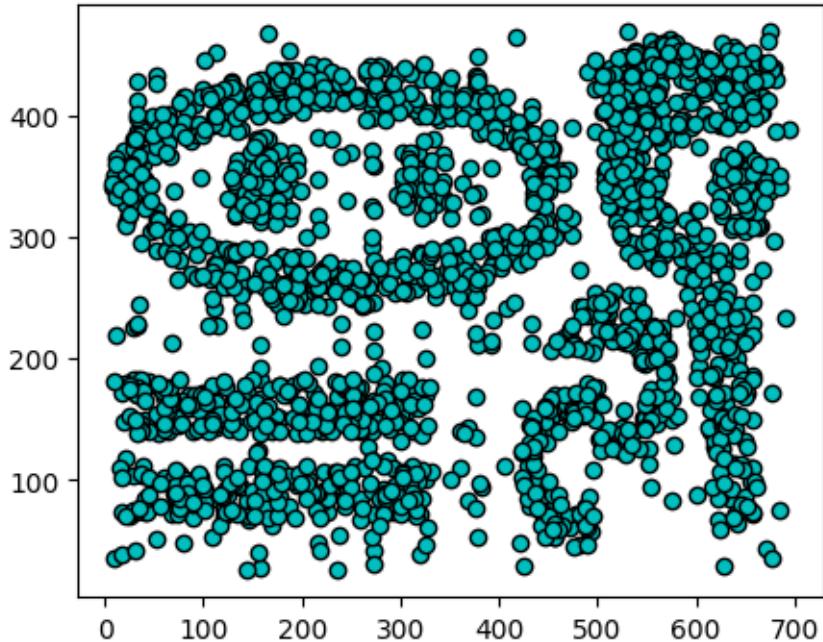
In scikit-learn, these points are labeled with cluster -1.

```
[14]: from sklearn.cluster import DBSCAN
import pandas as pd

df = pd.read_csv("chameleon.data", sep=" ", header=None, names=['x1', 'x2'])
X = df.values

fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X[:, 0], X[:, 1], c='c', edgecolors='k')
```

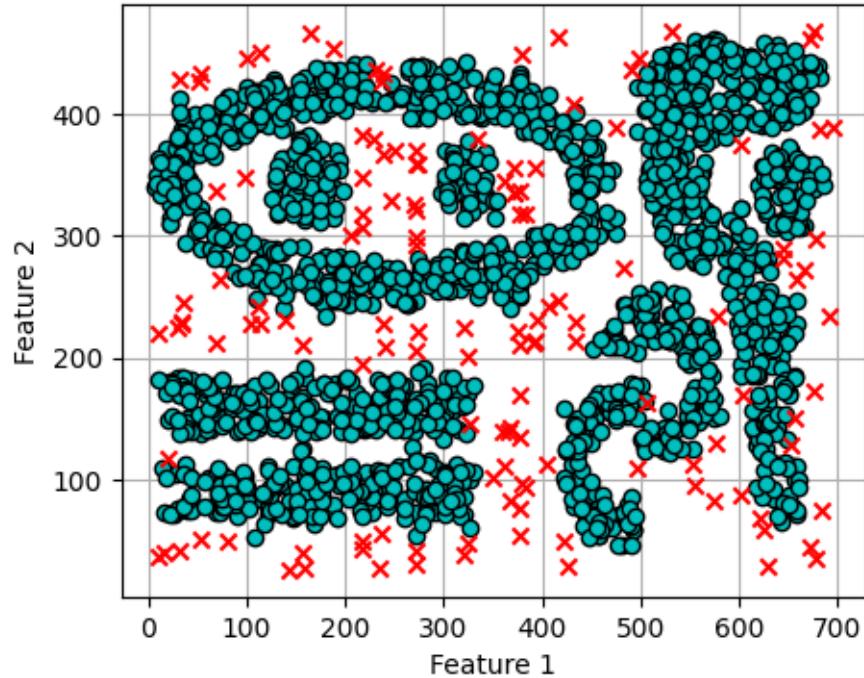
[14]: <matplotlib.collections.PathCollection at 0x17ab84ad0>



```
[15]: model = DBSCAN(eps=15, min_samples=5)
y_pred = model.fit_predict(X)

fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X[y_pred>-1, 0], X[y_pred>-1, 1], c='c', edgecolors='k')
ax.scatter(X[y_pred== -1, 0], X[y_pred== -1, 1], c='r', marker='x')

ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.grid()
ax.set_axisbelow(True)
```



(You can try to apply K-means to this dataset. Do you expect the algorithm to work well? How can you make it work to obtain the anomalies?)

0.3.3 Proximity-based methods

These methods rely on the distance or density of data points. Points that are far from others or in low-density regions are considered anomalies.

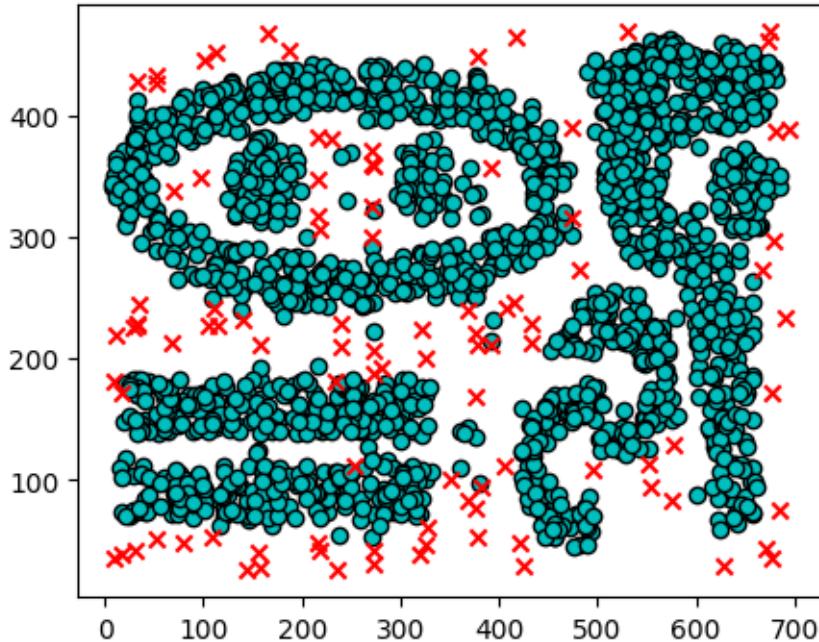
Local Outlier Factor (LOF)

```
[16]: from sklearn.neighbors import LocalOutlierFactor

lof = LocalOutlierFactor(n_neighbors=10, contamination=0.05) # contamination:
    ↪expected fraction of outliers in the data (will be used to define the
    ↪threshold)
y_anomaly = lof.fit_predict(X)

fig, ax = plt.subplots(figsize=(5, 4))
ax.scatter(X[y_anomaly==1, 0], X[y_anomaly==1, 1], c='c', edgecolors='k')
ax.scatter(X[y_anomaly==-1, 0], X[y_anomaly==-1, 1], c='r', marker='x')
```

[16]: <matplotlib.collections.PathCollection at 0x17aded710>



The results are similar to those obtained with DBSCAN, since the “high density” regions all have approximately the same density. What happens if some region of space becomes less dense?

```
[17]: # find the biggest cluster (according to DBSCAN), and downsample it (with [:3])
      ↪we are selecting approx. 1/3 of the dataset)

model = DBSCAN(eps=15, min_samples=5)
y_pred = model.fit_predict(X)

clusters, frequencies = np.unique(y_pred, return_counts=True)

biggest_cluster = clusters[frequencies.argmax()]
X_new = np.vstack([X[y_pred != biggest_cluster], X[y_pred == biggest_cluster] [::
      ↪3]])

# now, let's apply DBSCAN once again and LOF to the new dataset (with one
      ↪cluster downsampled) and let's compare the results.

fig, ax = plt.subplots(1, 4, figsize=(20, 4), sharey=True)

ax[0].scatter(X[:, 0], X[:, 1], c='c', edgecolors='k')
ax[0].set_xlabel('Feature 1')
ax[0].set_ylabel('Feature 2')
ax[0].grid()
ax[0].set_axisbelow(True)
```

```

ax[1].scatter(X_new[:, 0], X_new[:, 1], c='c', edgecolors='k')
ax[1].set_xlabel('Feature 1')
ax[1].set_ylabel('Feature 2')
ax[1].grid()
ax[1].set_axisbelow(True)

# DBSCAN
model = DBSCAN(eps=15, min_samples=5)
y_pred = model.fit_predict(X_new)

ax[2].scatter(X_new[y_pred>-1, 0], X_new[y_pred>-1, 1], c='c', edgecolors='k')
ax[2].scatter(X_new[y_pred== -1, 0], X_new[y_pred== -1, 1], c='r', marker='x')
ax[2].set_xlabel('Feature 1')
ax[2].set_ylabel('Feature 2')
ax[2].grid()
ax[2].set_axisbelow(True)

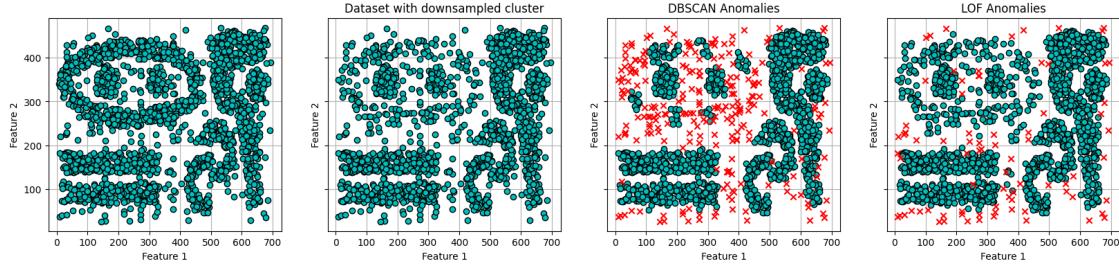
# LOF
lof = LocalOutlierFactor(n_neighbors=10, contamination=0.05) # contamination: the
# expected fraction of outliers in the data (will be used to define the
# threshold)
y_anomaly = lof.fit_predict(X_new)

ax[3].scatter(X_new[y_anomaly==1, 0], X_new[y_anomaly==1, 1], c='c', edgecolors='k')
ax[3].scatter(X_new[y_anomaly== -1, 0], X_new[y_anomaly== -1, 1], c='r', marker='x')
ax[3].set_xlabel('Feature 1')
ax[3].set_ylabel('Feature 2')
ax[3].grid()
ax[3].set_axisbelow(True)

ax[1].set_title('Original dataset')
ax[1].set_title('Dataset with downsampled cluster')
ax[2].set_title('DBSCAN Anomalies')
ax[3].set_title('LOF Anomalies')

```

[17]: Text(0.5, 1.0, 'LOF Anomalies')



0.3.4 Tree-based methods

Isolation Forests are ensemble methods that isolate anomalies by randomly partitioning the data using random decision trees. Anomalies are easier to isolate and thus have shorter average path lengths in the trees.

```
[18]: from sklearn.ensemble import IsolationForest

fig, ax = plt.subplots(2, 2, figsize=(10, 8))

isoforest = IsolationForest(n_estimators=100, contamination=0.05, random_state=1)
y_anomaly = isoforest.fit_predict(X_blob)

ax[0, 0].scatter(X_blob[y_anomaly==1, 0], X_blob[y_anomaly==1, 1], c='c', edgecolors='k')
ax[0, 0].scatter(X_blob[y_anomaly==-1, 0], X_blob[y_anomaly==-1, 1], c='r', marker='x')

# Heatmap of anomaly scores
ax[1,0].scatter(X_blob[:, 0], X_blob[:, 1], c=isoforest.score_samples(X_blob))

XX, YY = np.meshgrid(np.linspace(X_blob[:,0].min(), X_blob[:,0].max(), 100),
                     np.linspace(X_blob[:,1].min(), X_blob[:,1].max(), 100))
Z = isoforest.score_samples(np.c_[XX.ravel(), YY.ravel()]).reshape(XX.shape)
Z = (Z - Z.min()) / (Z.max() - Z.min())
cm = ax[1,0].contourf(XX, YY, Z, levels=50, cmap='RdBu')

isoforest = IsolationForest(n_estimators=100, contamination=0.05, random_state=1)
y_anomaly = isoforest.fit_predict(X)

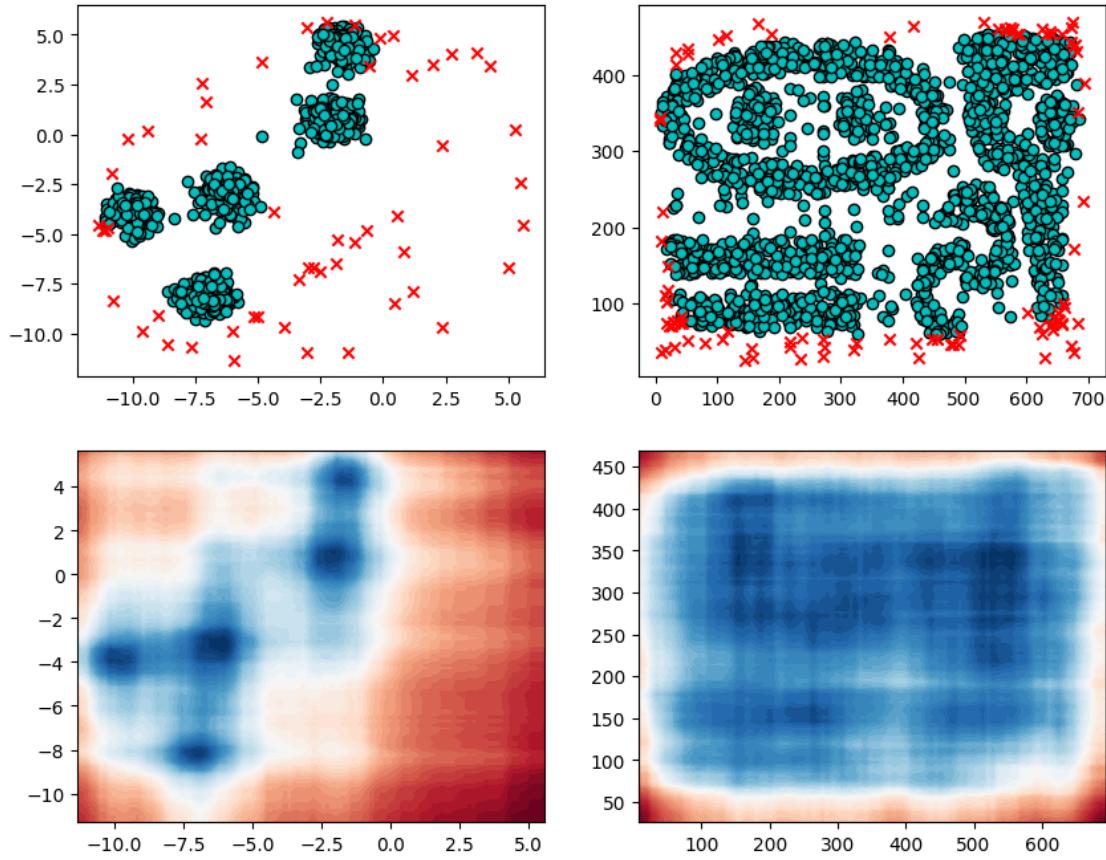
ax[0,1].scatter(X[y_anomaly==1, 0], X[y_anomaly==1, 1], c='c', edgecolors='k')
ax[0,1].scatter(X[y_anomaly==-1, 0], X[y_anomaly==-1, 1], c='r', marker='x')
```

```

# Heatmap of anomaly scores
ax[1,1].scatter(X[:, 0], X[:, 1], c=isoforest.score_samples(X))

XX, YY = np.meshgrid(np.linspace(X[:,0].min(), X[:,0].max(), 100),
                     np.linspace(X[:,1].min(), X[:,1].max(), 100))
Z = isoforest.score_samples(np.c_[XX.ravel(), YY.ravel()]).reshape(XX.shape)
Z = (Z - Z.min()) / (Z.max() - Z.min())
cm = ax[1,1].contourf(XX, YY, Z, levels=50, cmap='RdBu')

```



0.3.5 Boundary-based methods

Some approaches (e.g., One-Class SVMs) learn a decision boundary that separates normal data points from anomalies. Points that fall outside this boundary are considered anomalies.

For these approaches, it is important to set the expected fraction of anomalies in the dataset ν , as this influences the shape of the decision boundary (as ν defines how many points can fall outside the learned boundary?)

You can try to apply One-Class SVMs to the previous dataset. Try to change the “kernel” hyperparameter (which defines the shape of the decision boundary). What do the various kernels do?

Next, try to chnge the fraction of anomalies detected. How does it affect the decision boundary?

0.3.6 Neural Network-based methods

A simple approach is to use Autoencoders, which are neural networks trained to reconstruct their input. A Reconstruction Error (RE) can be computed as the “distance” between the input and the reconstructed output (e.g., Mean Squared Error for continuous variables, Binary Cross Entropy for binary variables, etc.).

```
[19]: import torch
import torch.nn as nn
from torchvision.datasets import MNIST
from torch.utils.data import TensorDataset, DataLoader
```

```
[20]: dataset = MNIST(root='data', train=True, download=True) # only use training set
       ↪for convenience
X = dataset.data.reshape(-1, 28*28) / 255.0 # normalize to [0, 1] and flatten
       ↪(==> 60000, 784), i.e., treat each image as a vector of 784 features
print(X.shape)
```

torch.Size([60000, 784])

```
[21]: class AutoEncoder(nn.Module):
    def __init__(self):
        # Hardcoding all dimensions for simplicity, we technically could pass
        ↪them as parameters
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(784, 256),
            nn.ReLU(),
            nn.Linear(256, 64),
            nn.ReLU(),
            nn.Linear(64, 10),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Linear(10, 64),
            nn.ReLU(),
            nn.Linear(64, 256),
            nn.ReLU(),
            nn.Linear(256, 784),
            nn.Sigmoid() # assuming input is normalized between 0 and 1 (as we
        ↪previously did)
```

```

    )

def forward(self, x):
    code = self.encoder(x)
    x_rec = self.decoder(code)
    return x_rec

```

```
[22]: model = AutoEncoder()
opt = torch.optim.Adam(model.parameters(), lr=1e-2)
loss_fn = nn.MSELoss()
ds = TensorDataset(X)
dl = DataLoader(ds, batch_size=256, shuffle=True)

n_epochs = 10

for epoch in range(n_epochs):
    epoch_loss = 0.0
    for x_batch, in dl:
        opt.zero_grad()
        batch_rec = model(x_batch)
        loss = loss_fn(batch_rec, x_batch)
        loss.backward()
        opt.step()
        epoch_loss += loss.item()
    epoch_loss /= len(dl)
    print(f"Epoch {epoch+1}/{n_epochs}, Loss: {epoch_loss:.6f}")
```

Epoch 1/10, Loss: 0.055307
 Epoch 2/10, Loss: 0.039010
 Epoch 3/10, Loss: 0.034029
 Epoch 4/10, Loss: 0.032201
 Epoch 5/10, Loss: 0.031260
 Epoch 6/10, Loss: 0.030655
 Epoch 7/10, Loss: 0.030282
 Epoch 8/10, Loss: 0.029869
 Epoch 9/10, Loss: 0.029597
 Epoch 10/10, Loss: 0.029360

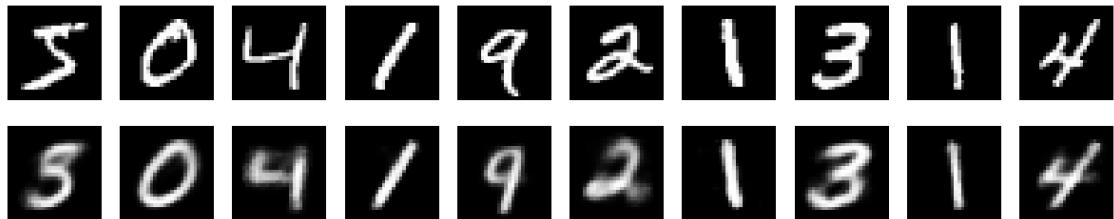
```
[23]: with torch.no_grad():
    X_rec = model(X).numpy()

    X_np = X.numpy()
```

```
[24]: fig, ax = plt.subplots(2, 10, figsize=(15, 3))

for i in range(10):
    ax[0, i].imshow(X_np[i].reshape(28, 28), cmap='gray')
```

```
ax[0, i].set_axis_off()  
  
ax[1, i].imshow(X_rec[i].reshape(28, 28), cmap='gray')  
ax[1, i].set_axis_off()
```



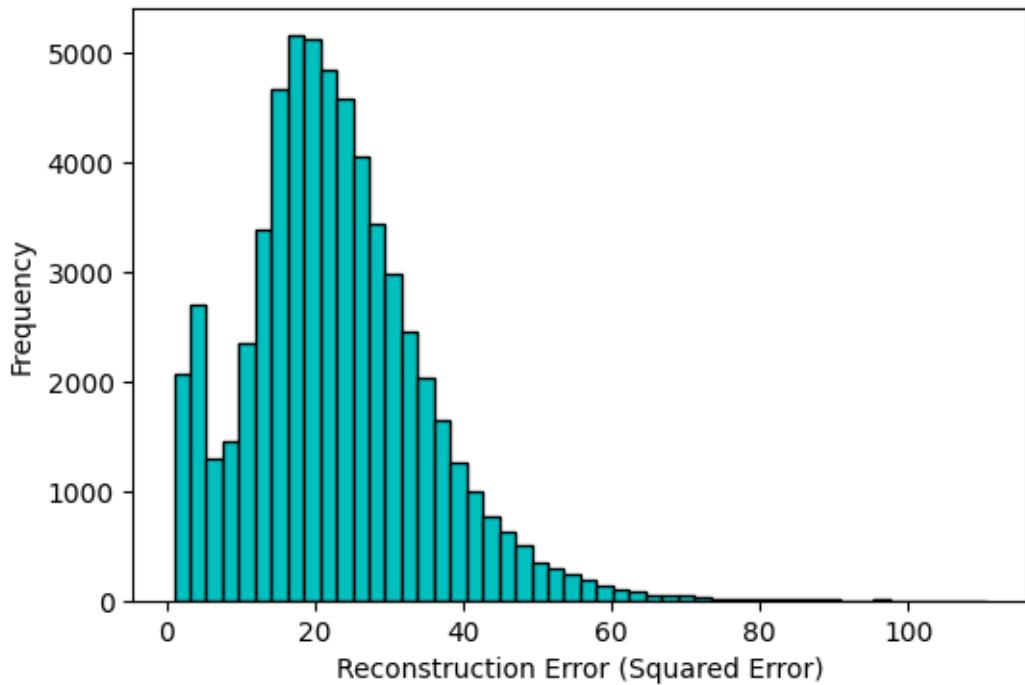
The RE (for instance, squared error across all pixels, in this case) can be used to compute how “poorly” each sample is reconstructed by the Autoencoder. High RE values can be used to flag anomalies in the dataset.

Let’s look at the worst-reconstructed images according to the MSE RE.

```
[25]: re = ((X_np - X_rec) ** 2).sum(axis=1) # Mean Squared Error per sample
```

```
[26]: fig, ax = plt.subplots(figsize=(6,4))  
ax.hist(re, bins=50, color='c', edgecolor='k')  
ax.set_ylabel('Frequency')  
ax.set_xlabel('Reconstruction Error (Squared Error)')
```

```
[26]: Text(0.5, 0, 'Reconstruction Error (Squared Error)')
```



```
[27]: fig, ax = plt.subplots(2, 10, figsize=(15, 3))

for i, pos in enumerate(re.argsort()[-10:]):
    ax[0, i].imshow(X_np[pos].reshape(28, 28), cmap='gray')
    ax[0, i].set_axis_off()

    ax[1, i].imshow(X_rec[pos].reshape(28, 28), cmap='gray')
    ax[1, i].set_axis_off()
```

