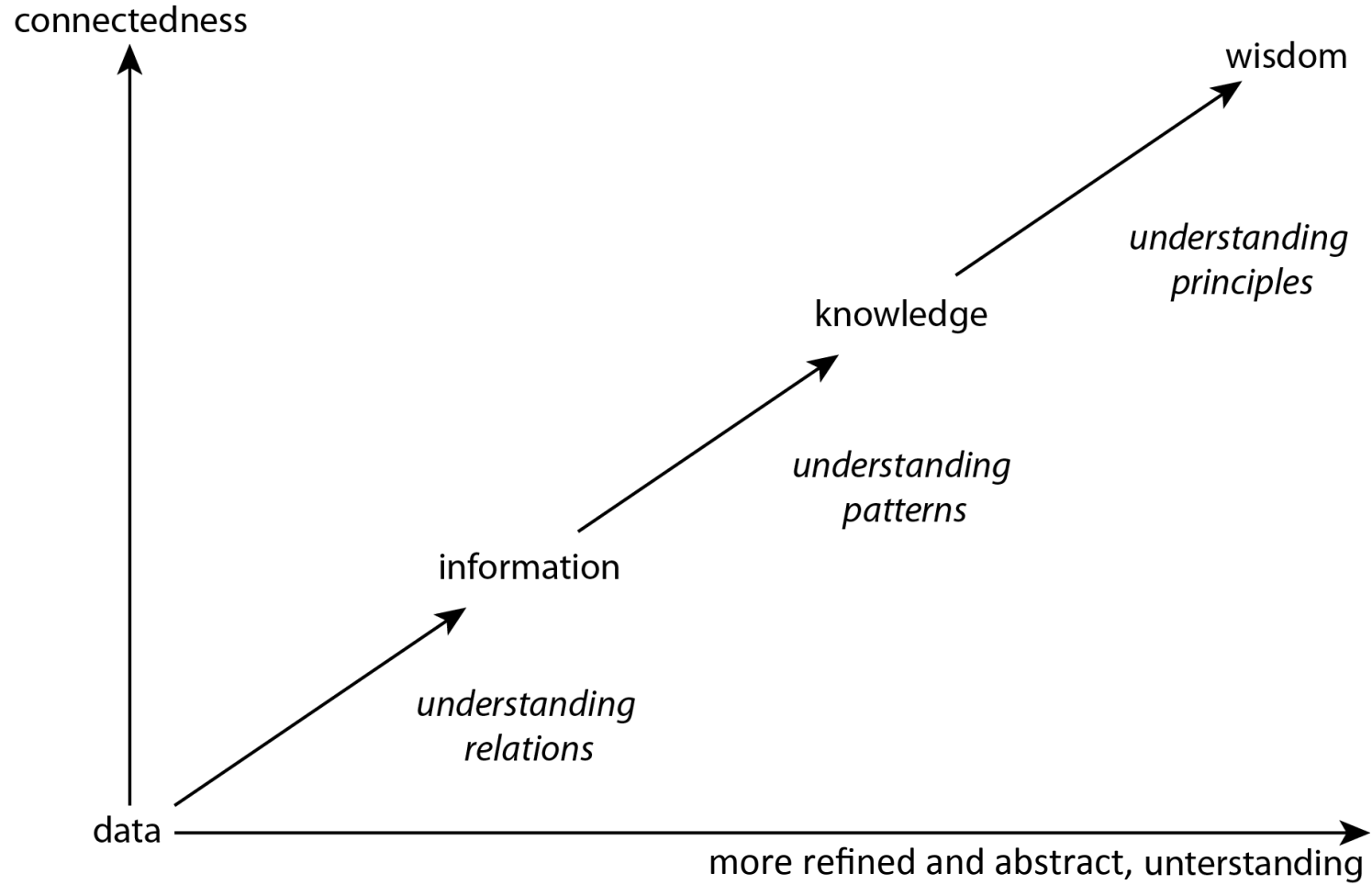


Data Mining and Pattern Recognition

Salvatore Orlando, Andrea Torsello,
Filippo Bergamasco

Information Hierarchy



A (Facetious) Example

- Data
 - 37°, 38.5°, 39.3°, 40°, ...
- Information
 - Hourly body temperature: 37°, 38.5°, 39.3°, 40°, ...
- Knowledge
 - If you have a temperature above 37°, you most likely have a fever
- Wisdom (actionable)
 - If you have a fever and don't feel well, go see a doctor

Content of CH resources

- Some CH content can be stored lossless in digital libraries and databases
 - text documents, digital photos, music, etc
- Digitalization may cause a loss of information
 - e.g., image resolution, audio/video quality
- In most cases content is not “digital” and surrogates are used
 - images for paintings, manuscripts, artifacts
 - 3D models for buildings, sculptures, etc.

DATA, FEATURES

What is Data we Analyze?

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

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

Objects

Unstructured Raw Data

- Consider a **digital text** or a **digital image**
 - We cannot recognize a record structure (a collection of a fixed number of data, with metadata describing them)
- A **digital text** is a list of characters encoded in by some kind of **numerical encoding** system (e.g., ASCII)
- A **digital image** is a numeric representation of a two-dimensional image
 - Raster images have a finite set of digital values, called picture elements or **pixels** (rows × columns)
 - Pixels hold quantized values that represent the **brightness** of a given **color** at any specific point.

INFORMATION RETRIEVAL

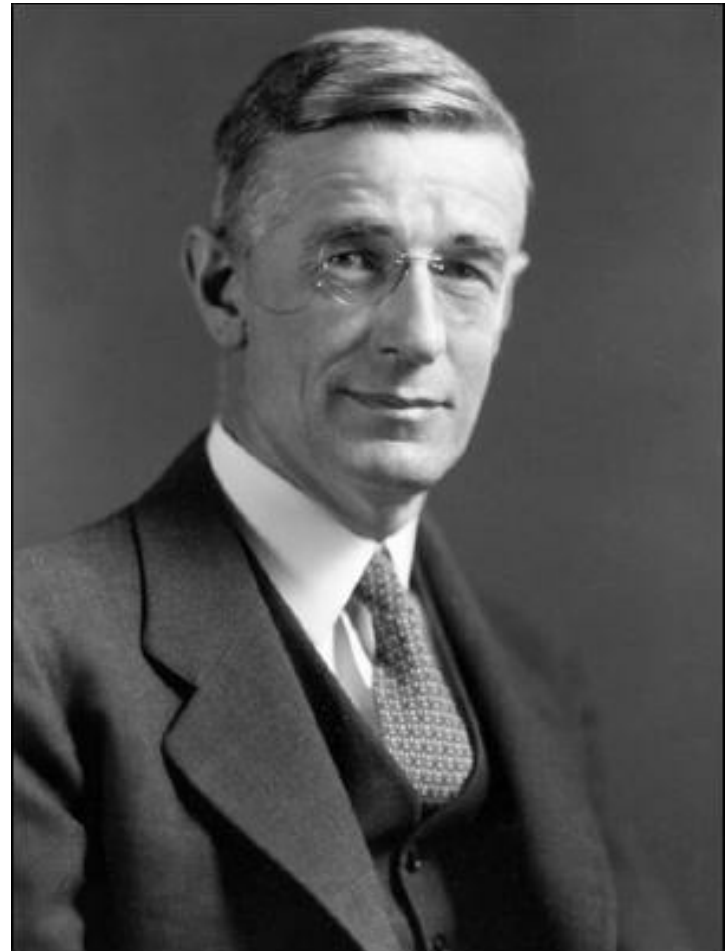
Digital data deluge

- Plain text (documents and portions thereof)
- XML and structured documents
- Web docs
- Images
- Audio (sound effects, songs, etc.)
- Video
- Graphs/networks
- Source code
- Apps/Web services



IR Origins

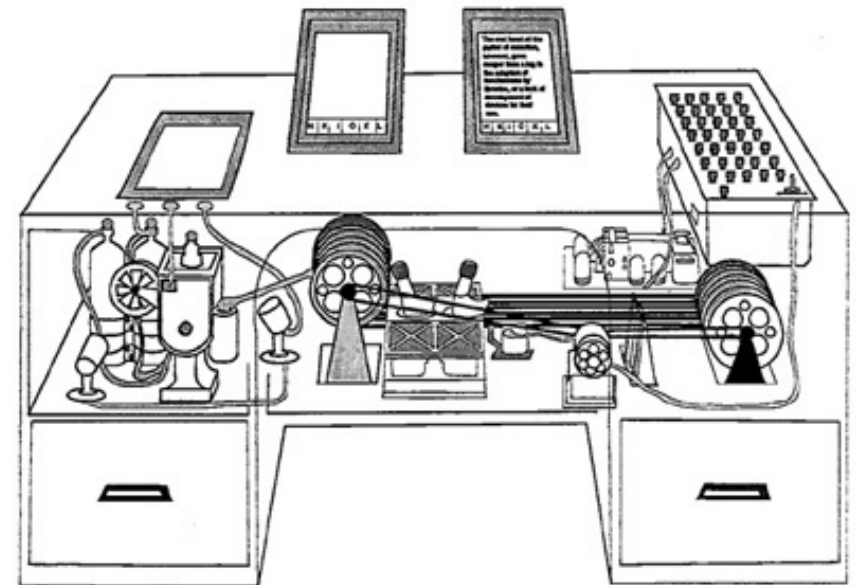
- The “information overload” problem is much older than you may think, much before the WEB
- Origins in period immediately after World War II
 - Tremendous scientific progress during the war
 - Rapid growth in amount of scientific publications available
- The “Memex Machine”
<http://en.wikipedia.org/wiki/Memex>
 - Conceived by Vannevar Bush, President Roosevelt's science advisor
 - Foreshadows the development of hypertext (the Web) and information retrieval systems



The Memex Machine

Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and, to coin one at random, “memex” will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

It consists of a desk, and while it can presumably be operated from a distance, it is primarily the piece of furniture at which he works. On the top are slanting translucent screens, on which material can be projected for convenient reading. There is a keyboard, and sets of buttons and levers. Otherwise it looks like an ordinary desk.



(Vannevar Bush; As We May Think;
Atlantic Monthly; July 1945)

The Central Problem in IR

Information Seeker



Concepts



Query Terms

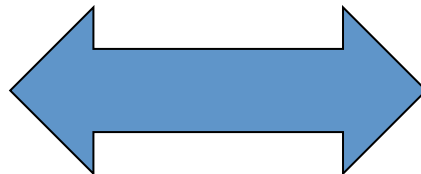
Authors/Content providers



Concepts

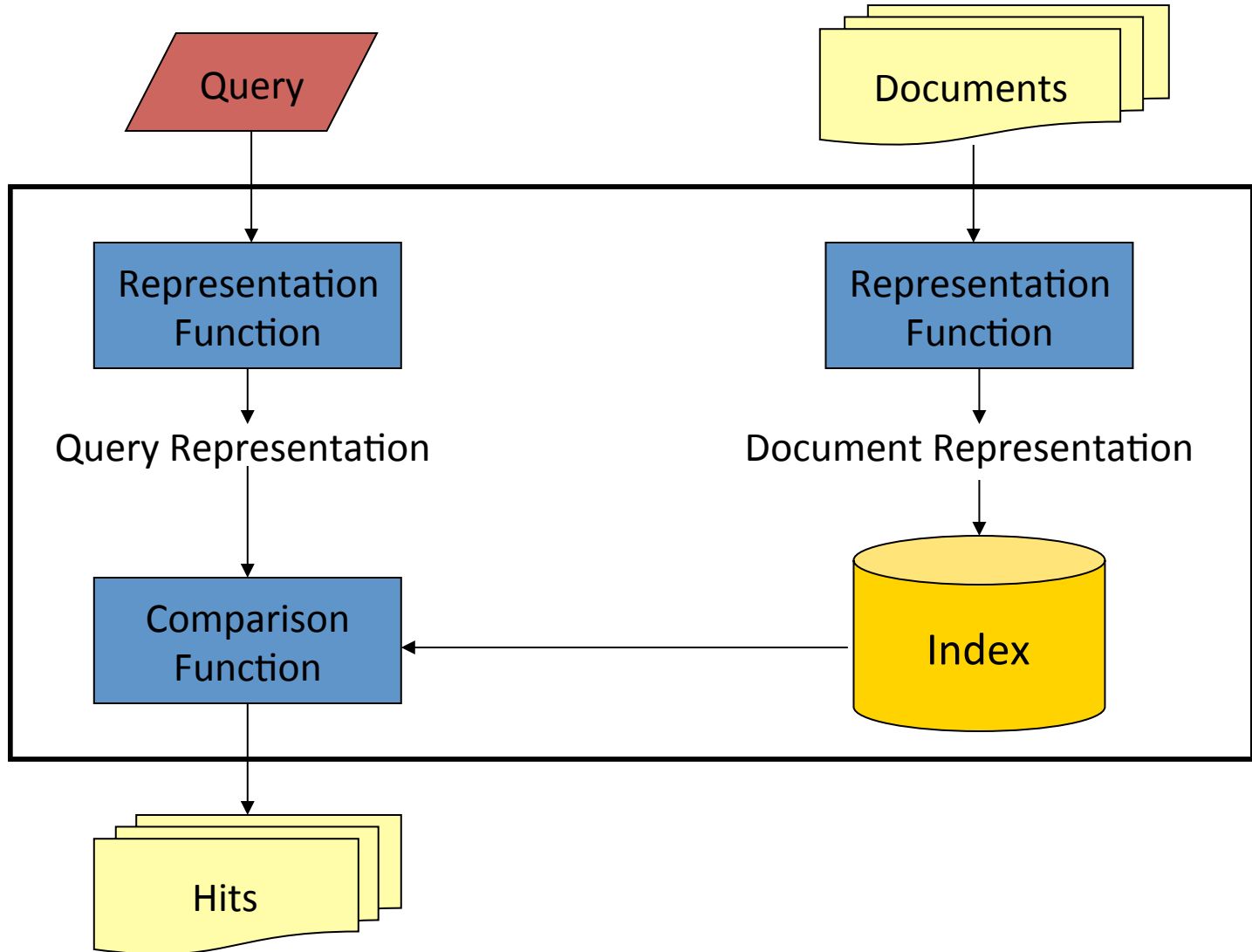


Document Terms



Do these represent the same concepts?

Inside The IR Black Box



Web Searches

- WSE
 - index **billions** of pages
 - Answer **hundreds of millions** of queries per day
 - In less than **0,5 sec.** per query
- Users
 - Submit **short queries** (on avg. 2.5 terms), often with orthographic errors
 - Expect to receive the **most relevant results** of the Web
 - In a **blink of eye**



Feature extraction

- Feature extraction is concerned with
 - Representing each unstructured data element in terms of a record/vector of alphanumeric values, also called **features**
 - It requires to manipulate raw data to extract features
- Why is it needed to represent data as sets of feature?
 - The reason is that many Information retrieval, Data mining, and Pattern recognition methods need to use these representations to apply their algorithms

How do we represent text?

- How do we represent the complexities of language?
 - Keeping in mind that computers don't “understand” documents or queries
- Simple, yet effective approach: “bag of words”
 - Treat all the words in a document as index terms for that document
 - Assign a “weight” to each term based on its “importance”
 - Disregard order, structure, meaning, etc. of the words

Vector Representation

- “Bags of words” can be represented as vectors
 - Why? Computational efficiency, ease of manipulation
- A vector is a set of values recorded in any consistent order

“The quick brown fox jumped over the lazy dog’s back”

→ [1 1 1 1 1 1 1 1 2]

1st position corresponds to “back”

2nd position corresponds to “brown”

3rd position corresponds to “dog”

4th position corresponds to “fox”

5th position corresponds to “jump”

6th position corresponds to “lazy”

7th position corresponds to “over”

8th position corresponds to “quick”

9th position corresponds to “the”

Representing Documents

Document 1

The quick brown
fox jumped over
the lazy dog's
back.

Document 2

Now is the time
for all good men
to come to the
aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Feature extraction to Represent Text Documents

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Document Vector, where each feature represents the number of occurrences of each term

Basic IR Models

- Boolean model
 - Based on the notion of sets
 - Documents are retrieved only if they satisfy Boolean conditions specified in the query
 - Does not impose a ranking on retrieved documents
 - Exact match
- Vector space model
 - Based on geometry, the notion of vectors in high dimensional space
 - Documents are ranked based on their similarity to the query (ranked retrieval)
 - Best/partial match

Boolean Retrieval

- Weights assigned to terms are either “0” or “1”
 - “0” represents “absence”: term **isn’t** in the document
 - “1” represents “presence”: term **is** in the document
- Build queries by combining terms with Boolean operators
 - AND, OR, NOT
- The system returns all documents that satisfy the query

Boolean View of a Collection

Term	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8
aid	0	0	0	1	0	0	0	1
all	0	1	0	1	0	1	0	0
back	1	0	1	0	0	0	1	0
brown	1	0	1	0	1	0	1	0
come	0	1	0	1	0	1	0	1
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0
good	0	1	0	1	0	1	0	1
jump	0	0	1	0	0	0	0	0
lazy	1	0	1	0	1	0	1	0
men	0	1	0	1	0	0	0	1
now	0	1	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1
party	0	0	0	0	0	1	0	1
quick	1	0	1	0	0	0	0	0
their	1	0	0	0	1	0	1	0
time	0	1	0	1	0	1	0	0

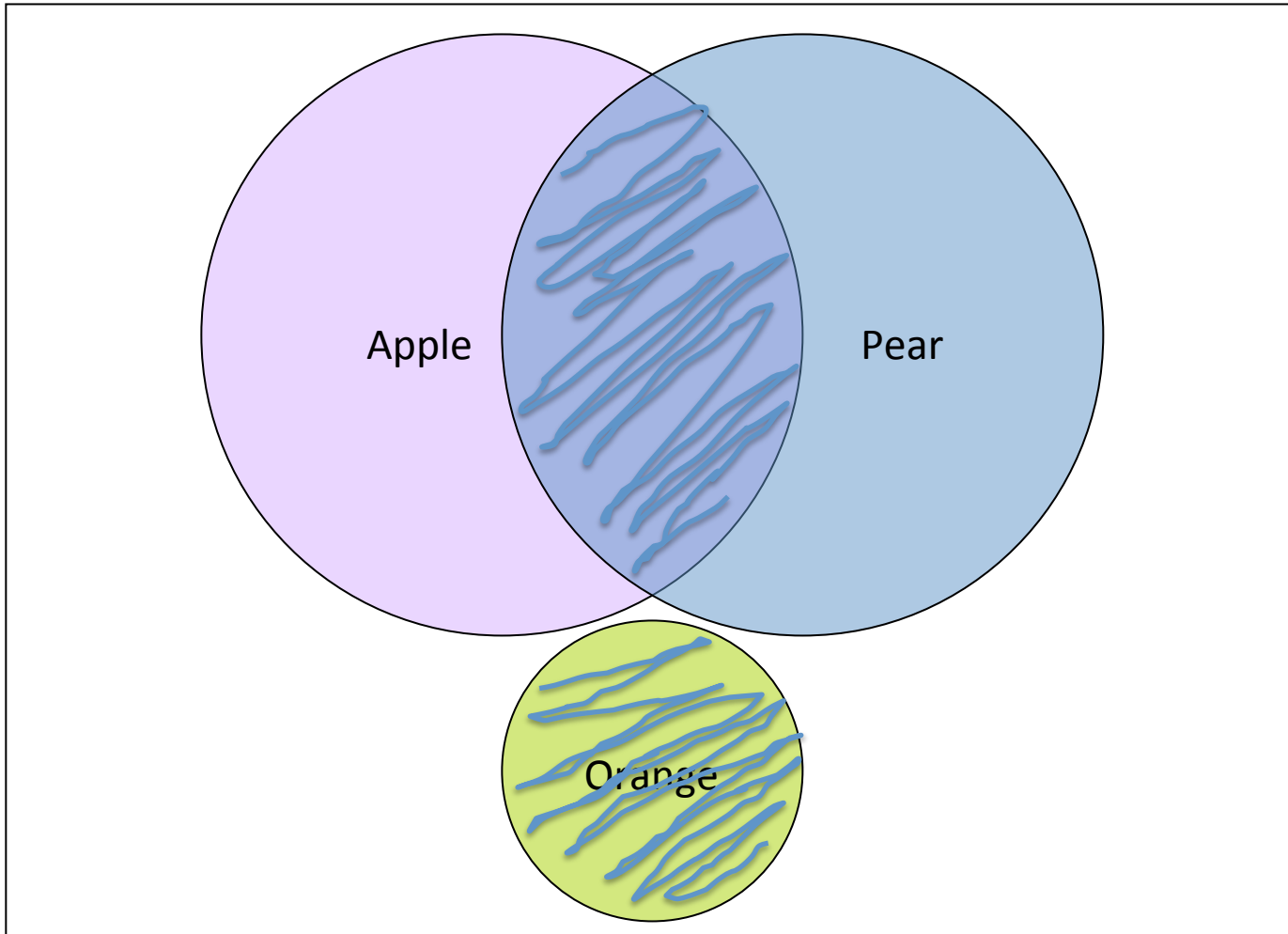
Each column represents the view of a particular document: What terms are contained in this document?

Each row represents the view of a particular term: What documents contain this term?

To execute a query, pick out rows corresponding to query terms and then apply logic table of corresponding Boolean operator

AND/OR/NOT

All documents



(Apple and Pear) or Orange

Logic Tables

A \ B	0	1
0	0	1
1	1	1

A OR B

B	0	1
	1	0

NOT B

A \ B	0	1
0	0	0
1	0	1

A AND B

A \ B	0	1
0	0	0
1	1	0

A NOT B
(= A AND NOT B)

Term-document incidence matrices

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

1 if play contains
word, 0 otherwise

Brutus AND Caesar AND (NOT Calpurnia)

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for **Brutus**, **Caesar** and **Calpurnia** (complemented) → bitwise *AND*.

– 110100 *AND*

– 110111 *AND*

– 101111 =

– **100100**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus AND Caesar AND (NOT Calpurnia)

Answers to query

- Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

- Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius **Caesar** I was killed i' the
Capitol; **Brutus** killed me.



See: <http://www.opensourceshakespeare.org>
<http://sydney.edu.au/engineering/it/~matty/Shakespeare/test.html>

Why Boolean Retrieval Works

- Boolean operators *approximate* natural language
 - Find documents about a good party that is not over
- AND can discover relationships between concepts
 - good party
- OR can discover alternate terminology
 - excellent party, wild party, etc.
- NOT can discover alternate meanings
 - Democratic party

See: <http://sydney.edu.au/engineering/it/~matty/Shakespeare/test.html>

for a search engine on Shakespeare exploiting the Boolean model

See: <http://www.perunaenciclopediadantescadigitale.eu:8080/dantesearch/>

for a search engine on Dante exploiting the Boolean model

Why Boolean Retrieval Fails

- Natural language is way more complex
- AND “discovers” nonexistent relationships
 - Terms in different sentences, paragraphs, ...
- Guessing terminology for OR is hard
 - good, nice, excellent, outstanding, awesome, ...
- Guessing terms to exclude is even harder!
 - Democratic party, party to a lawsuit, ...

Strengths and Weaknesses

Strengths

- Precise, if you know the right strategies
- Precise, if you have an idea of what you're looking for
- Efficient for the computer

Weaknesses

- Users must learn Boolean logic
- Boolean logic insufficient to capture the richness of language
- No control over size of result set: either too many documents or none
 - All documents in the result set are considered “equally good”
 - Does not fit huge collections
- No support for partial matches

Ranked Retrieval

- Order documents by how likely they are to be relevant to the information need
 - Present hits one screen at a time
 - Closer to how humans think: some documents are “better” than others
 - Closer to user behavior: users can decide when to stop reading
 - Fits better huge collections of documents



Similarity-Based Queries

- Rank documents by their similarity with the query
- Treat the query as if it were a document
 - **Free text queries**: Rather than a query language of operators and expressions, the user's query is just one or more words in a human language
- **Score** its **similarity** to each document in the collection
- **Rank** the documents by similarity score
- Documents need not have all query terms
 - Although documents with more query terms should be “better”

Documents as vectors

- So we have vector space with $|V|$ dimensions
 - Terms are axes of the space
 - Documents (and Queries) are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
 - These are very sparse vectors - most entries are zero.

Queries as vectors

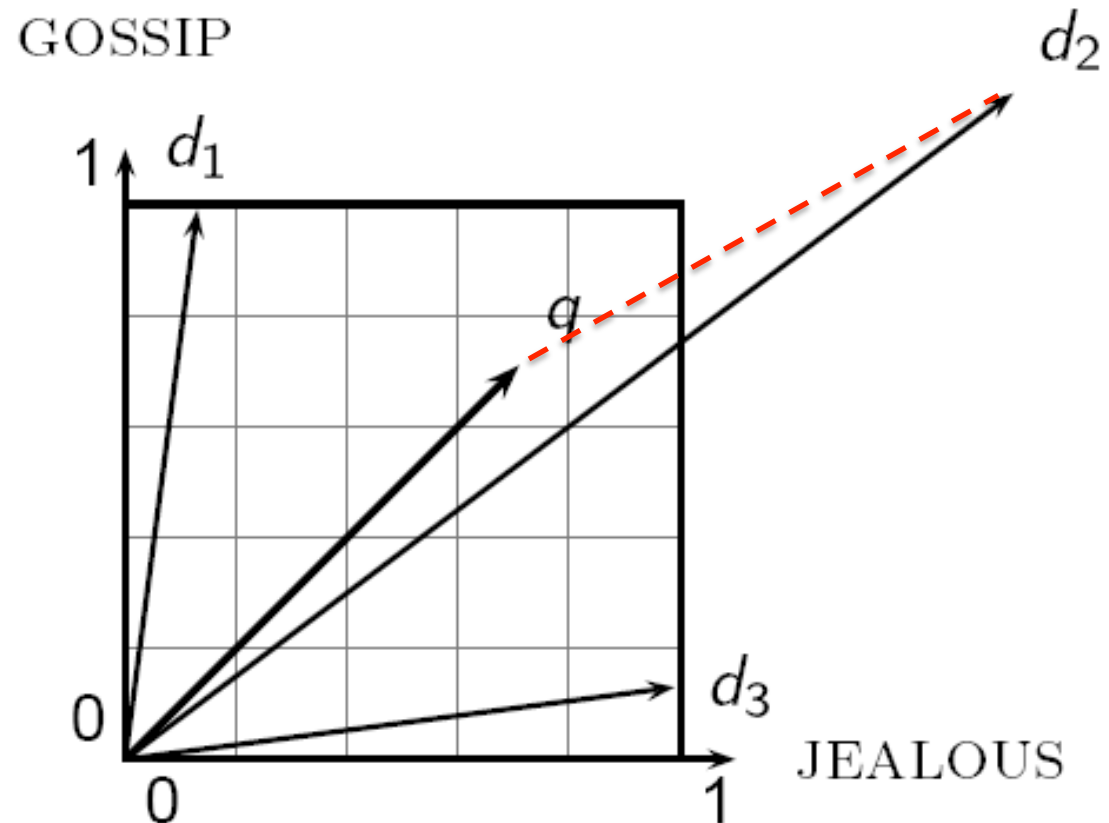
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = **similarity** of vectors
- proximity $\approx$ inverse of **distance**

Formalizing vector space proximity

- First cut: distance between two points
 - (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

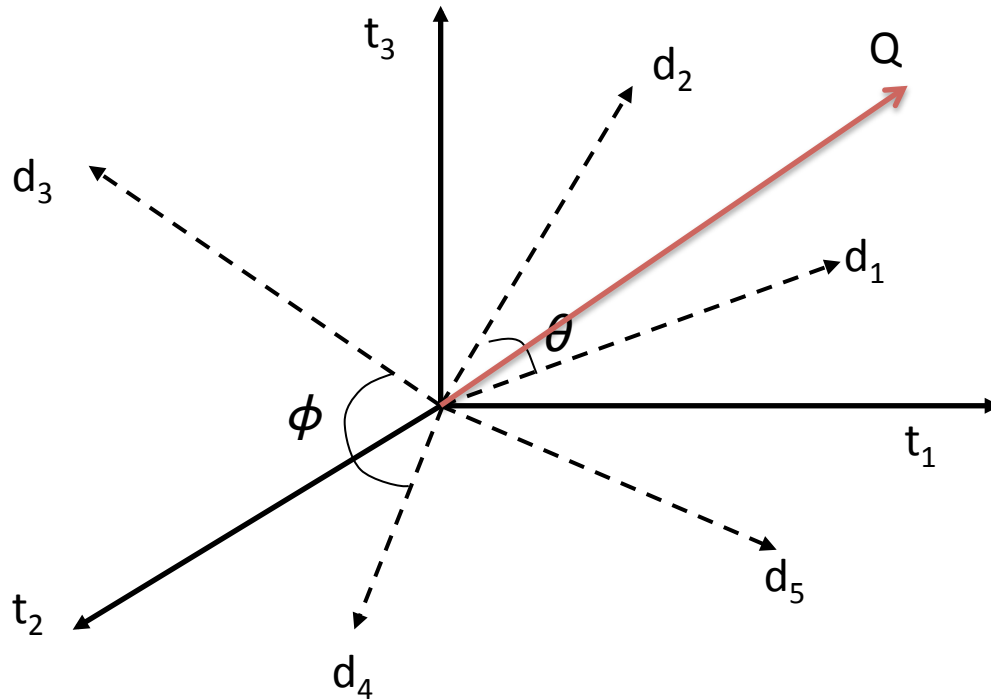
The Euclidean distance between q and $\vec{d}_2$ is large even though the distribution of terms in the query q and the distribution of terms in the document $\vec{d}_2$ are very similar.



Use angle instead of distance

- Thought experiment: take a document d and append it to itself. Call this document d' .
 - “Semantically” d and d' have the same content
 - The Euclidean distance between the two documents can be quite large
 - The angle between the two documents is 0, corresponding to maximal similarity.
- **Key idea:** Rank documents according to angle with query
 - Rank documents in decreasing order of the angle between query and document

Vector Space Model



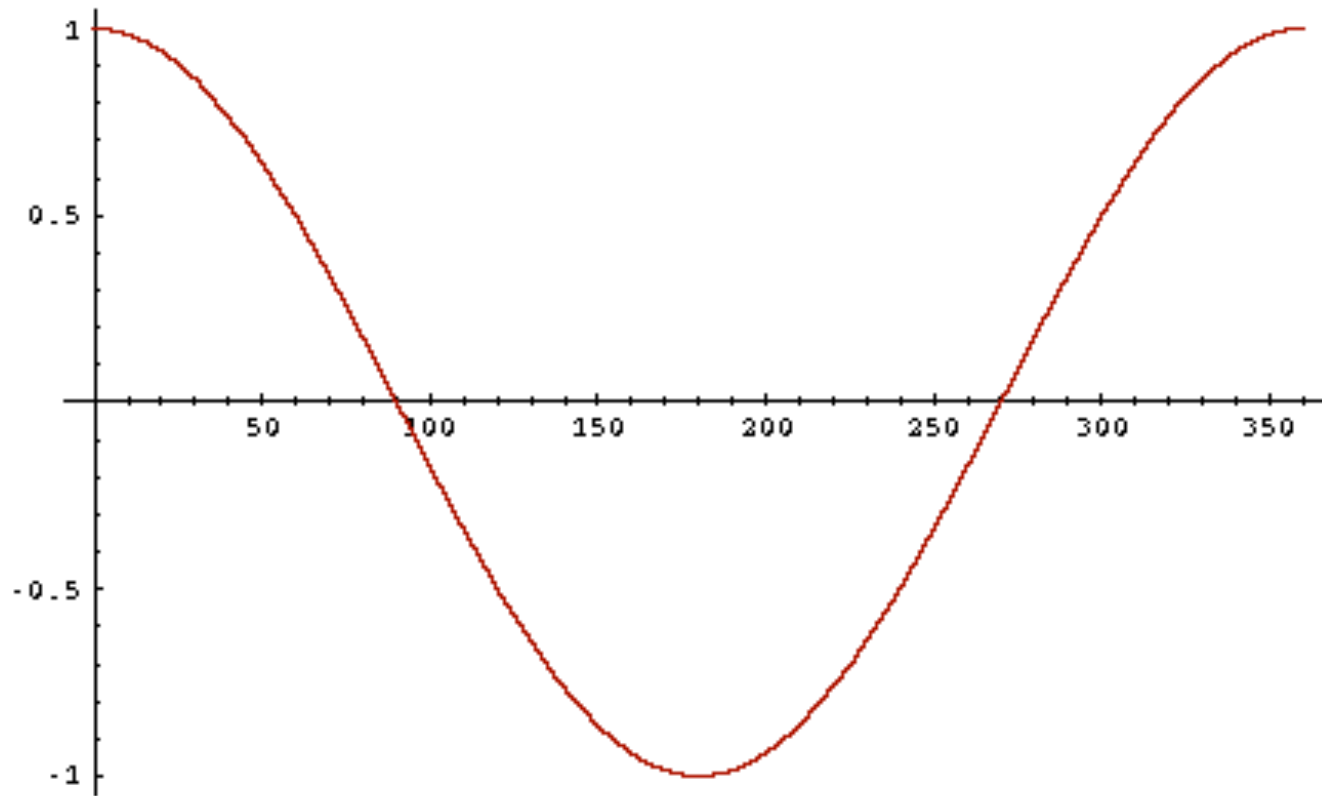
Postulate: Documents that are “close together” in vector space “talk about” the same things

Therefore, retrieve documents based on how close the document is to the query (e.g., similarity $\sim$ cosine of the angle)

From angles to cosines

- The following two notions are equivalent.
 - Rank documents in decreasing order of the angle between query and document
 - Rank documents in increasing order of $\cos(\text{angle}(\text{query}, \text{document}))$
- Cosine is a monotonically decreasing function for the interval $[0^\circ, 180^\circ]$

From angles to cosines



- But how – *and why* – should we be computing cosines?

How do we weight doc terms in the vectors?

- Here's the intuition:
 - Terms that appear often in a document should get high weights

The more often a document contains the term “dog”, the more likely that the document is “about” dogs.
 - Terms that appear in many documents should get low weights

Words like “the”, “a”, “of” appear in (nearly) all documents.
- How do we capture this mathematically?
 - Term frequency
 - Inverse document frequency

TFIDF

For a term i in document j :

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

N = total number of documents



Cosine similarity amongst 3 documents

- How similar are the novels

- SaS: *Sense and Sensibility*
- PaP: *Pride and Prejudice*
- WH: *Wuthering Heights*?

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)



Jane Austen (1775 –1817)



Emily J. Brontë (1818 –1848)

3 documents example contd.

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

$$\cos(\text{SaS}, \text{PaP}) \approx$$

$$0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0$$

$$\approx 0.94$$

$$\cos(\text{SaS}, \text{WH}) \approx 0.79$$

$$\cos(\text{PaP}, \text{WH}) \approx 0.69$$

Still data deluge

- Plain text (documents and portions thereof)
- XML and structured documents
- Web docs
- Images
- Audio (sound effects, songs, etc.)
- Video
- Graphs/networks
- Source code
- Apps/Web services



Knowledge Discovery in Databases

“The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”

Fayyad, Piatetsky-Shapiro, Smith [1996]

Knowledge Discovery in Databases

KDD is an iterative process. The more you extract knowledge from the data the more you learn to ask better questions



“The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”

Fayyad, Piatetsky-Shapiro, Smith [1996]

Knowledge Discovery in Databases

KDD is an iterative process. The more you extract knowledge from the data the more you learn to ask better questions



“The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”

Fayyad, Piatetsky-Shapiro, Smith [1996]



Not something we already know

Knowledge Discovery in Databases

KDD is an iterative process. The more you extract knowledge from the data the more you learn to ask better questions



“The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”

Fayyad, Piatetsky-Shapiro, Smith [1996]



Process leads to human insight. Visualization is a crucial part for human comprehension

Not something we already know

Knowledge Discovery in Databases

KDD is an iterative process. The more you extract knowledge from the data the more you learn to ask better questions

“The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”

Fayyad, Piatetsky-Shapiro, Smith [1996]

Process leads to human insight. Visualization is a crucial part for human comprehension

Not something we already know

Can generalize the future

Machine Learning

Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world

Usually, machine learning is focused on making prediction from examples (supervised learning) while KDD or Data Mining is focused on finding patterns (unsupervised learning).

Historically developed for different contexts (AI vs. Data Analytics), in practice based on the same ideas and techniques

Kinds of learning

Supervised learning (classification):

given a set of example input/output pairs find a function that does a good job of predicting the output associated to a new input

Unsupervised learning (clustering):

given a set of examples, with no additional information to them, group the examples into “natural” groups

Classification vs. Clustering

- Both aim at grouping objects represented as vectors/tuples/records/...
- **Classification** : supervised learning
 - Supervised knowledge: data in the training set have class labels
 - Novel records in the test set are labeled by applying a classification model, in turn learned from the training data
- **Clustering** : unsupervised learning
 - There are no examples with class labels from which we can learn
 - The goal of clustering is to find groups of related objects, if they exist, on the basis of a similarity relationship

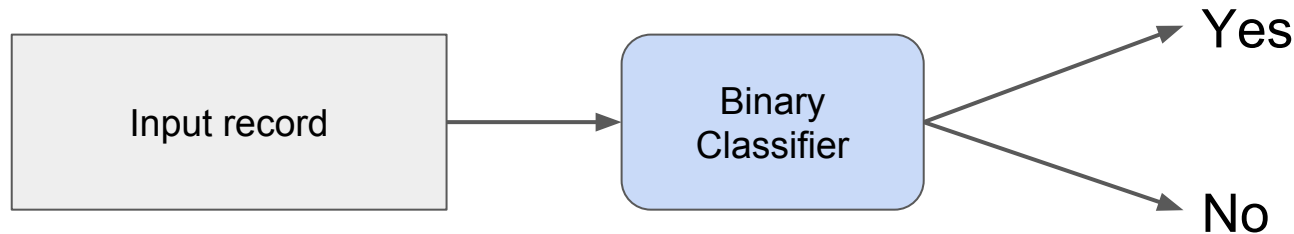
Classification: Definition

- Given a collection of records (*training set*)
 - Each *record* contains a set of *attributes*, where one of the attributes is the *class*.
 - The values of the *class label* represent the *supervised knowledge*
- *Induce a model* from the training set as a function of the values of other attributes
 - The function has to map a set of attributes X to a predefined class label y
- **Goal:** the induced model should assign a class label to previously unseen record as accurately as possible.
 - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Binary classifiers

The simplest form of classifiers are the binary classifiers:

Only two output classes: “yes” or “no”



A multi-class classifier can be created from a set of binary classifiers predicting the inclusion of each record to one of the multiple classes.

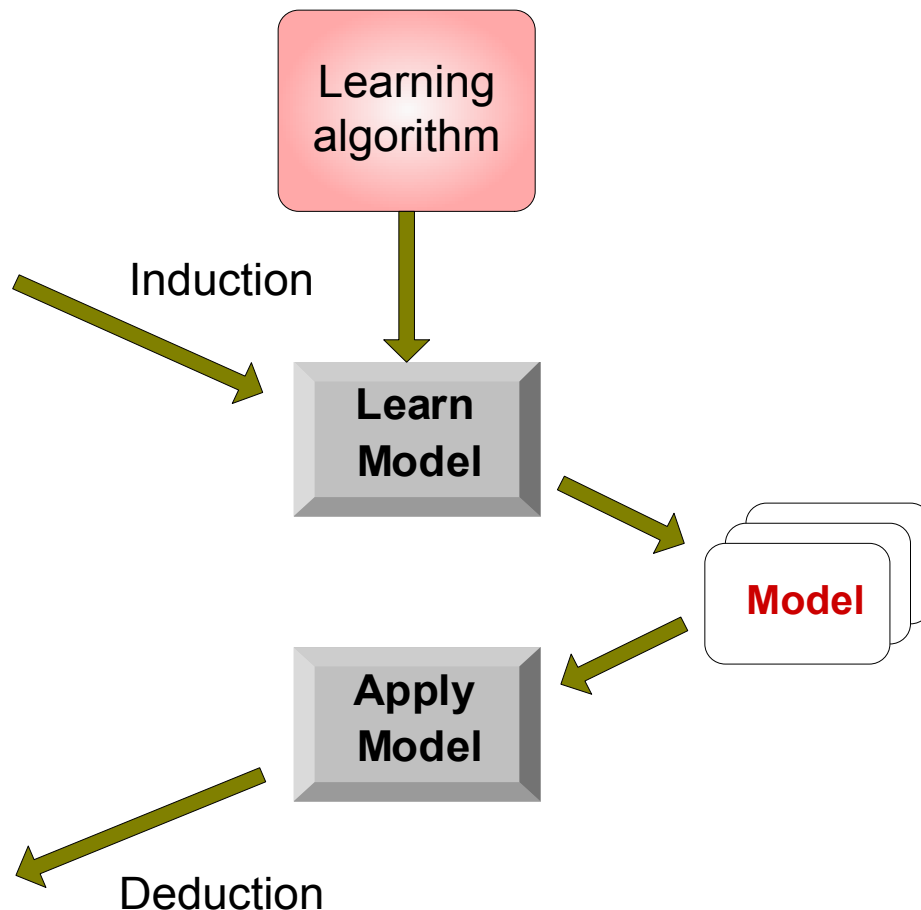
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

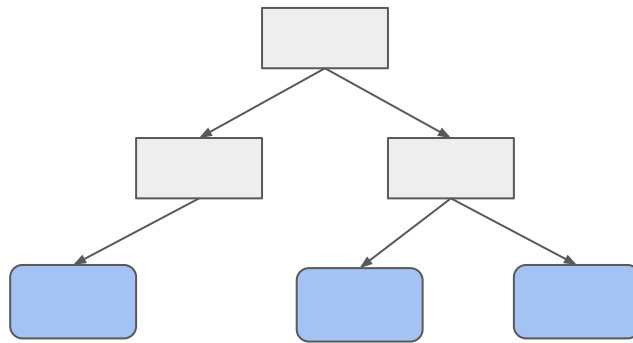
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision trees

A classical example of such model is a **decision tree**



A decision tree is a flowchart-like structure in which each internal node represents a test on an **attribute**, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

Classification

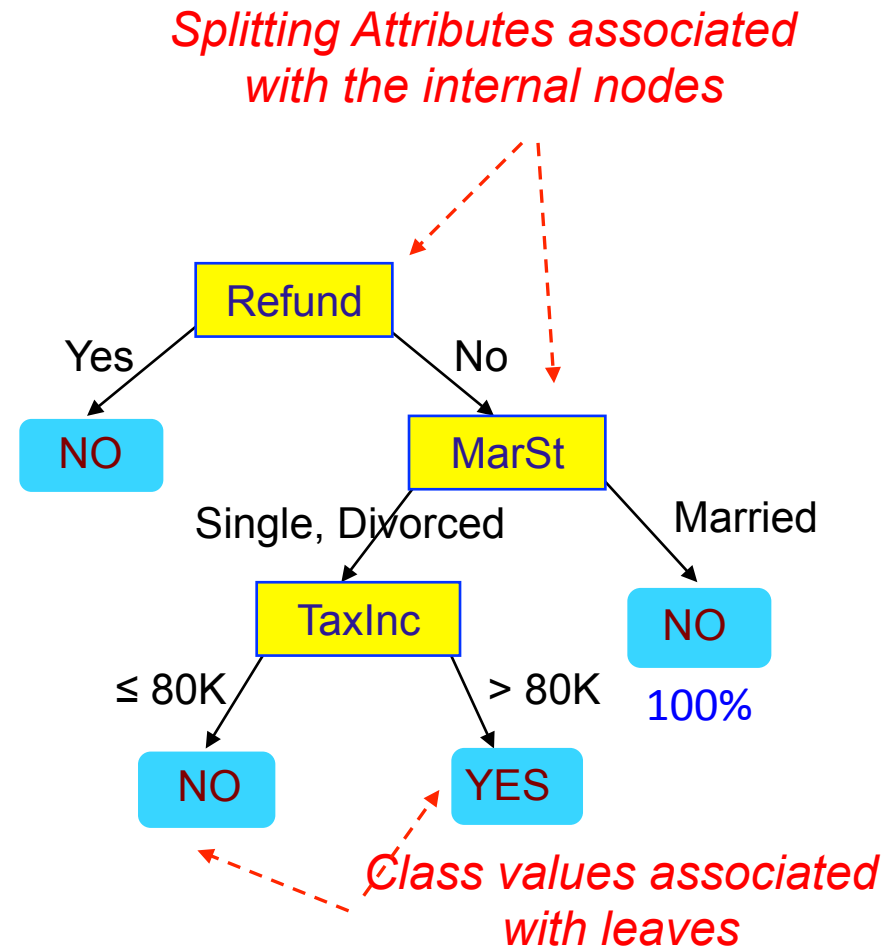
Example of a Decision Tree Model

categorical *categorical* *continuous* *class*

<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

Training Set

Model
induction



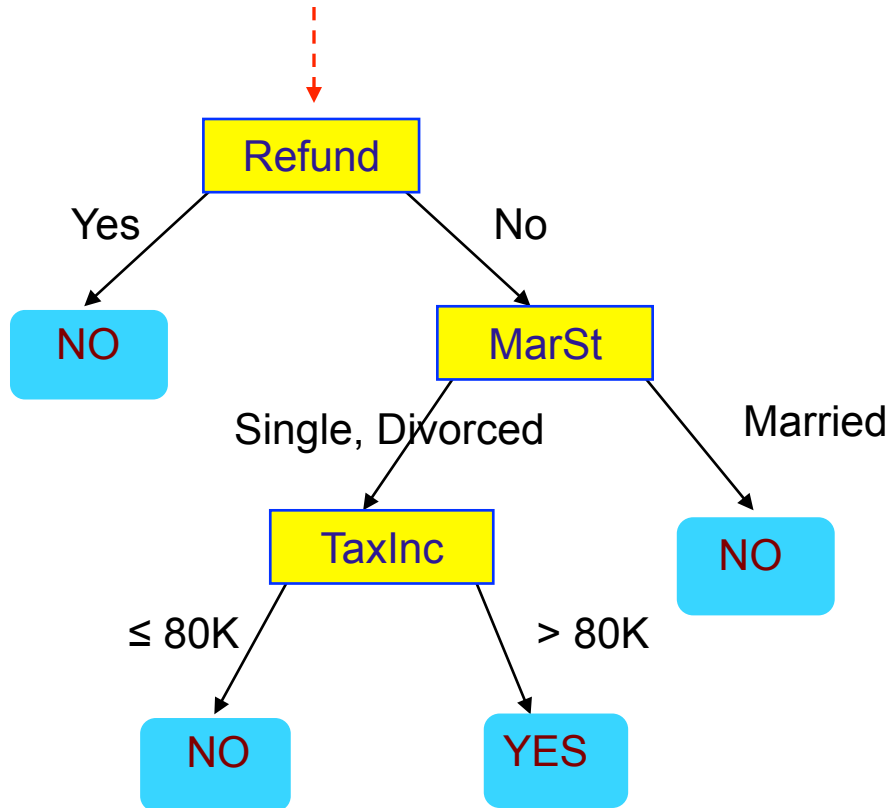
Model: Decision Tree

Apply the Model to Test Data

Test Data

Start from the root of tree.

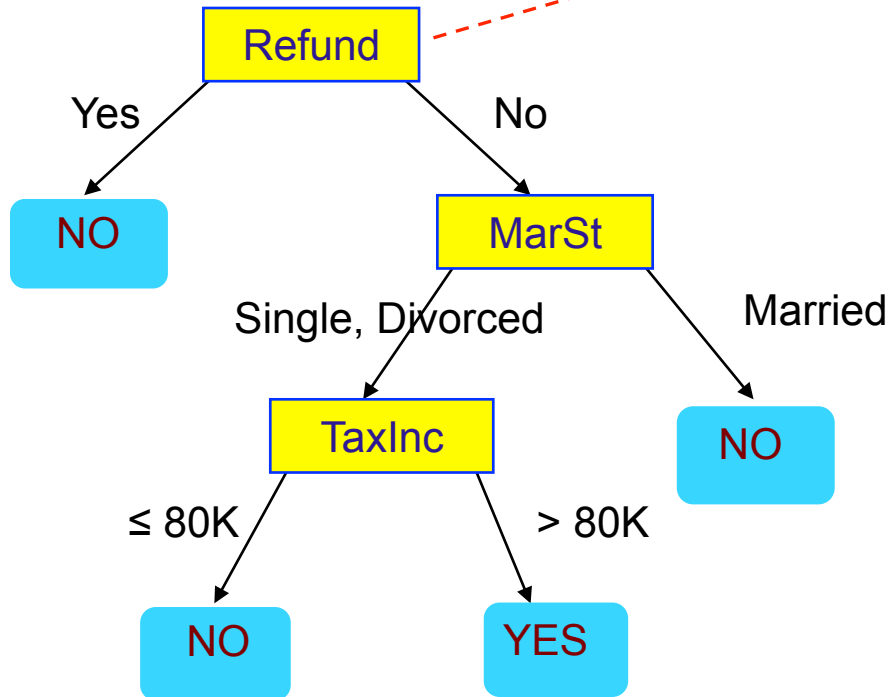
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply the Model to Test Data

Test Data

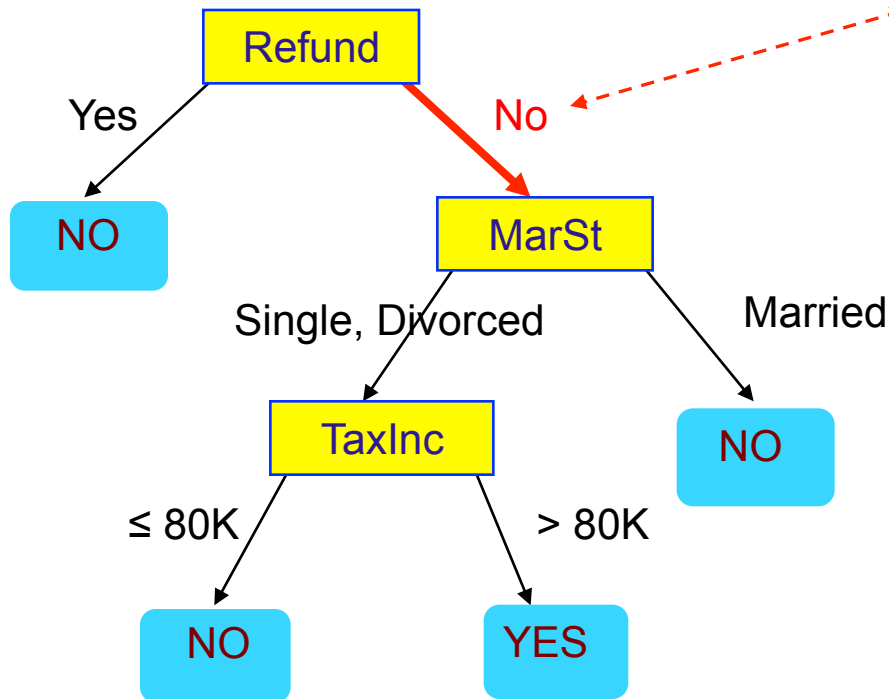
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply the Model to Test Data

Test Data

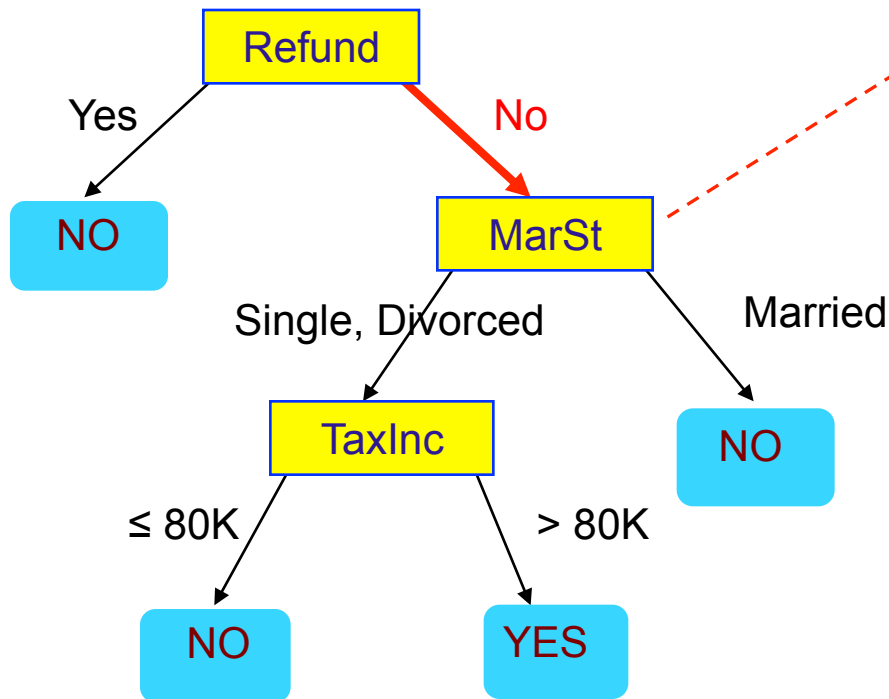
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply the Model to Test Data

Test Data

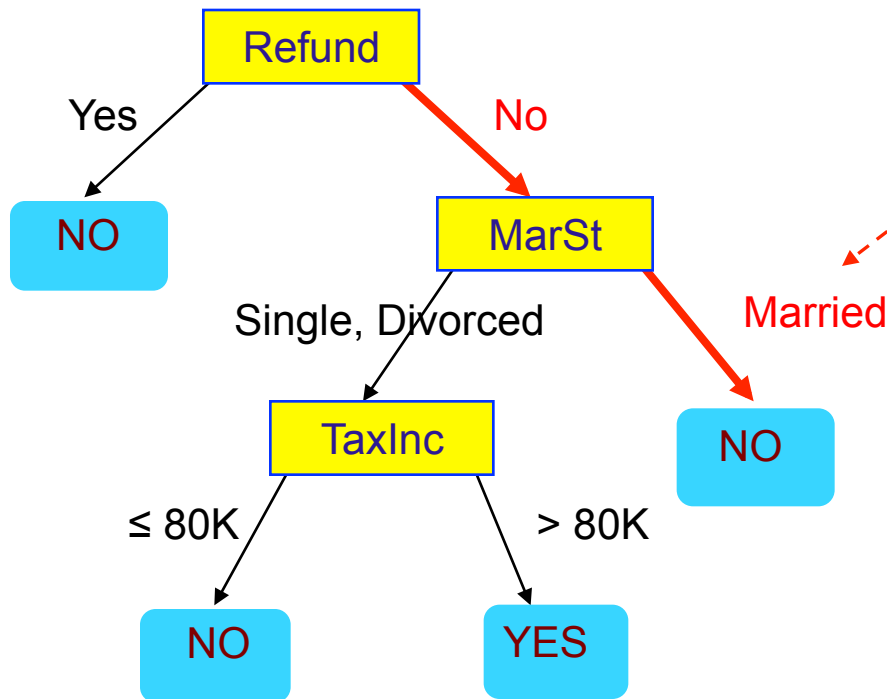
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply the Model to Test Data

Test Data

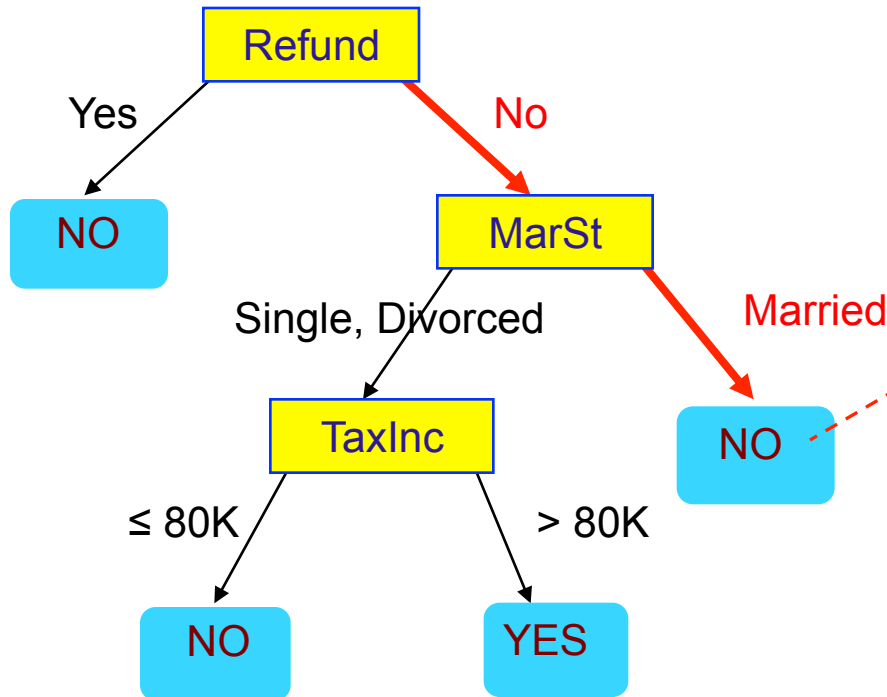
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply the Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to "No"

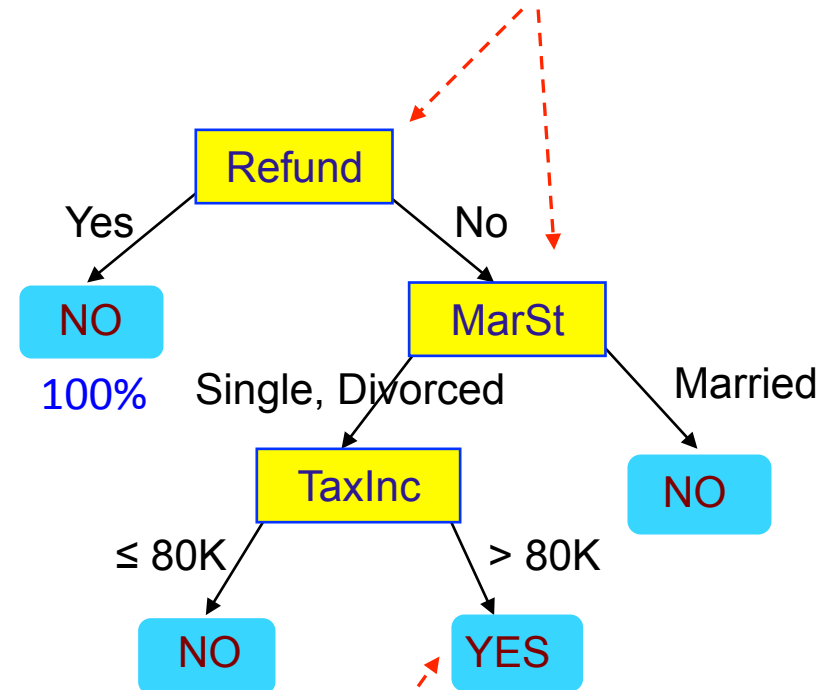
Discussion of a Decision Tree Model

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

Splitting Attributes associated with the internal nodes



Class values associated with leaves

Model: Decision Tree

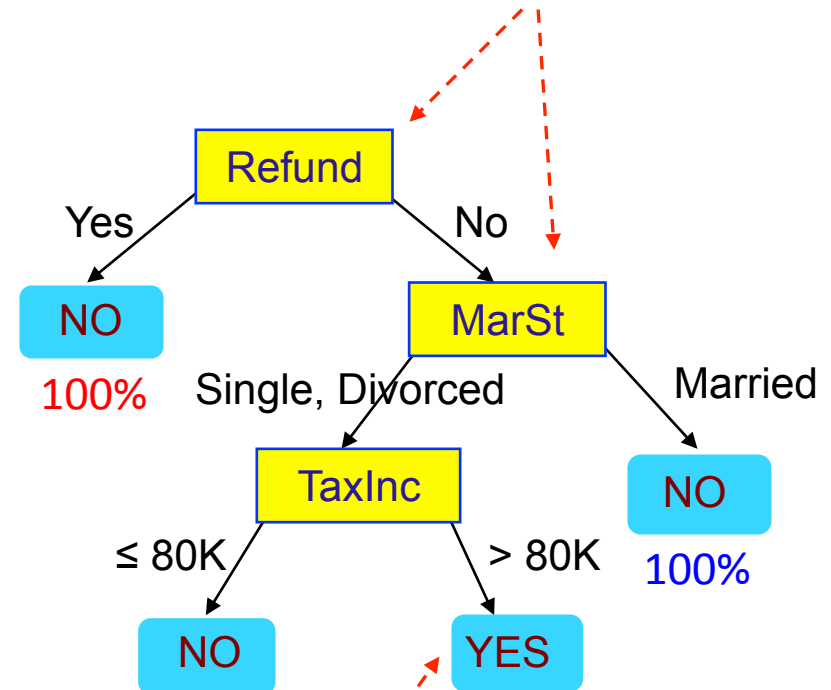
Discussion of a Decision Tree Model

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

Splitting Attributes associated with the internal nodes



Class values associated with leaves

Model: Decision Tree

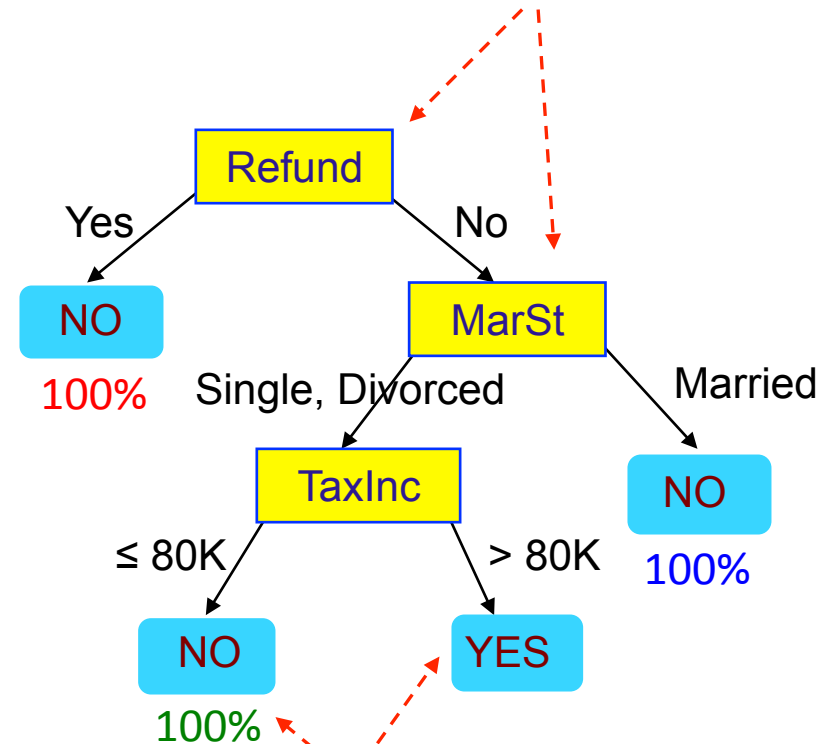
Discussion of a Decision Tree Model

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

Splitting Attributes associated with the internal nodes



Class values associated with leaves

Model: Decision Tree

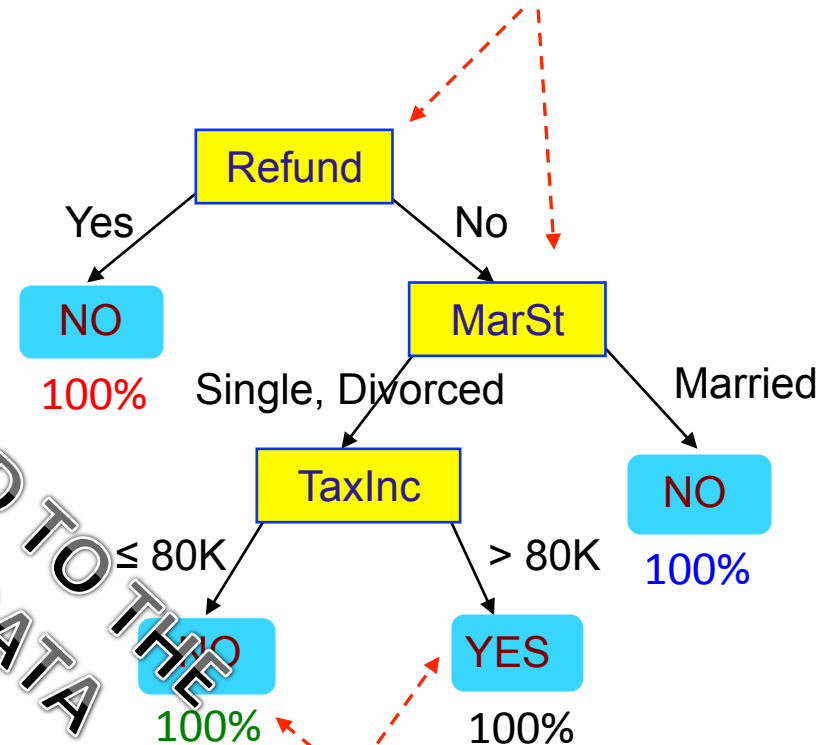
Discussion of a Decision Tree Model

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

Splitting Attributes associated with the internal nodes



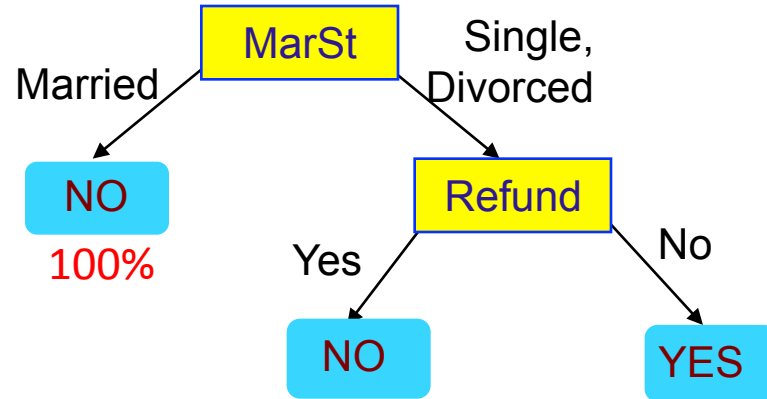
Class values associated with leaves

Model: Decision Tree

Another Example of Decision Tree

categorical
categorical
continuous
class

<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

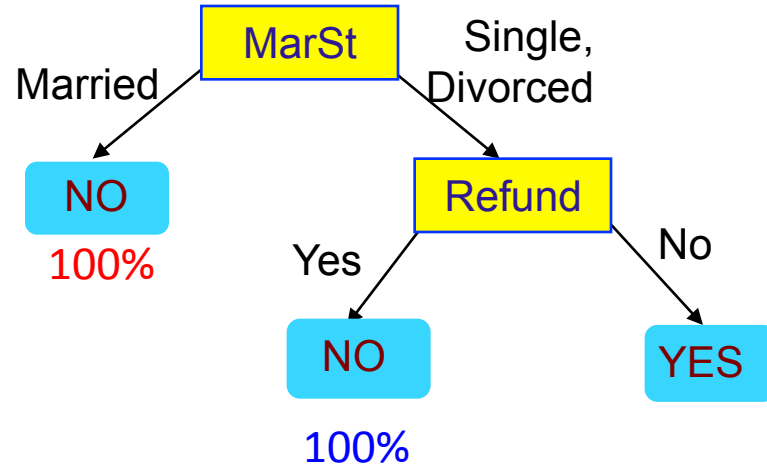


There could be more than one tree that fits the same data!

Another Example of Decision Tree

categorical
categorical
continuous
class

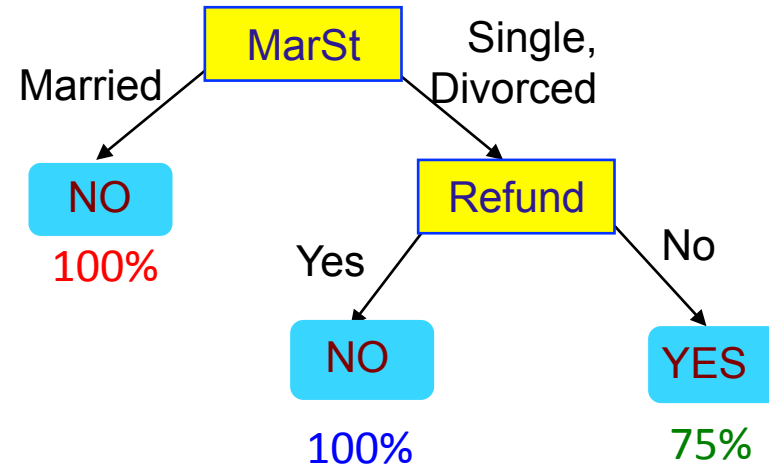
<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Another Example of Decision Tree

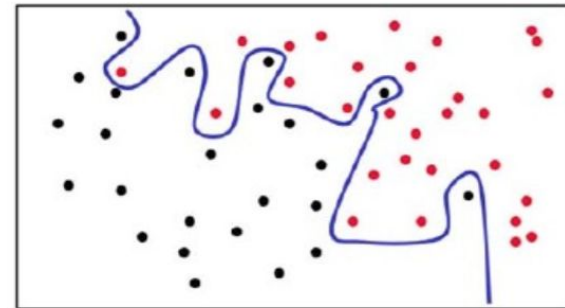
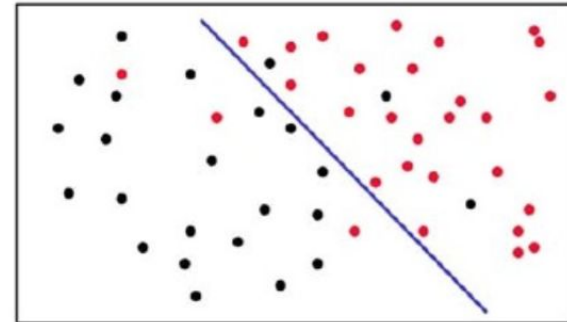
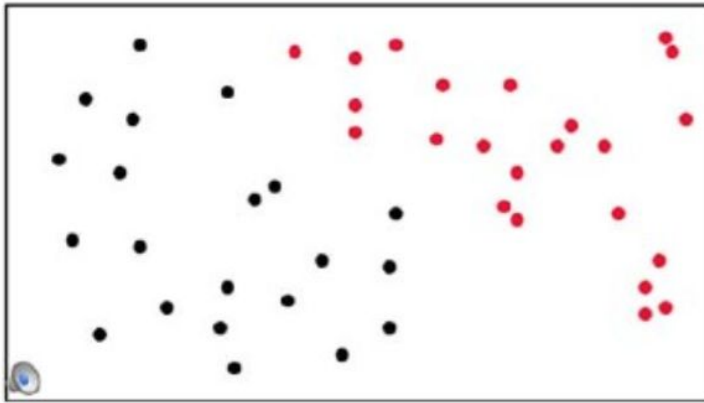
categorical *categorical* *continuous* *class*

<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



Which one is better?

A decision tree that perfectly models the training set will less likely generalize unseen data.



A good model should:

- Do a good job describing the data
- **Not be too complex!** ...to generalize the unseen data

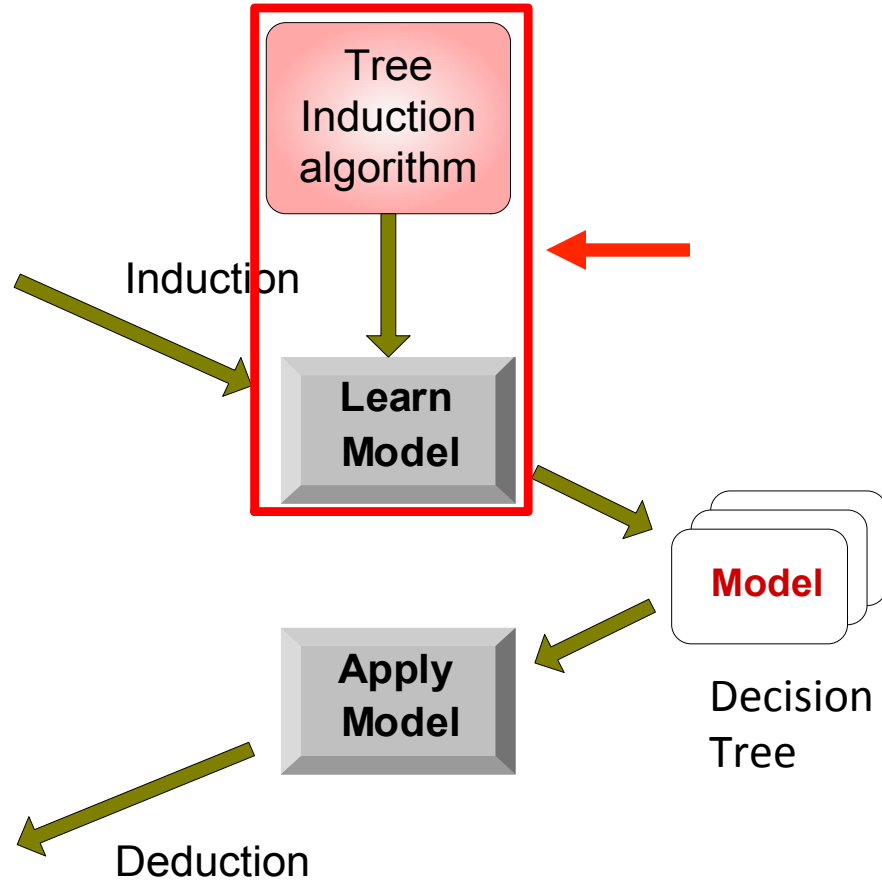
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



How do we test a classifier performance?

Once a classifier is created, we can use it to evaluate a **test set** for which we know the answers but they were not be used during the creation of the model.

General very important rule:

Never test a classifier on the same data it was used for training

In practice, we can always create a classifier that obtains a perfect classification on the training set but this will likely produce overfitting

How do we test a classifier performance?

A binary classifier can behave in one of the following ways:

1. Can **correctly** predict “**yes**” on a record with class “yes”
2. Can **correctly** predict “**no**” on a record with class “no”
3. Can **wrongly** predict “**yes**” on a record with class “no”
4. Can **wrongly** predict “**no**” on a record with class “yes”

Its overall performance on the whole test set can be summarized in the so-called confusion matrix

	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
ACTUAL CLASS	Class=No	c (FP)	d (TN)

How do we test a classifier performance?

Depending on the application, more or less importance can be given to answer correctly to “yes” and “no” classes.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

High precision means that every item labeled as “positive” does indeed belong to class “positive” (but says nothing about the number of items from class “positive” that were not labeled correctly).

What is the precision of a classifier that correctly answers “yes” just once? Is it useful?

How do we test a classifier performance?

Depending on the application, more or less importance can be given to answer correctly to “yes” and “no” classes.

Specificity = $TN / (TN + FP)$

measures the proportion of negatives that are correctly identified as such.

Similar to precision, but focusing the negative cases...

How do we test a classifier performance?

Depending on the application, more or less importance can be given to answer correctly to “yes” and “no” classes.

Sensitivity (or recall) = $TP / (TP + FN)$

High sensitivity means that **every item** from class positive was labeled as “yes” (but says nothing about how many other items were incorrectly also labeled “yes”).

What is the sensitivity of a classifier that always answers “yes”? But, in this case, what happens to the precision?

Some examples

We are developing a classifier that detects fraud in bank transactions.

Should we favor sensitivity (the ability to find most of the frauds) or precision (being absolutely sure that a detected fraud was indeed a fraud)?

Some examples

We are developing a classifier that detects fraud in bank transactions.

Should we favor sensitivity (the ability to find most of the frauds) or precision (being absolutely sure that a detected fraud was indeed a fraud)?

...it is desirable that we have a very high sensitivity, ie. most of the fraudulent transactions are identified, probably at loss of precision, since it is very important that all fraud is identified or at least suspicions are raised

Some examples

The zombie apocalypse is in progress, we want a classifier that accepts or rejects people in our safe zone.

Should we favor sensitivity (the ability to identify most of the healthy people) or precision (we should be absolutely sure that only healthy people should pass)?

Some examples

The zombie apocalypse is in progress, we want a classifier that accepts or rejects people in our safe zone.

Should we favor sensitivity (the ability to identify most of the healthy people) or precision (we should be absolutely sure that only healthy people should pass)?

Since one single mistakenly zombie in our safe zone will result in a disaster, we should favor precision over the ability to accept as many healthy people as possible...

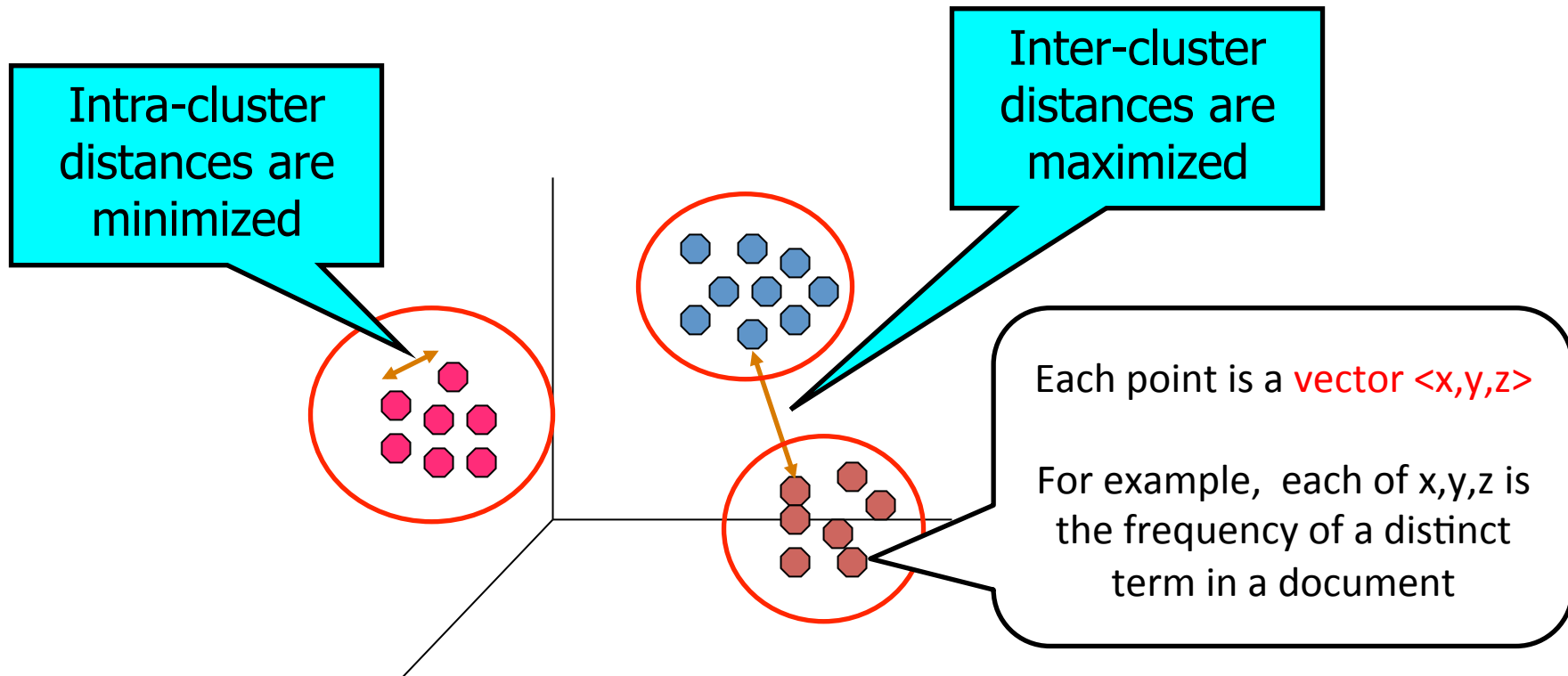
Clustering

Clustering Definition

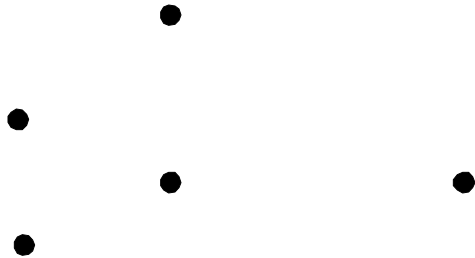
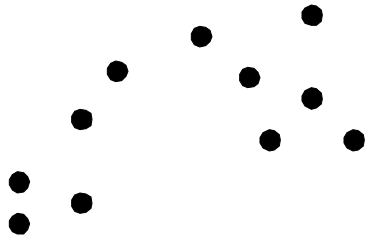
- Given a set of “data points”, each having a set of attributes, and a similarity measure among them, find clusters such that
 - **Intrac**luster: Data points in one cluster are more similar to one another.
 - **Inter**cluster: Data points in separate clusters are less similar to one another.
- Similarity / Distance Measures:
 - For vector-based object representations
 - Euclidean Distance.
 - Cosine Similarity
 - etc.
 - Other Problem-specific Measures.

What is Cluster Analysis?

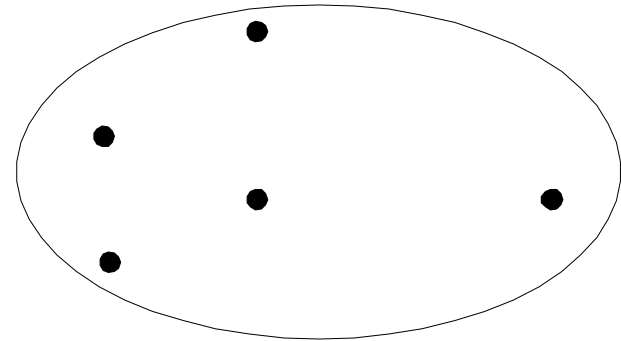
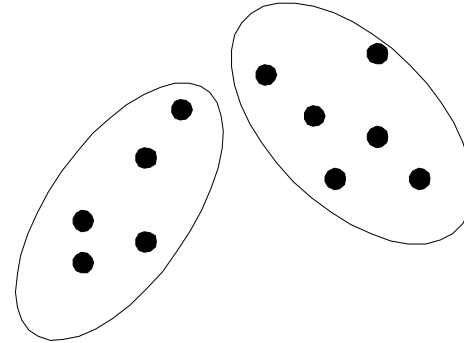
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Partitional Clustering

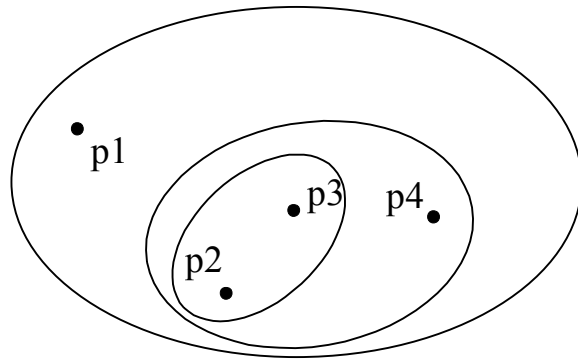


Original Points

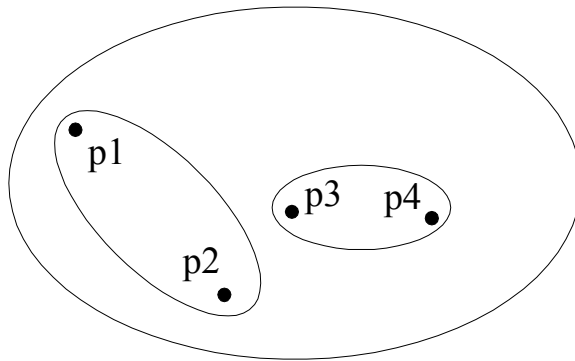
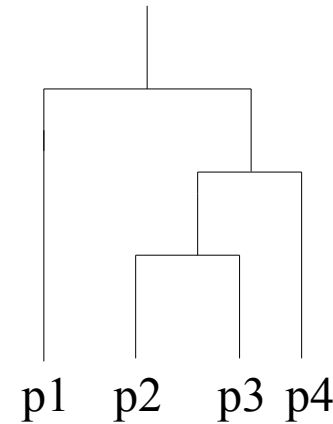


A Partitional Clustering

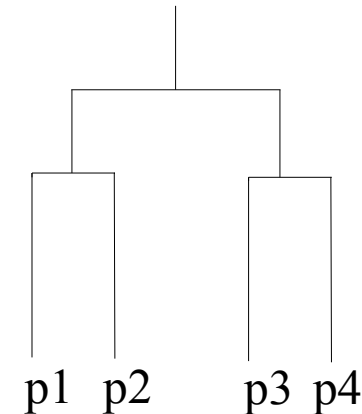
Hierarchical Clustering



Traditional Hierarchical Clustering



Non-traditional Hierarchical Clustering



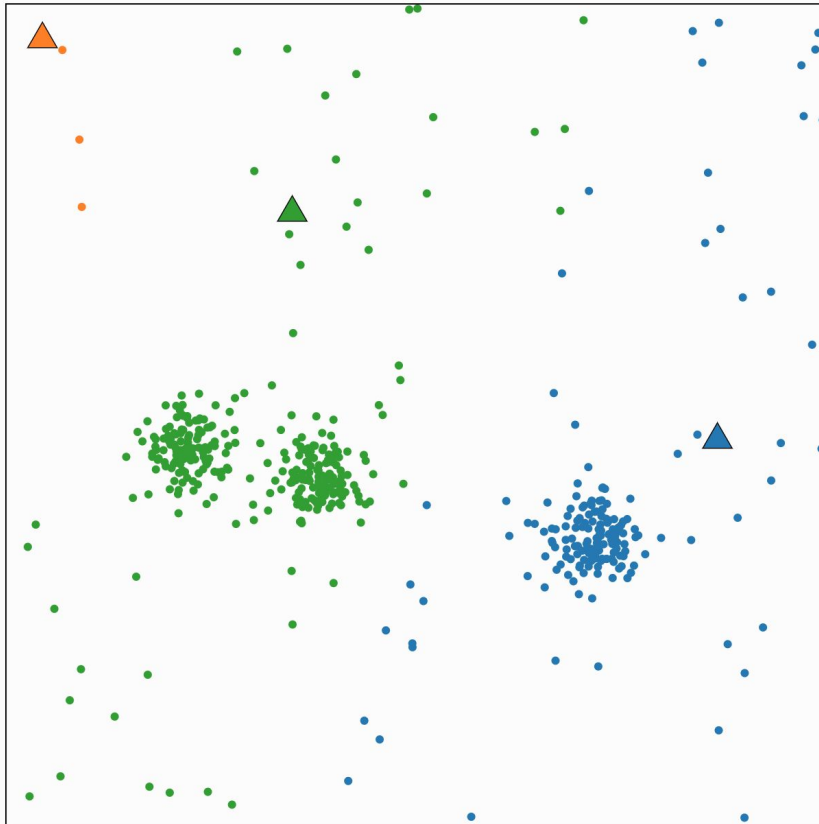
Dendrograms: Each merge is represented by a horizontal line. The y-coordinate of the horizontal line is the distance of the two clusters that were merged. The distance between merged clusters is monotone increasing with the level of the merger

K-means Clustering Algorithm

- 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 interactive demo



The k -means algorithm is an iterative method for clustering a set of N points (vectors) into k groups or clusters of points.

Algorithm

Repeat until convergence:

Find closest centroid

Find the closest centroid to each point, and group points that share the same closest centroid.

Update centroid

Update each centroid to be the mean of the points in its group.

Update centroid

Data

Clustered points ☒ Random

Number of clusters :

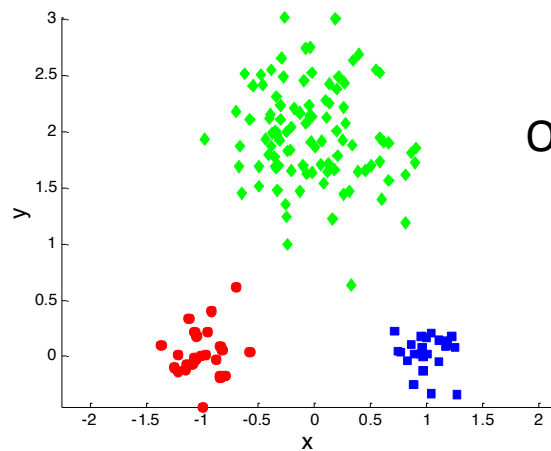
Number of centroids:

New points

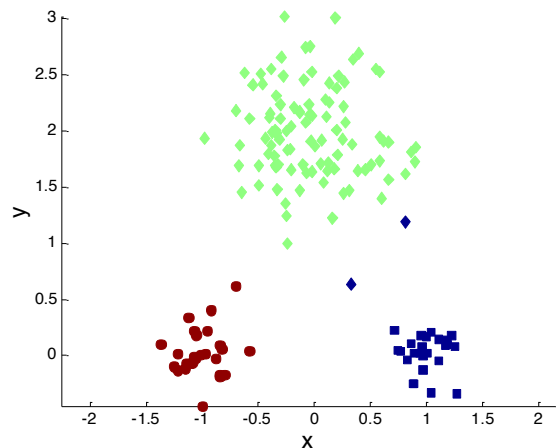
New centroids

<http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html>

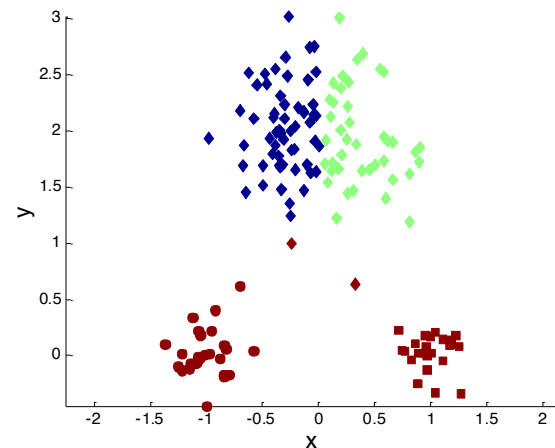
Two different K-means Clusterings



Original Points

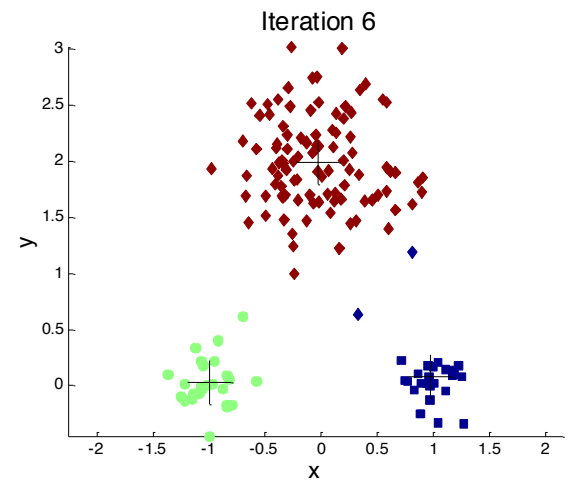
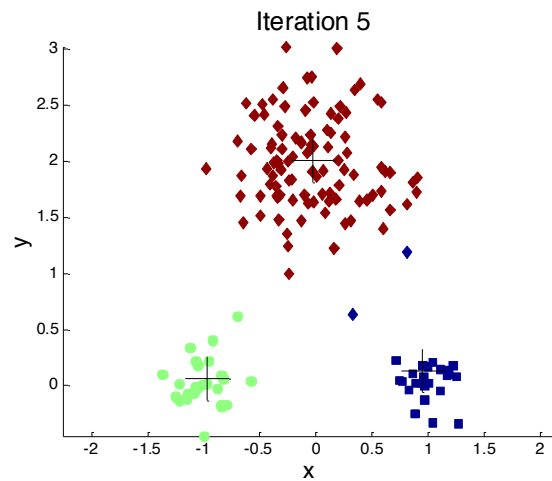
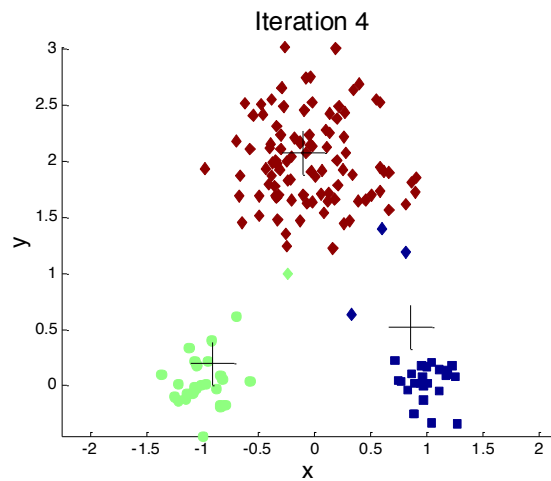
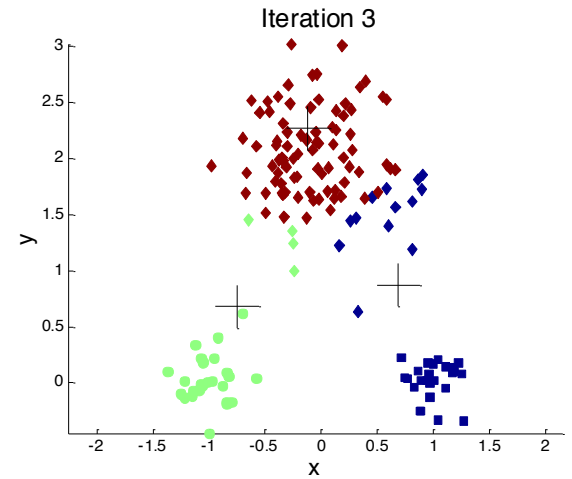
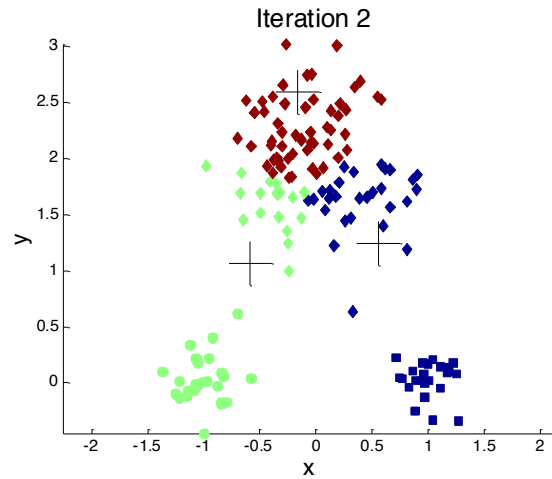
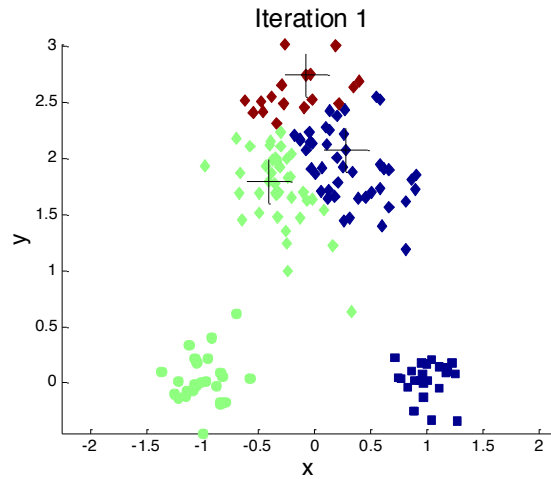


Optimal Clustering

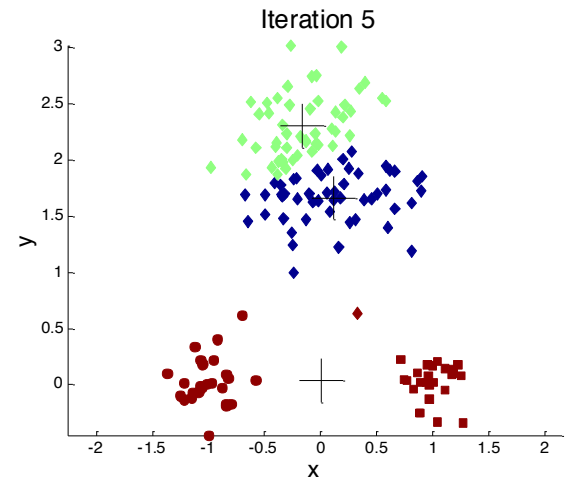
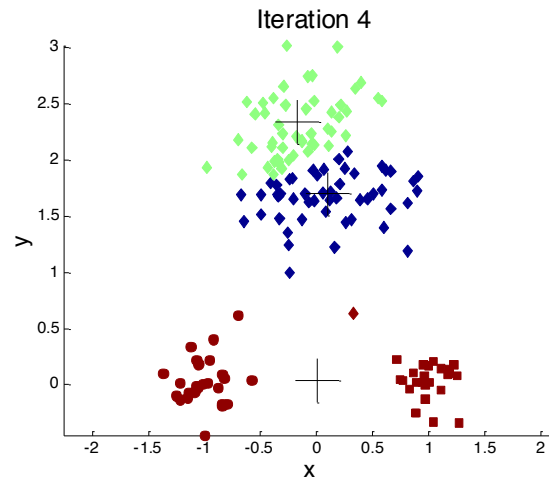
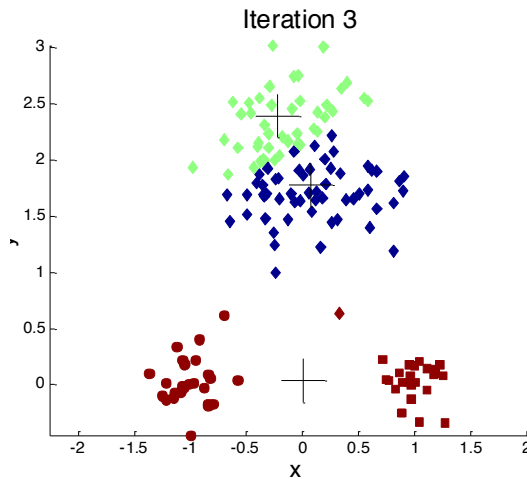
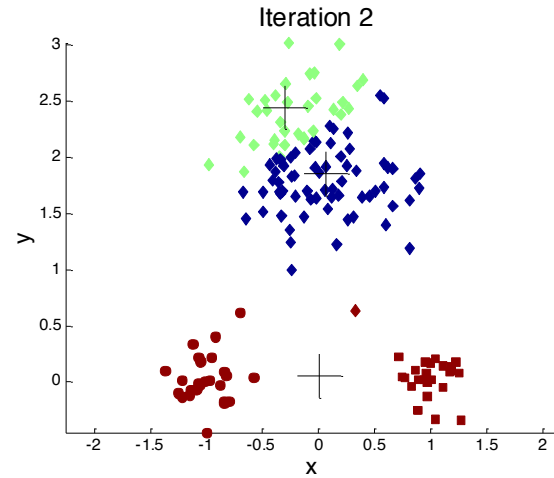
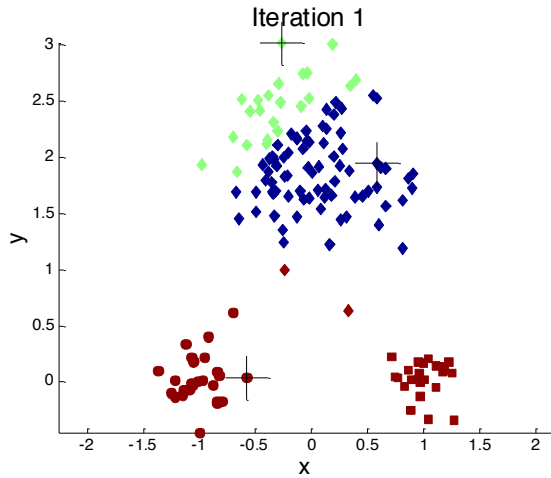


Sub-optimal Clustering

Importance of Choosing Initial Centroids

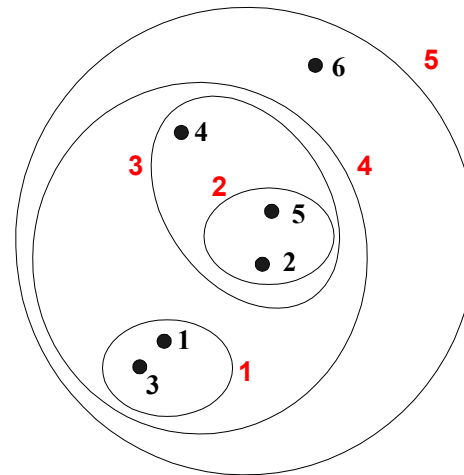
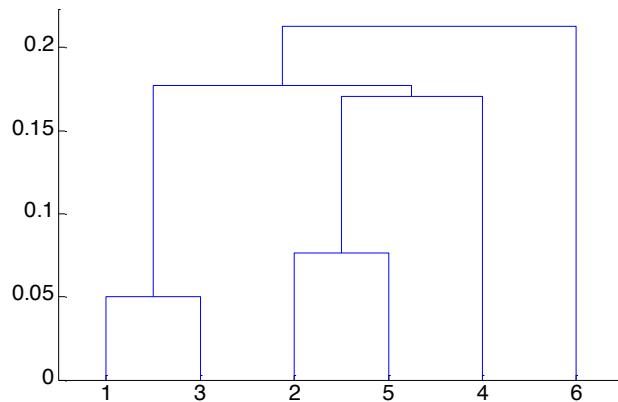


Importance of Choosing Initial Centroids ...



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, and also captures the measured distances between points/clusters



Strengths of Hierarchical Clustering

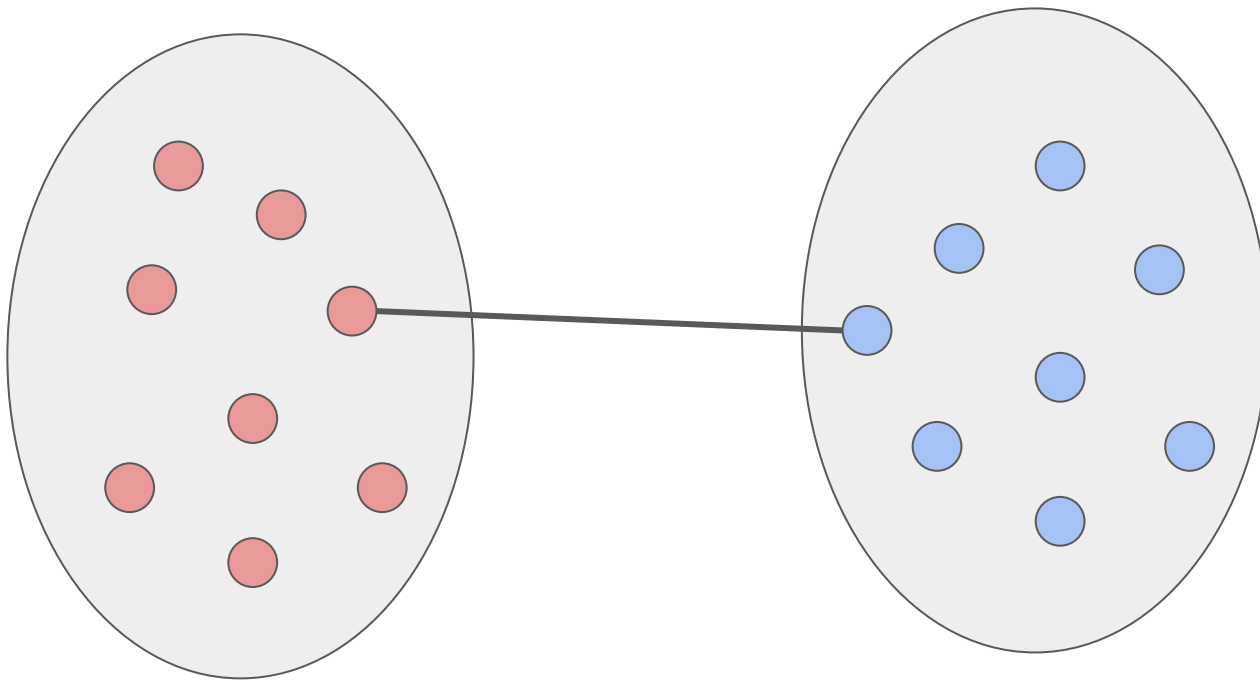
- Do not have to assume any particular number of clusters
 - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level
- They may correspond to meaningful taxonomies
 - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)

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)
 - Bisecting k -means
- Traditional hierarchical algorithms use a similarity or distance matrix
 - Merge or split one cluster at a time

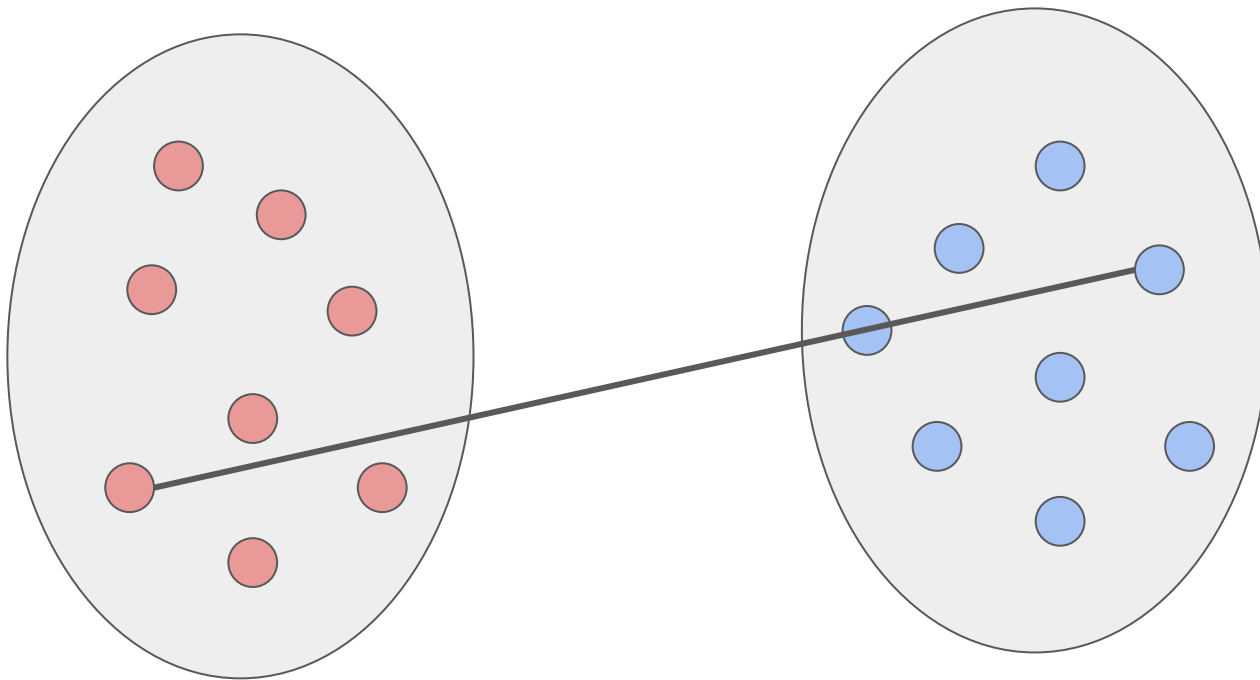
Single-linkage clustering

Distance between groups is defined as the distance between the closest pair of points from each group.



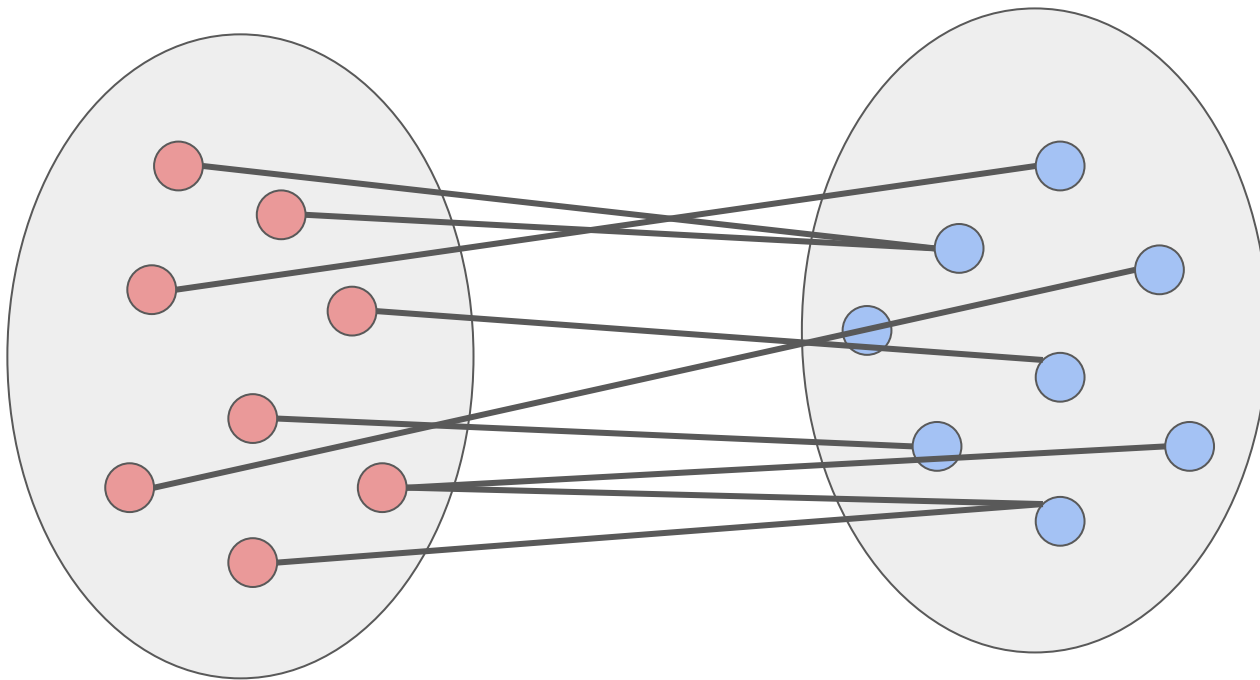
Complete-linkage clustering

Distance between groups is defined as the distance between the the most distant pair of points from each group.



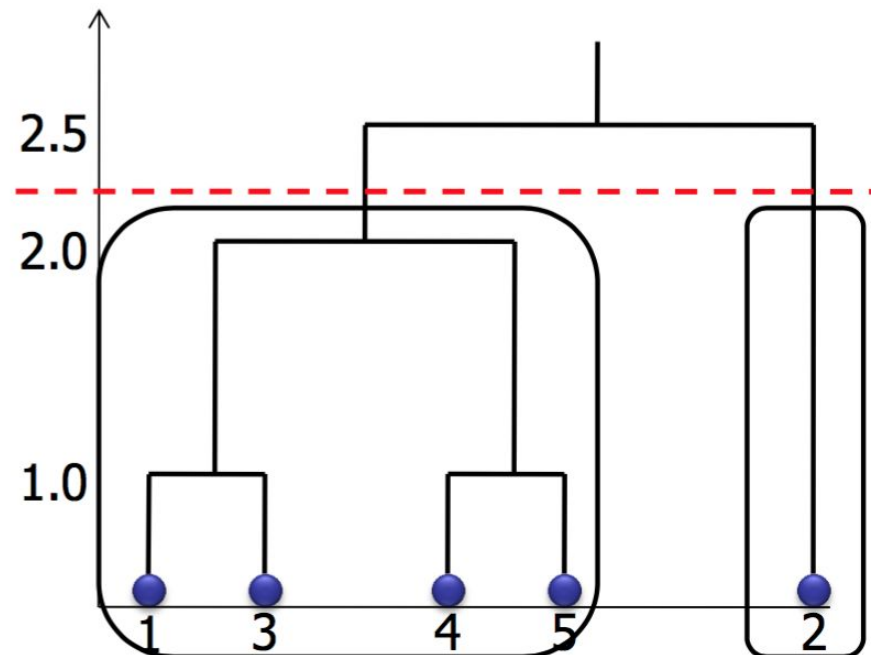
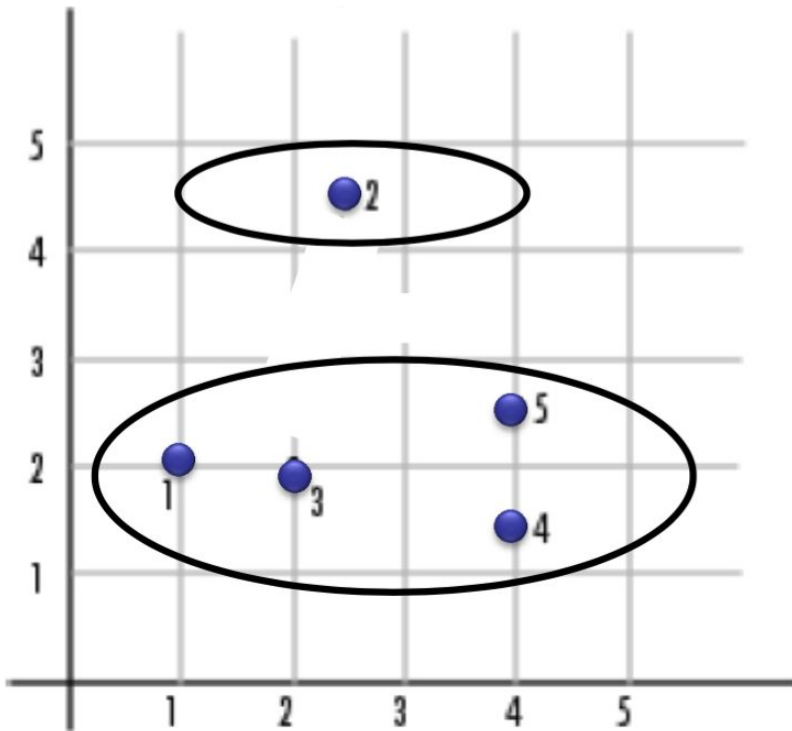
Average-linkage clustering

The distance between two clusters is defined as the average of distances between all pairs of points (of opposite clusters)



Cutting the dendrogram

When we cut the dendrogram at a specific height we generate a set of clusters. The number of clusters can be specified a-posteriori by cutting the dendrogram



Where to cut the dendrogram?

1. At arbitrary height (if we know how many cluster we want)
2. At inconsistency links, by comparing the height of each link in the dendrogram with the **heights of links below it**

If approx. equal: we have
consistent links

If heights are different: we have **inconsistent links**

