Lecture 3-4:

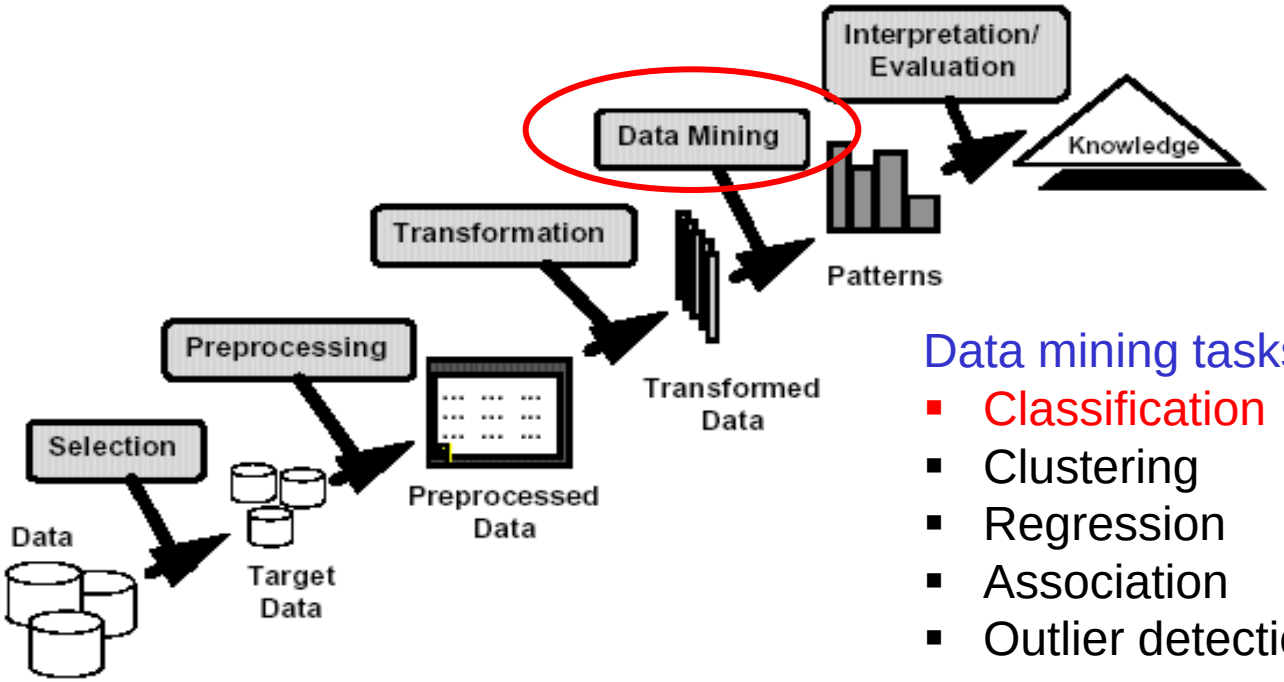
Data classification (I)

Outline

- Motivation
- Classification models – basic concepts
- Performance measures
- Classifiers
  - Simple voting (ZeroR)
  - Simple classification rules (OneR)
  - Decision trees
  - Decision rules

Motivation

Reminder: knowledge discovery workflow



Data mining tasks:

- Classification
- Clustering
- Regression
- Association
- Outlier detection

Motivation

Reminder: examples of classification problems

- Predicting tumor cells as benign or malignant (medical diagnosis)
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports, etc (document classification)
- Classifying e-mails as spam or ham (spam filtering)

Motivation

- Medical diagnosis = predict the presence/absence of an illness based on characteristics available in a health record

Example of a data subset (breast-cancer-wisconsin - arff format – see Lab 1)

```
@relation wisconsin-breast-cancer
@attribute Clump_Thickness integer [1,10]
@attribute Cell_Size_Uniformity integer [1,10]
@attribute Cell_Shape_Uniformity integer [1,10]
@attribute Marginal_Adhesion integer [1,10]
@attribute Single_Epi_Cell_Size integer [1,10]
@attribute Bare_Nuclei integer [1,10]
@attribute Bland_Chromatin integer [1,10]
@attribute Normal_Nucleoli integer [1,10]
@attribute Mitoses integer [1,10]
@attribute Class { benign, malignant}
@data
5,1,1,1,2,1,3,1,1,benign
5,4,4,5,7,10,3,2,1,benign
3,1,1,1,2,2,3,1,1,benign
8,10,10,8,7,10,9,7,1,malignant
1,1,1,1,2,10,3,1,1,benign
```

Basic concepts

What is known?

- a collection of records for which it is known to which class they belong (labeled data set)
- each record contains a set of attributes, one of the attributes is the class label

What is desired?

- a model which captures the relationship between the class attribute and the other attributes (the model is inferred using a training set through a process which is called supervised learning/ training)

Which is the final aim?

- Use the inferred model to identify the right class for a previously unseen record

Remarks:

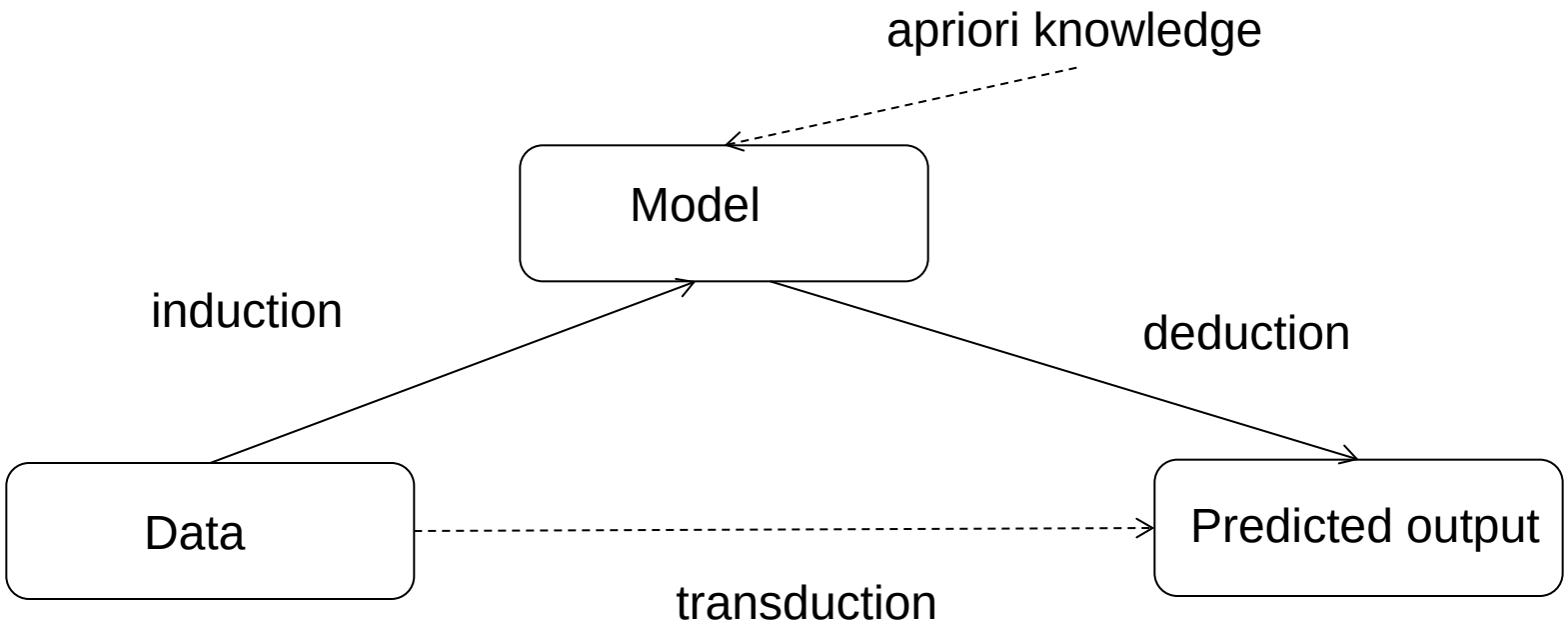
- a useful model should be accurate; the model accuracy can be estimated using the data which have not been used during the learning (test set)

Basic concepts

Learning/ induction/ inference = construct a model starting from data (and some apriori knowledge specific to the domain)

Different ways of using data, models and knowledge:

induction vs deduction vs transduction

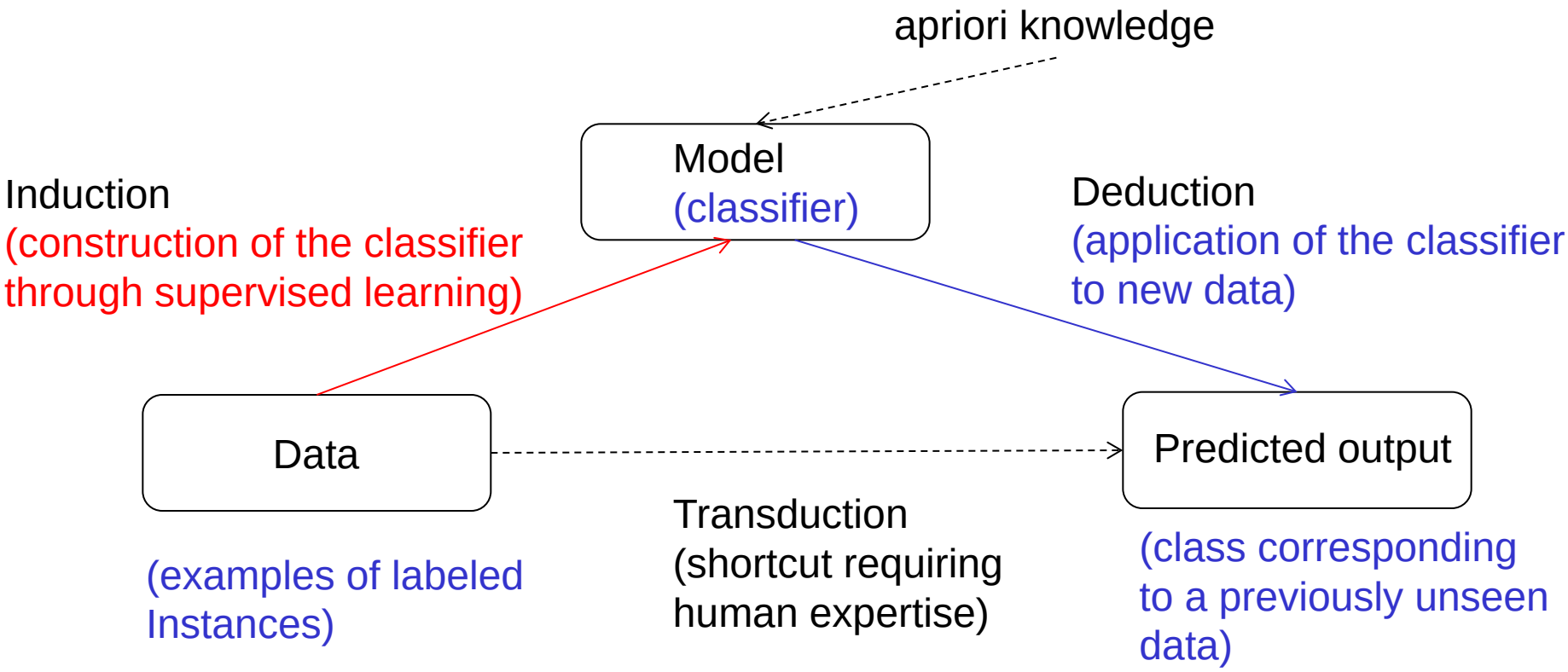


Basic concepts

Learning/ induction/ inference = construct a model starting from data (and some apriori knowledge specific to the domain)

Different ways of using data, models and knowledge:

induction vs deduction vs transduction



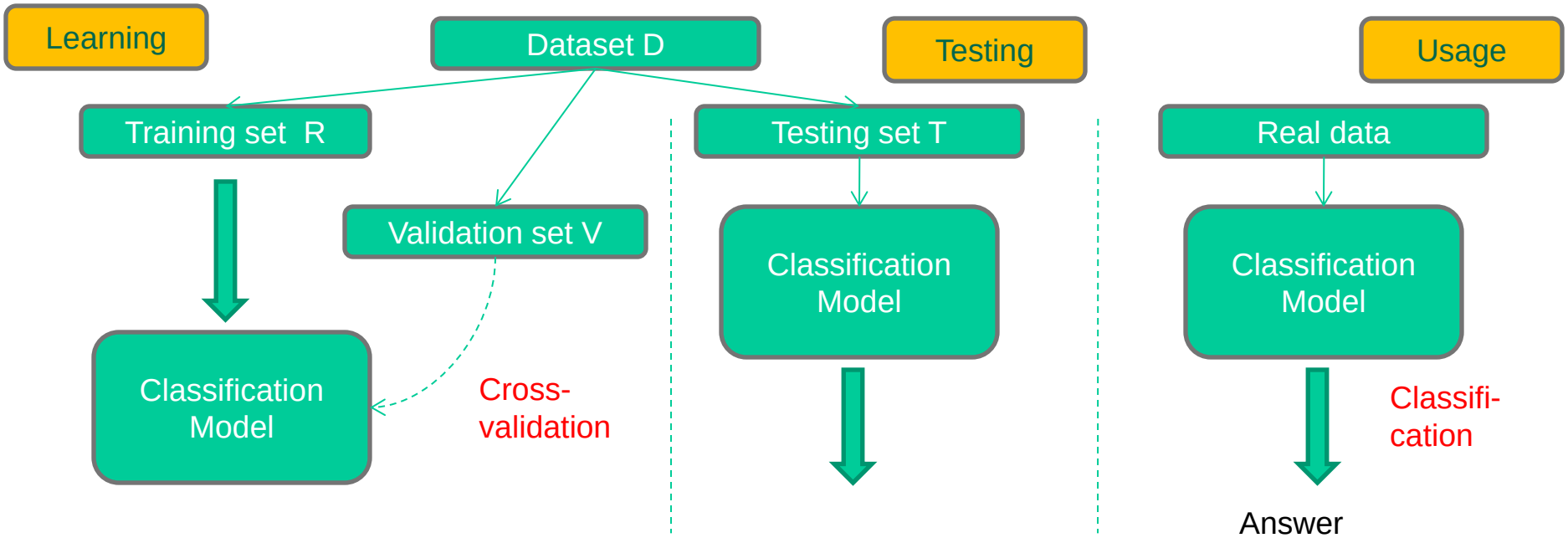
Basic concepts

Available information:

- set of labeled data:
  - $D=\{(x_1,c_1),(x_2,c_2),\ldots,(x_N,c_N)\}$
  - Each x_i has n attributes (features)
  - The class label c_i belongs to $\{1,2,\ldots,K\}$

Goal:

- Based on data from D, design a classification model C such that
  - C can predict the class to which a new data x belongs



Classification models

A classification model is a "mapping" between attributes and class labels

Example of classification models:

- Decision trees
- Classification rules
- Prototypes (exemplars)
- Probabilistic models
- Neural networks etc.

The classification model should be:

- Accurate:
  - Identify the right class
- Compact / comprehensible
  - Easy to be understood/ interpreted by the user (it is preferable to not be a black box)
- Efficient in the
  - Learning/training step
  - Classification step

Classification models

Example

@relation wisconsin-breast-cancer
@attribute Clump_Thickness integer [1,10]
@attribute Cell_Size_Uniformity integer [1,10]
@attribute Cell_Shape_Uniformity integer [1,10]
@attribute Marginal_Adhesion integer [1,10]
@attribute Single_Epi_Cell_Size integer [1,10]
@attribute Bare_Nuclei integer [1,10]
@attribute Bland_Chromatin integer [1,10]
@attribute Normal_Nucleoli integer [1,10]
@attribute Mitoses integer [1,10]
@attribute Class { benign, malignant}
@data
5,1,1,1,2,1,3,1,1,benign
5,4,4,5,7,10,3,2,1,benign
3,1,1,1,2,2,3,1,1,benign
8,10,10,8,7,10,9,7,1,malignant
1,1,1,1,2,10,3,1,1,benign
….

Simple classification rule:

IF (Cell_Size_Uniformity< 3.5)
  THEN benign
  ELSE malignant

Classification models

Example

@relation wisconsin-breast-cancer
@attribute Clump_Thickness integer [1,10]
@attribute Cell_Size_Uniformity integer [1,10]
@attribute Cell_Shape_Uniformity integer [1,10]
@attribute Marginal_Adhesion integer [1,10]
@attribute Single_Epi_Cell_Size integer [1,10]
@attribute Bare_Nuclei integer [1,10]
@attribute Bland_Chromatin integer [1,10]
@attribute Normal_Nucleoli integer [1,10]
@attribute Mitoses integer [1,10]
@attribute Class { benign, malignant}
@data
5,1,1,1,2,1,3,1,1,benign
5,4,4,5,7,10,3,2,1,benign
3,1,1,1,2,2,3,1,1,benign
8,10,10,8,7,10,9,7,1,malignant
1,1,1,1,2,10,3,1,1,benign
….

Simple classification rule:

IF (Cell_Size_Uniformity< 3.5)
  THEN benign
  ELSE malignant

Question: How good is this rule?

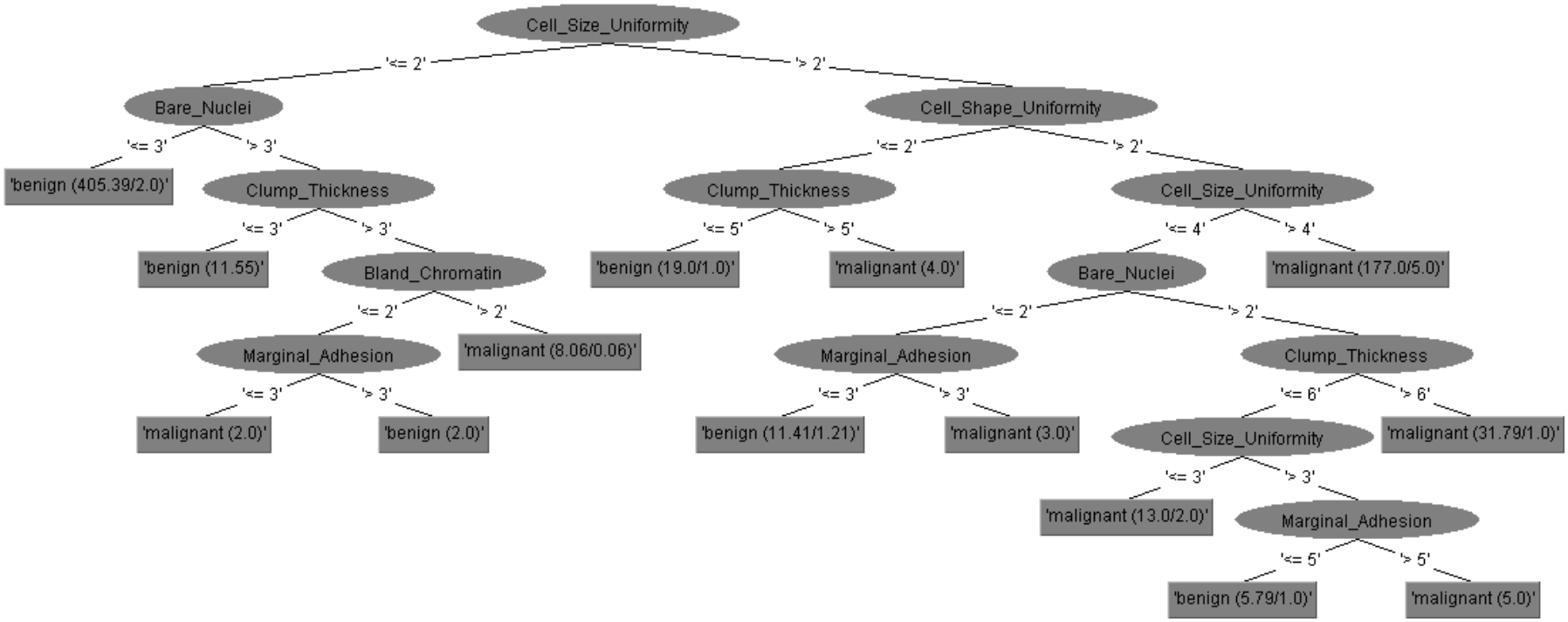
In 92.7% of cases the classifier produced the right class

How was this value computed?

How should it be interpreted?

Classification models

Example: a more complex model (decision tree)



Performance: in 94.56% of cases the classifier produced the right class

What about its readability?

Performance measures

Context: let us consider a 2-classes problem

- Class 1 – positive (e.g. malignant)
- Class 2 – negative (e.g. benign)

The simplest way to measure the performance is to analyze in how much cases the classifier provides the right answer – this information can be summarized in the so-called confusion matrix

Confusion matrix:

	C1	C2
C1	TP	FN
C2	FP	TN

← provided by the classifier

↑ true class

TP = True Positive = nr of cases which are in C1 and are correctly classified in C1

TN = True Negative = nr of cases which are in C2 and are correctly classified in C2

FP = False Positive = nr of cases which are in C2 but are wrongly classified in C1

FN = False Negative = nr of cases which are in C1 but are wrongly classified in C2

Performance measures

The case of K classes:

- One can construct a confusion matrix for each class (the current class is considered the positive one and all the other classes are grouped into an aggregated negative class)
- One can extend the confusion matrix to K classes by considering K rows and K columns

Confusion matrix:

	C_1	C_2	...	C_j	…	C_K
C_1	T_1	F_{12}	…	F_{1j}	…	F_{1K}
C_2	F_{21}	T_2	…	F_{2j}	…	F_{2K}
…		…	…	…		
C_i	F_{i1}	F_{i2}		F_{ij}		F_{iK}
…		…	…	…		
C_K	F_{K1}	F_{K2}		F_{1Kj}		T_K

↑ true class

← provided by the classifier

F_{ij} = nr of cases which should be classified in C_i but are classified in C_{1j}

Performance measures

TP = True Positive = nr of cases which are in C1 and are correctly classified in C1
TN = True Negative = nr of cases which are in C2 and are correctly classified in C2
FP = False Positive = nr of cases which are in C2 but are wrongly classified in C1
FN = False Negative = nr of cases which are in C1 but are wrongly classified in C2

Accuracy = (TP+TN)/(TP+TN+FP+FN) = ratio of correctly classified data

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

Specificity = TN/ (TN+FP) (TN rate), 1-specificity=FP/(TN+FP) = FP rate

Precision = TP/(TP+FP)

Remarks:

- In all cases the value is in [0,1]; higher value suggest better performance
- Sensitivity and specificity are frequently used in medical data analysis while precision and recall are used in information retrieval

Performance measures

TP = True Positive = nr of cases which are in C1 and are correctly classified in C1
TN = True Negative = nr of cases which are in C2 and are correctly classified in C2
FP = False Positive = nr of cases which are in C2 but are wrongly classified in C1
FN = False Negative = nr of cases which are in C1 but are wrongly classified in C2

In the context of information retrieval:

Precision = TP/(TP+FP) = card(relevant and retrieved)/ card(retrieved)
Recall = TP/(TP+FN) = card(relevant and retrieved)/ card(relevant)

Both of them should be maximized; an aggregated variant is their harmonic mean

F-measure=2*precision*recall/(precision+recall)

Performance measures

TP = True Positive = nr of cases which are in C1 and are correctly classified in C1
TN = True Negative = nr of cases which are in C2 and are correctly classified in C2
FP = False Positive = nr of cases which are in C2 but are wrongly classified in C1
FN = False Negative = nr of cases which are in C1 but are wrongly classified in C2

Cost sensitive accuracy

- In some cases (e.g. medical diagnosis) misclassifying the data belonging to one class may be more costly than misclassifying data belonging to the other class (e.g. missing a malignant case is more costly than missing a normal case) – in such cases the accuracy should take into account the cost

- $CostAccuracy=(cost_1*n_1*sensitivity+cost_2*n_2*specificity)/(cost_1*n_1+cost_2*n_2)$
  - $cost_i$ = cost of misclassifying data from C_i
  - n_i = numer of data in C_i

The simplest classifier

Example:

- let us consider the dataset “sick” from UCI Machine Learning
- It contains records for 3772 patients, out of which:
  - 231 are sick (class C1 – positive)
  - 3541 are not sick (class C2 – negative)
- We are interested in a classifier having an accuracy at least equal to 0.9 (90%) for the available dataset
- Which is the simplest classifier which satisfies this requirement?

The simplest classifier

Example:

- let us consider the dataset “sick” from UCI Machine Learning
- It contains records with 29 attributes for 3772 patients, out of which:
  - 231 are sick (class C1 – positive)
  - 3541 are not sick (class C2 – negative)
- We are interested in a classifier having an accuracy at least equal to 0.9 (90%) for the available dataset
- Which is the simplest classifier which satisfy this requirement?
- By considering the rule: “disregarding the values of the attributes the class is C2 (negative)” we obtain that accuracy=3541/3772=0.94>0.9
- Is this an appropriate classifier? Is such a classifier of any use?

The simplest classifier

- Is this an appropriate classifier? Is such a classifier of any use?

- This is the so-called ZeroR classifier which uses only the distribution of data over classes and always returns the label of the most frequent class in the data set (it relies on a simple voting mechanism)

- It is not appropriate as it will produce a wrong answer for all data in the less frequent classes

- However it can be used to establish a lower bound for the accuracy of a classifier: if another classifier behaves worse than ZeroR then it probably should not be taken into consideration

Back to performance evaluation

- Using the entire set of available data to construct the classifier is not a wise approach, as the classifier could overfit the training data:
  - It produces good results for data used in the learning process
  - It behaves poorly for data not used in the learning process

- A better approach is to split the available dataset in:
  - a training subset (used to construct the classifier)
  - a testing subset (used to estimate the performance)

- There are different splitting strategies and variants for sampling data to be included in the training and testing subsets

Remark:

Besides testing subset, another subset (validation subset) could be used in order to tune the parameter of the classifier

Back to performance evaluation

Splitting strategies:

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout (the performance is computed by averaging the results obtained for each repeat)
- Cross validation
  - Split the dataset randomly into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling (useful in the case of unbalanced datasets)
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

Back to performance evaluation

Cross validation

- Partition data into k disjoint subsets
- k-fold: train on k-1 partitions, test on the remaining one

The overall performance is computed as the average of the performance measures computed in the k folds

Remarks:

- It is one of the most frequently used approach

One step beyond ZeroR

Dataset: sick.arff, 29 attributes, 3772 instances (231 in class C1, 3541 in class C2), 2 classes

ZeroR (always classify into the C2): accuracy=0.94

OneR: construct classification rules which contains only one attribute in the left hand side

Inferred rules (via Weka OneR):

If T3< 0.25 then C2 (negative)

If T3 in [0.25, 0.35) then C1 (sick)

If T3 in [0.35, 0.55) then C2 (negative)

If T3 in [0.55, 1.15) then C1 (sick)

If T3 >= 1.15 then C2 (negative)

If T3 value is missing then C2 (negative) Accuracy: 0.96

OneR

Main idea: find the attribute with the highest discriminative power and use it to define classification rules

Remark: it is appropriate for attributes which take discrete values

Algorithm:

FOR each attribute A_i do

FOR each value v_{ij} of A_i construct

R_{ij} : if $A_i = v_{ij}$ then class $C_{k(i,j)}$

(the majority class amongst the subset of instances having $A_i = v_{ij}$)

aggregate in a set of rules R_i corresponding to A_i and compute Err_i

(number of wrongly classified data)

ENDFOR

ENDFOR

Select the set of rules characterized by the smallest error

OneR

Example: weather/play dataset

Outlook: err=4

sunny: 2 yes/ 3 no (→ no)

overcast: 4 yes/ 0 no (→ yes)

rainy: 3 yes/2 no (→yes)

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

OneR

Example: weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Outlook: err=4

sunny: 2 yes/ 3 no (→ no)

overcast: 4 yes/ 0 no (→ yes)

rainy: 3 yes/2 no (→yes)

Temperature: err=5

hot: 2 yes/2 no (→ yes)

mild: 4 yes/2 no (→ yes)

cool: 3 yes/ 1no (→ yes)

OneR

Example: weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Outlook: err=4

sunny: 2 yes/ 3 no (→ no)

overcast: 4 yes/ 0 no (→ yes)

rainy: 3 yes/2 no (→yes)

Temperature: err=5

hot: 2 yes/2 no (→ yes)

mild: 4 yes/2 no (→ yes)

cool: 3 yes/ 1no (→ yes)

Humidity: err=5

high: 4 yes/ 4 no (→ yes)

normal: 6 yes/ 1 no (→ yes)

OneR

Example: weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Outlook: err=4

sunny: 2 yes/ 3 no (→ no)

overcast: 4 yes/ 0 no (→ yes)

rainy: 3 yes/2 no (→yes)

Temperature: err=5

hot: 2 yes/2 no (→ yes)

mild: 4 yes/2 no (→ yes)

cool: 3 yes/ 1no (→ yes)

Humidity: err=5

high: 4 yes/ 4 no (→ yes)

normal: 6 yes/ 1 no (→ yes)

Windy: err=5

true: 3 yes/3 no (→ yes)

false: 6 yes/ 2 no (→ yes)

Rules: weather/play dataset

If outlook=sunny then "no"

If outlook=overcast then "yes"

If outlook=rainy then "yes"

Accuracy (training set): 0.71

Accuracy (cross-validation): 0.43 (!!)

OneR

Example: weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Rules: weather/play dataset

If outlook=sunny then "no"

If outlook=overcast then "yes"

If outlook=rainy then "yes"

Classification step:

- New day: (outlook=rainy, temperature=cool, humidity=high, windy=false)
- Answer: Yes

OneR

Summary of OneR implementation

- Construction of the set of rules (training step)
  - Input: training set (labelled data instances)
  - Output: set of simple rules (all rules involve only one attribute – the same attribute in all rules)
  - Algorithm: search all attributes and corresponding values and select the attribute leading to the smallest classification error (see slide 26)

- Using the set of rules (classification step)
  - Input: set of rules, new data instance
  - Output: class label
  - Algorithm:
    - Find the rule which match the input data
    - Return the class corresponding to the matching rule

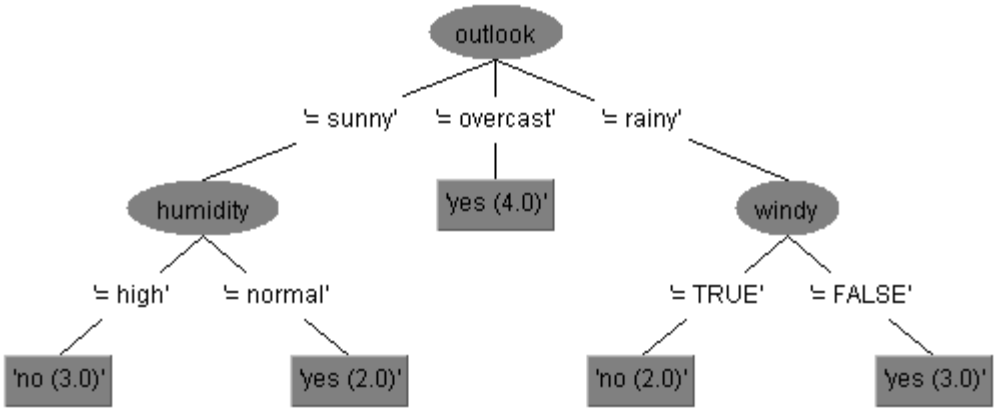
One step beyond OneR

Weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Decision tree (constructed using Weka)



How can be used?

Which class corresponds to a new instance:
(outlook=sunny, temperature=mild, humidity=normal, windy=False)?

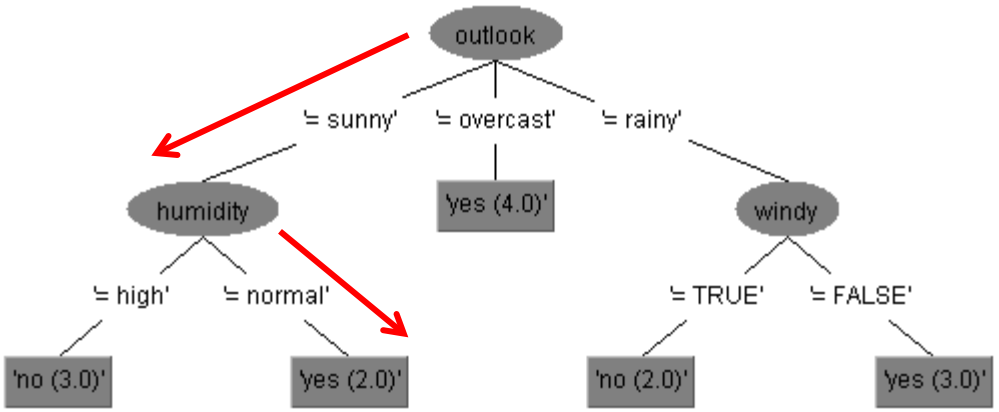
Decision Trees

Weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Decision tree (constructed using Weka)



How can be used?

Which class corresponds to a new instance
(outlook=sunny, temperature=mild, humidity=normal, windy=False)?

Class: Yes

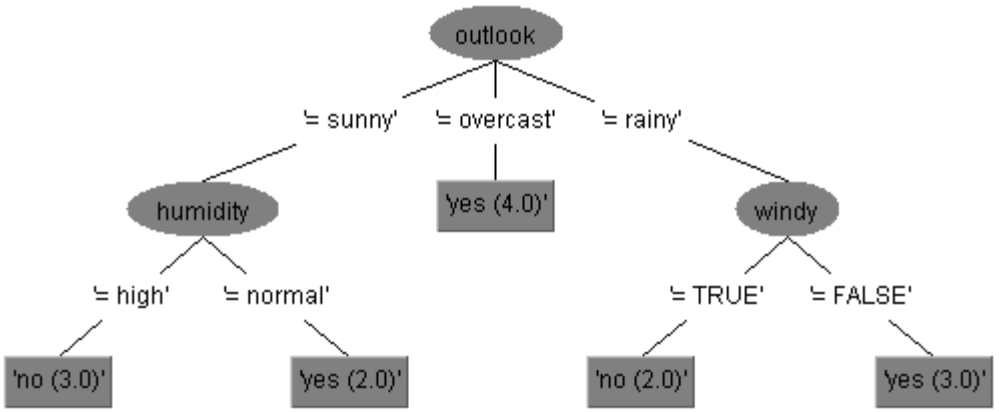
Decision Trees

Weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Decision tree (constructed using Weka)



How can be translated in a set of classification rules? Each branch leads to a rule

Rule 1: IF outlook=sunny and humidity=high THEN play=no
Rule 2: IF outlook=sunny and humidity=normal THEN play=yes
Rule 3: IF outlook=overcast THEN play=yes
Rule 4: IF outlook=rainy and windy=True THEN play=no
Rule 5: IF outlook=rainy and windy=False THEN play=yes

Decision Trees

How can a decision tree be constructed (inferred) from a dataset?

- Choose an attribute and put it as the root of the tree
- For each possible value of the attribute (present in the dataset) construct a branch (split the node)
- Split the dataset in subsets corresponding to each branch
  - If a subset contains data from only one class then it will correspond to a leaf node (no more splitting on that branch) – pure node
  - If in a subset there are data belonging to different classes then the splitting process is continued until
    - is arrived to a pure node
    - all attributes have been analyzed on that branch
    - the subset of data corresponding to that branch is empty

Weather/play dataset

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Problem: in which order should be analyzed the attributes? which test condition should be assigned to a node?

Decision Trees

In which order should be analyzed the attributes?

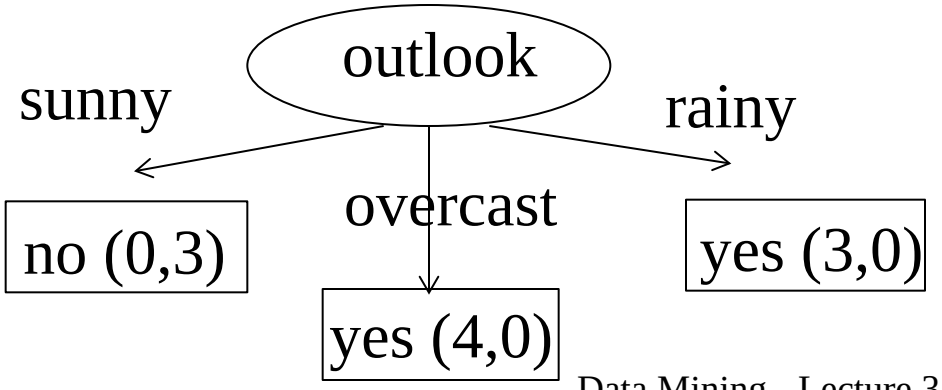
Main idea:

- Select the attribute which leads to a simple tree, i.e. an attribute with a high purity level (ideally, for each possible value of the attribute the corresponding data instances belong to the same class)

Weather/play dataset (selected instances)

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	overcast	cool	normal	TRUE	yes
7	sunny	mild	high	FALSE	no
8	rainy	mild	normal	FALSE	yes
9	overcast	mild	high	TRUE	yes
10	overcast	hot	normal	FALSE	yes

Example:



Remark:

- All leaves are “pure” (contain data belonging to the same class)
- Such a flat tree leads to classification rules involving only one attribute in the left-hand side
- This situation happens rarely for real-world data

Decision trees

The main decisions to be taken during the decision tree induction

- Which are the test conditions to be assigned to the branches corresponding to a node?
  - It depends on the attributes type
    - Nominal, ordinal, continuous
  - It depends on the desired degree of the splitting node:
    - 2-way split (the current subset is divided in 2 subsets)
    - Multi-way split (the current subset is divided in several subsets)

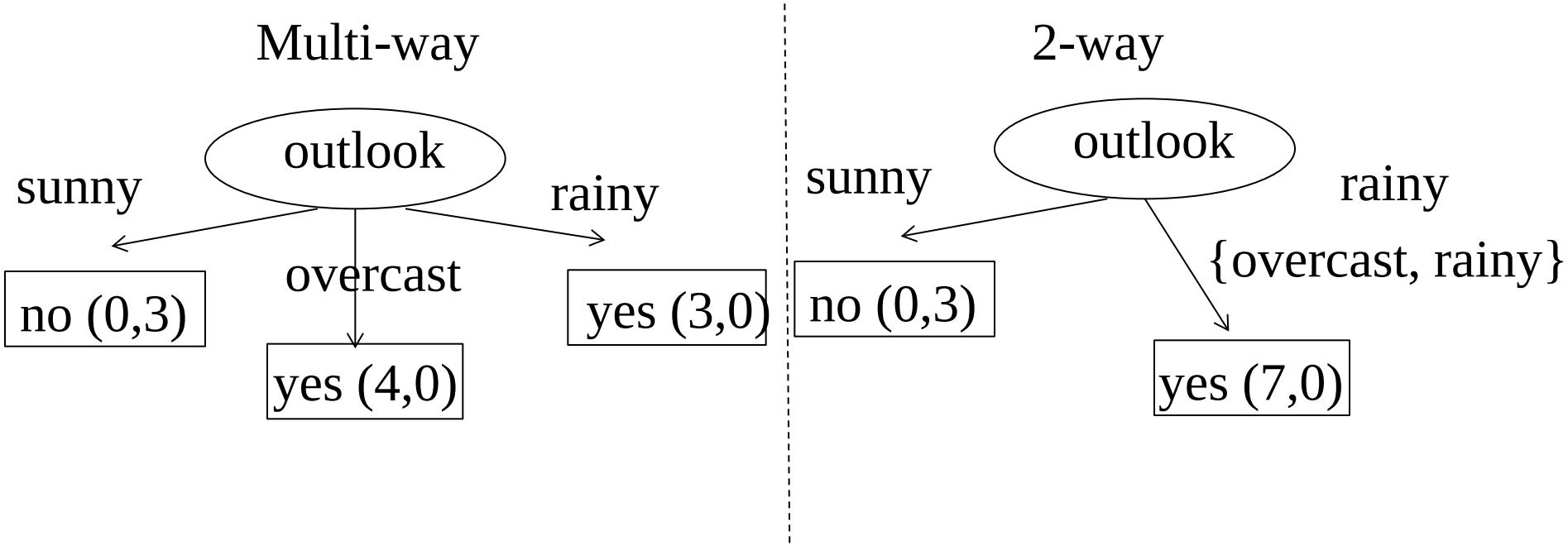
- Which attribute should be selected for splitting?
  - The most discriminative one – that which ensures a partition of the current dataset in subsets with a high degree of purity
  - There are several criteria which can be used:
    - Entropy (variants: information gain, gain ratio)
    - Gini index
    - Misclassification

Decision trees

- Which are the test conditions to be assigned to the branches corresponding to a node?

Nominal and ordinal attributes:

- Multi-way: as many branches as possible values
- 2-way: two branches



Decision trees

Which are the test conditions to be assigned to the branches corresponding to a node?

Numerical attributes:

- The numerical attributes are previously discretized and then is applied the approach which is specific to attributes with discrete values (using the mult-way or 2-way approaches)

Which attribute should be selected for splitting?

- That which leads to the maximal reduction in the information needed to take the right decision
  - Information gain = Entropy(distribution of data before splitting) – AveragedEntropy(distribution of data after splitting)

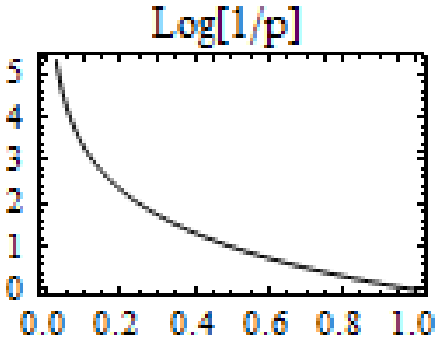
Reminder: entropy

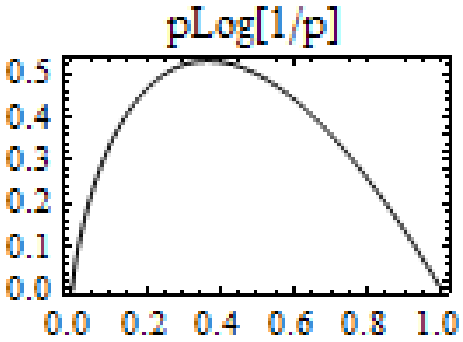
Let D=(p_1,p_2, …, p_k) be a distribution probability. The entropy associated to this distribution probability is defined as

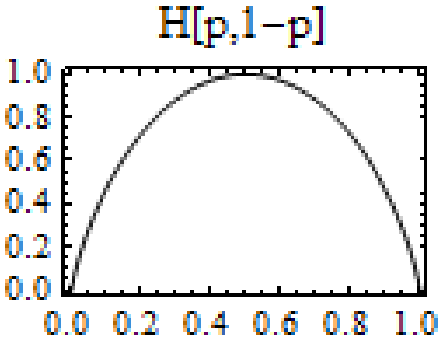
$$H(D) = H(p_1, p_2, ..., p_k) = -\sum_{i=1}^{k} p_i \log p_i$$

And can be interpreted as a measure of the amount of uncertainty (or surprise) when we sample data according to the distribution probability

Particular case: k=2 => p_1=p, p_2=1-p







Remark:

Log[1/p] interpretation: the surprise of seeing an event characterized by a small probability (unexpected event) is higher than in the case of an event of high probability (which is expected)

Reminder: entropy

In the context of a classification problem:

- $D=\{C_1,C_2,\ldots,C_k\}$ (dataset of instances belonging to k classes)
- Distribution probability $(p_1,p_2, \ldots, p_k)$, $p_i=card(C_i)/card(D)$
- Let A be an attribute and $v_1,v_2,\ldots,v_{mA}$ the set of values taken by this attribute
- Let D_j=set of instances from D for which attribute A has the value v_j and P_j the distribution of data of D_j in the k classes (C_{ji}=number of instances having the value v_j for attribute A which belong to class C_i)
- Information Gain obtained by splitting the dataset according to attribute A

$$IG(D,A) = H(D) - \sum_{j=1}^{m_A} P(D_j \mid A = v_j) H(D_j \mid A = v_j), \quad H(D) = -\sum_{i=1}^{k} p_i \log p_i$$

$$H(D_j \mid A = v_j) = -\sum_{i=1}^{k} p_{ij} \log p_{ij}, \quad p_{ij} = \frac{card(C_{ji})}{card(C_i)}$$

$$P(D_j \mid A = v_j) = \frac{card(D_j)}{card(D)}$$

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

- Class distribution (C_1="yes", C_2="no"):
- p_1=9/14, p_2=5/14
- $H(p_1,p_2)$=0.94

Outlook

	C1 (yes)	C2(no)	Frequency
Sunny	2/5	3/5	5/14
Overcast	4/4	0/4	4/14
Rainy	3/5	2/5	5/14

H(sunny)=-2/5*log(2/5)-3/5*log(3/5)=0.97
H(overcast)=-1*log(1)-0=0
H(rainy)=-3/5*log(3/5)-2/5*log(2/5)=0.97

IG(outlook)=0.94-5/14*0.97-4/14*0-5/14*0.97=0.94-0.69=0.25

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

- Class distribution (C_1="yes", C_2="no"):
- p_1=9/14, p_2=5/14
- $H(p_1,p_2)$=0.94

Temperature

	C1 (yes)	C2(no)	Frequency
Hot	2/4	2/4	4/14
Mild	4/6	2/6	6/14
Cool	3/4	1/4	4/14

H(hot)=-2/4*log(2/4)-2/4*log(2/4)
H(mild)=-4/6*log(4/6)-2/6*log(2/6)
H(cool)=-3/4*log(3/4)-1/4*log(1/4)

IG(temperature)=0.03

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

- Class distribution (C_1="yes", C_2="no"):
- p_1=9/14, p_2=5/14
- $H(p_1,p_2)$=0.94

Humidity

	C1 (yes)	C2(no)	Frequency
High	3/7	4/7	7/14
Normal	6/7	1/7	7/14

H(high)=-3/7*log(3/7)-4/7*log(4/7)
H(normal)=-6/7*log(6/7)-1/7*log(1/7)

IG(humidity)=0.15

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

H(false)=-6/8*log(6/8)-2/8*log(2/8)
H(true)=-3/6*log(3/6)-3/6*log(3/6)

IG(windy)=0.05

- Class distribution (C_1="yes", C_2="no"):
- p_1=9/14, p_2=5/14
- $H(p_1,p_2)$=0.94

Windy

	C1 (yes)	C2(no)	Frequency
False	6/8	2/8	8/14
True	3/6	3/6	6/14

Choosing the splitting attribute

Example

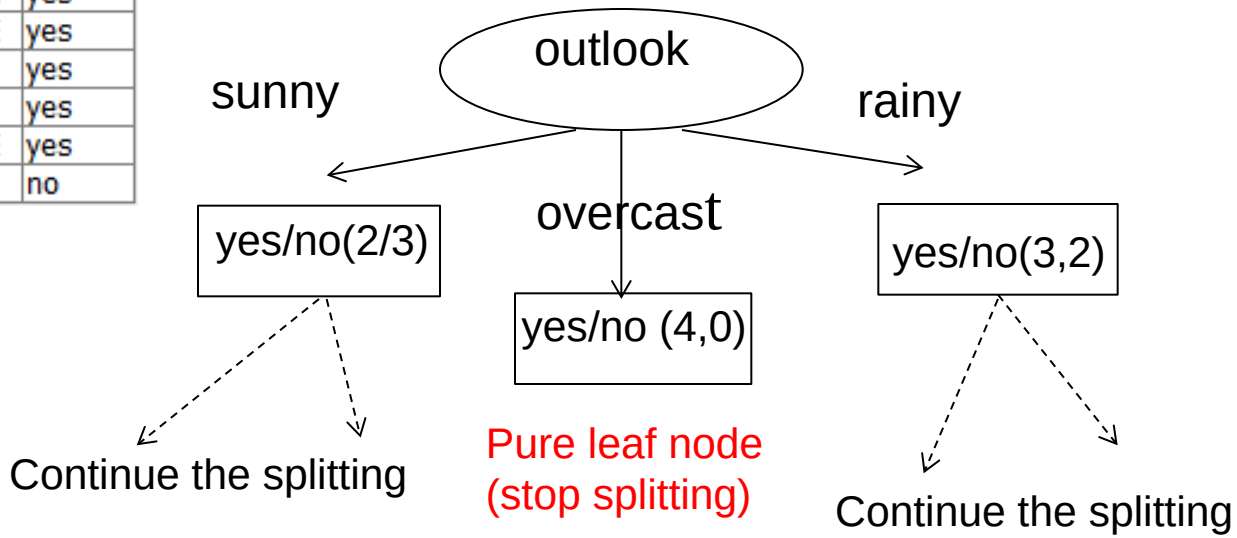
Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Information gain for each attribute:

- IG(outlook)=0.25
- IG(temperature)=0.03
- IG(humidity)=0.15
- IG(windy)=0.05

First splitting attribute: outlook

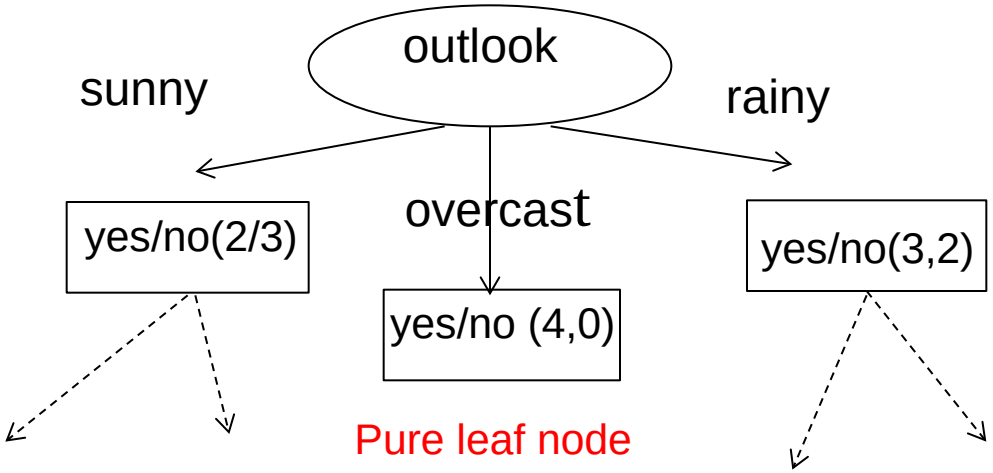


Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no



Pure leaf node (stop splitting)

Temperature

	C1 (yes)	C2(no)	Freq.
Hot	0/2	2/2	2/5
Mild	1/2	1/2	2/5
Cool	1/1	0/1	1/5

Information gain for each remaining attribute:

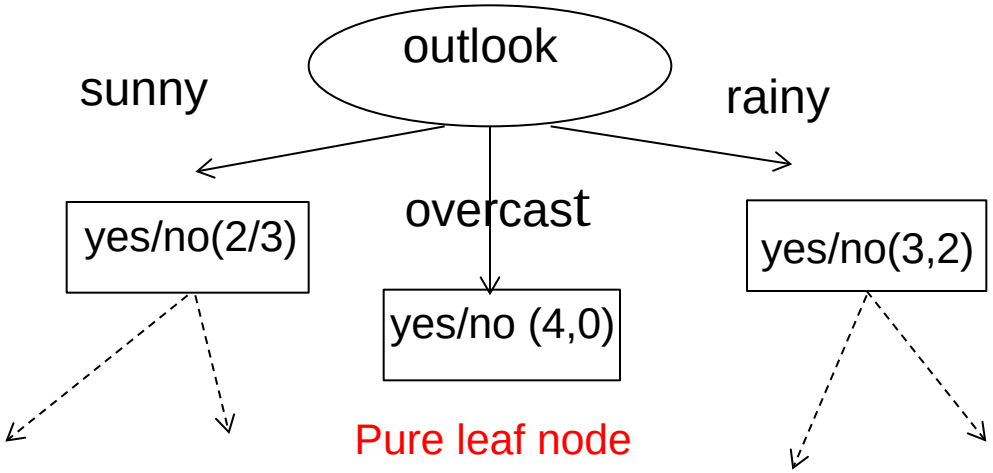
- Entropy for class distribution on “sunny” subset: H(D(sunny))=-2/5*log(2/5)-3/5*log(3/5)=0.97
- H(hot)=0, H(mild)=1, H(cool)=0
- IG(temperature)=0.97-2/5=0.57

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no



Pure leaf node (stop splitting)

Information gain for each remaining attribute:

- Entropy for class distribution on “sunny” subset: H(D(sunny))=-2/5*log(2/5)-3/5*log(3/5)=0.97
- H(high)=0, H(normal)=0
- IG(humidity)=0.97-0=0.97

Humidity

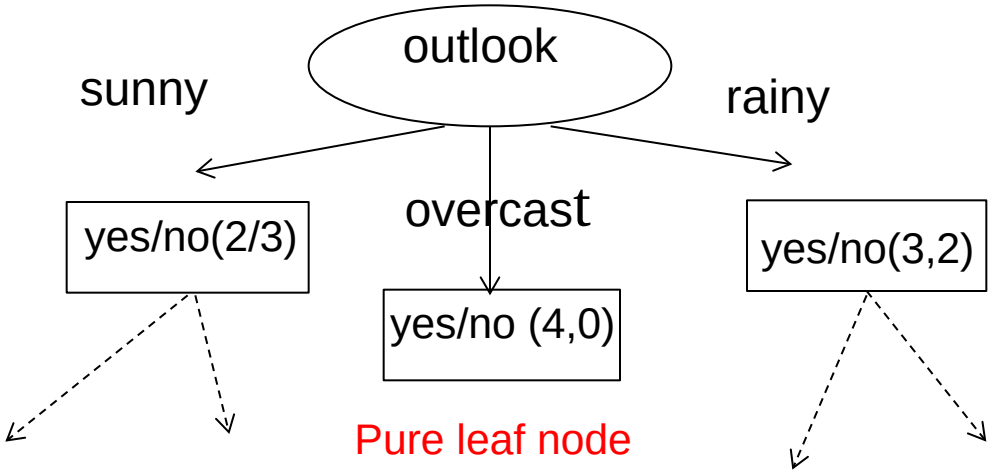
	C1 (yes)	C2(no)	Freq.
High	0/3	3/3	3/5
Normal	2/2	0/2	2/5

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no



Pure leaf node (stop splitting)

Windy

	C1 (yes)	C2(no)	Freq.
false	1/3	2/3	3/5
true	1/2	1/2	2/5

Information gain for each remaining attribute:

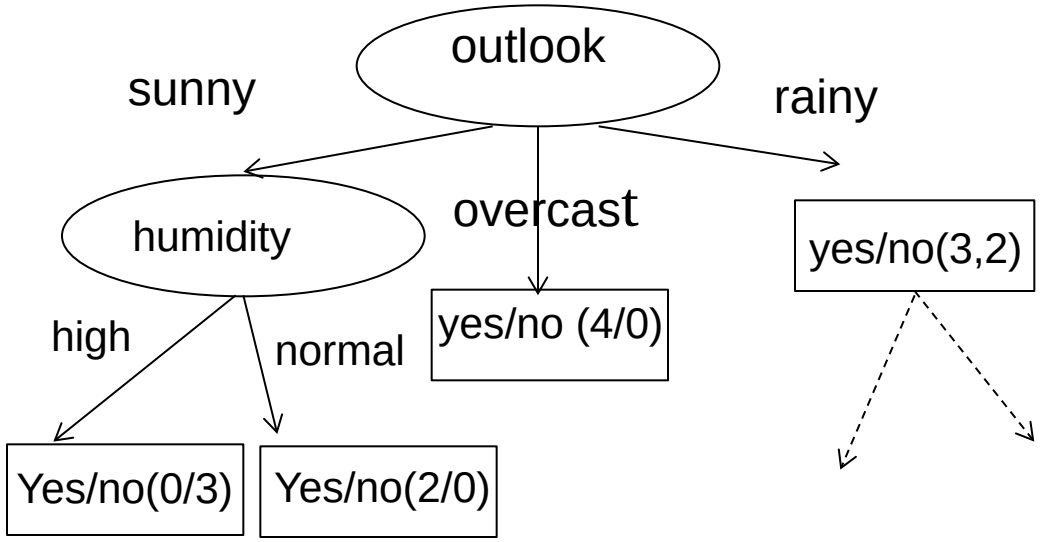
- Entropy for class distribution on “sunny” subset: H(D(sunny))=-2/5*log(2/5)-3/5*log(3/5)=0.97
- H(false)=0, H(true)=1
- IG(windy)=0.97-0.95=0.02

Choosing the splitting attribute

Example

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	**play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no



Pure leaf nodes
(stop splitting)

Information gain for each remaining attribute:

- IG(temperature)=0.97-2/5=0.57
- IG(humidity)=0.97-0=0.97
- IG(windy)=0.97-0.95=0.02

Choosing the splitting attribute

Remarks:

- Information Gain favors the attributes characterized by a larger number of values
- In order to avoid this bias it can be used the Gain Ratio:

$$GainRatio(D, A) = \frac{IG(D, A)}{H(p_1^A, p_2^A, ..., p_{m_A}^A)}$$

$$p_j^A = \frac{card(D, A = v_j)}{card(D)} \text{ (ratio of data having value } v_j \text{ for attr. } A)$$

- The splitting attribute can be selected by using the Gini impurity measure = how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset corresponding to a branch (smaller values are better)

$$Gini(p_1, p_2, ..., p_n) = 1 - \sum_{i=1}^{n} p_i^2$$

Algorithms for Decision Tree Induction

ID3:

- Input: dataset D
- Output: decision tree (input nodes labelled with attributes, leaf nodes labelled with classes, edges labelled with attribute values)

```
DTinduction (D, DT, N)  /* D=dataset, DT=decision tree, N=node */
    find the best splitting attribute A
    label node N with A
    construct the splitting predicates (branches) for N
    FOR each branch i from N DO
       construct the corresponding data set Di
       create a new child node Ni
       IF  <stopping condition>
           THEN   label Ni with the dominant class in Di (Ni is a leaf node)
           ELSE    DTinduction(Di,DT, Ni)
```

Algorithms for Decision Tree Induction

C4.5 = improvement of ID3 with respect to

- Continuous attributes:
  - incorporates a discretization procedure for continuous attributes
- Missing values:
  - During the induction process the instances with missing values are ignored
  - During the classification the missing values of the instance to be classified are imputed
- Splitting attribute:
  - It uses the Gain Ratio as attribute selection (in order to be more robust with respect to the number of values)
- Pruning:
  - Some subtrees are replaced with leaf nodes (if the classification error is not increased significantly) – bottom-up approach

Remark:

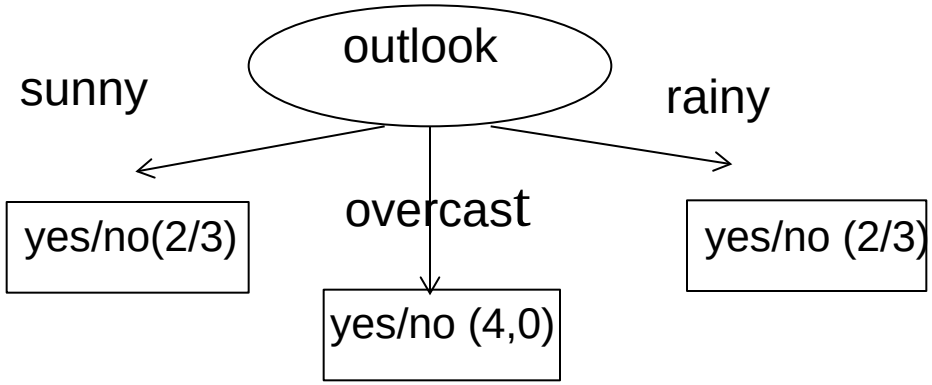
C5.0 – commercial variant of C4.5

J48 – Weka implementation of C4.5

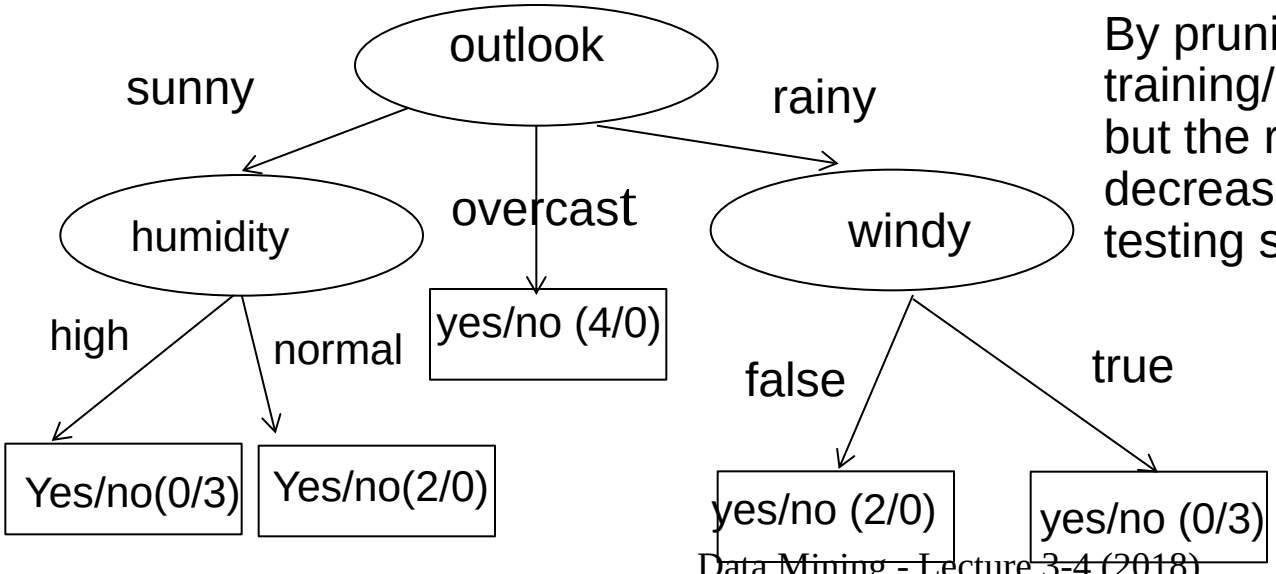
Algorithms for Decision Tree Induction

Pruning:

- Some subtrees are replaced with leaf nodes (if the classification error is not increased significantly) – bottom-up approach



Unpruned tree: error = 0



Pruned tree: error = 4/14

By pruning the error on the training/ validation set is increased but the risk of overfitting could be decreased (error of an unseen testing set could be smaller)

Classification rules induction

Reminder: classification rules are IF … THEN statements containing:

- In the antecedent part (left hand side): conditions on the attribute values (it could be a condition concerning only one attribute or logical expressions involving several attributes)
- In the consequent part (right hand side): a class label

Example:
IF outlook=sunny THEN play=no
IF outlook=rainy THEN play=no
IF outlook=overcast THEN play=yes

Remark:
- These rules are extracted from a decision tree – each branch of the tree leads to a rule
- The conditions related to nodes on the same branch should be combined by AND : IF (outlook=sunny) and (humidity=high) THEN play=no
- Rules corresponding to different branches but leading to the same consequent part (same class label) can by merged by using OR between the antecedent parts:

IF (outlook=sunny) OR (outlook=rainy) THEN play=no

Classification rules induction

The classification rules can be extracted directly from the data during a learning process by using covering algorithms

Notions:

- A rule covers a data instance if the values of the attributes match the antecedent part of the rule
- Similarly, a data instance triggers a rule if the values of the attributes match the antecedent part of the rule
- Ruleset = set of rules
- Support of a rule = fraction of the dataset which is covered by the rule and belong to the same class = |cover(R) ∩class(R) |/|D|
- Confidence of a rule = fraction of data instances covered by a rule which have the same class as the rule = |cover(R) ∩class(R)|/|cover(R)|

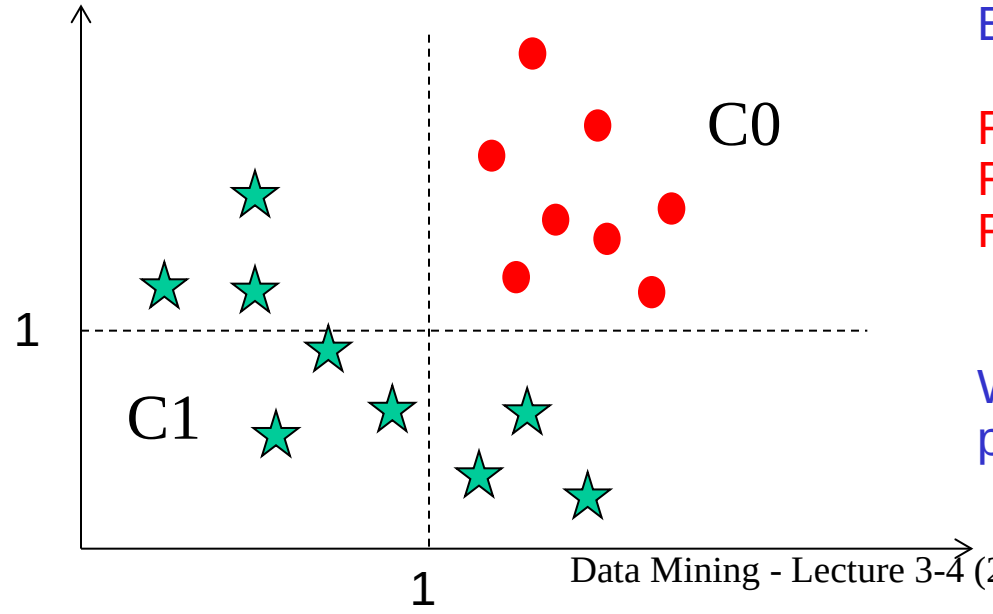
cover(R) = set of the instances covered by R
class(R) = set of instances having the same class as R
D= dataset

Classification rules induction

Notions:

- Mutually exclusive rules = the regions covered by rules are disjoint (an instance triggers only one rule)
- Exhaustive ruleset = each instance triggers at least one rule

Remark: if a ruleset is both exhaustive and the rules are mutually exclusive then it is easy to take a decision for a given instance



Example:

R1: IF x>1 and y>1 THEN C0
R2: IF x<=1 THEN C1
R3: IF x>1 and y<=1 THEN C1

What about the case when these properties are not satisfied?

Classification rules induction

Remark: if the rules are not mutually exclusive then may appear conflicts (one data instance trigger several rules which have different classes)

The conflicts can be solved in one of the following ways:

- The rules are ordered (based on a specific criterion) and the decision is taken according to the first rule triggered by the data instance (rule which matches to the instance). The sorting criteria can be related to:
  - the rule quality (e.g. high confidence) – higher confidence is better
  - the rule specificity – the rules are considered better if they are more specific (e.g. those corresponding to rare classes)
  - the rule complexity (e.g. number of conditions in the antecedent part) – simpler rules are better

  Remark: this criteria can be conflicting (a high confidence rule is not necessarily a simple rule)

- The result is the dominant class from the set of rules triggered by the data instance

Classification rules induction

Sequential covering algorithm:

Input: data set
Output: ordered set of rules

Step 1: Select a class label and determine the “best” rule which cover the data instances from D having the selected class label. Add this rule to the bottom of the ordered rule list

Step 2: Remove all data from D which match to the antecedent of the added rule. If there are still class labels to select and data in D go to Step 1

Remark:

- This is the general structure of sequential covering algorithms
- Particular algorithms differ with respect to the ordering strategy

Classification rules induction

Example: RIPPER

Particularities:

- Class-based ordering: the classes are selected in order of their size (the rare classes are selected first)
- The rules corresponding to one class are placed contiguously in the ordered list of rules
- The addition of a new rule corresponding to one class is stopped:
  - when the rule becomes too complex
  - when the new rule has a classification error (on a validation set) which is larger than a predefined threshold
- If at the end remain some uncovered data then is defined a "catch all" rule to which is assigned the dominant class

Instance based classifiers

Main idea: similar instances have similar class labels

- The classification model consists of the examples from the training dataset
  - The training process consists only on storing the training examples
- The classification of a new data instance consists of:
  - Compute the similarities (or dissimilarities) between the new data instance and the examples in the training set and find the most similar examples
  - Choose the most frequent class from the set of the similar examples

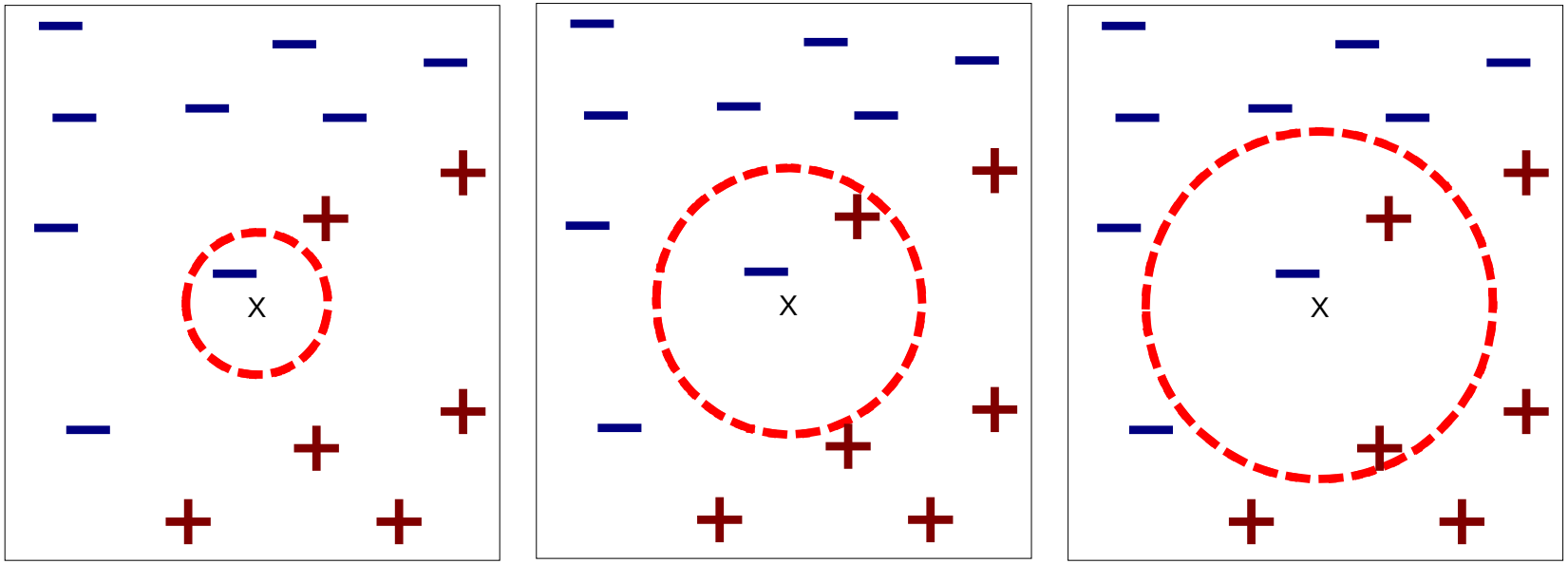
Remarks:

- Such classifiers are usually called “lazy” because the training step does not involve any effort (the entire computational effort is postponed for the classification step)
- The most popular lazy classifiers are those based on the Nearest Neighbors

Instance based classifiers

kNN – k Nearest Neighbour

- For each data to be classified:
  - Find the closest (most similar) k examples from the training dataset
  - Identify the most frequent class



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

[Tan, Steinbach, Kumar; Introduction to Data Mining, slides, 2004]

Instance based classifiers

kNN – k Nearest Neighbour

- For each data to be classified:
  - Find the closest (most similar) k examples from the training dataset
  - Identify the most frequent class

The performance of kNN classifiers depend on:

- The similarity/ dissimilarity (distance) measure
  - It depends on the attribute types and on the particularities of the problem
- The value of k (the number of neighbors)
  - Simplest case: k=1 (not very good in the case of noisy data)

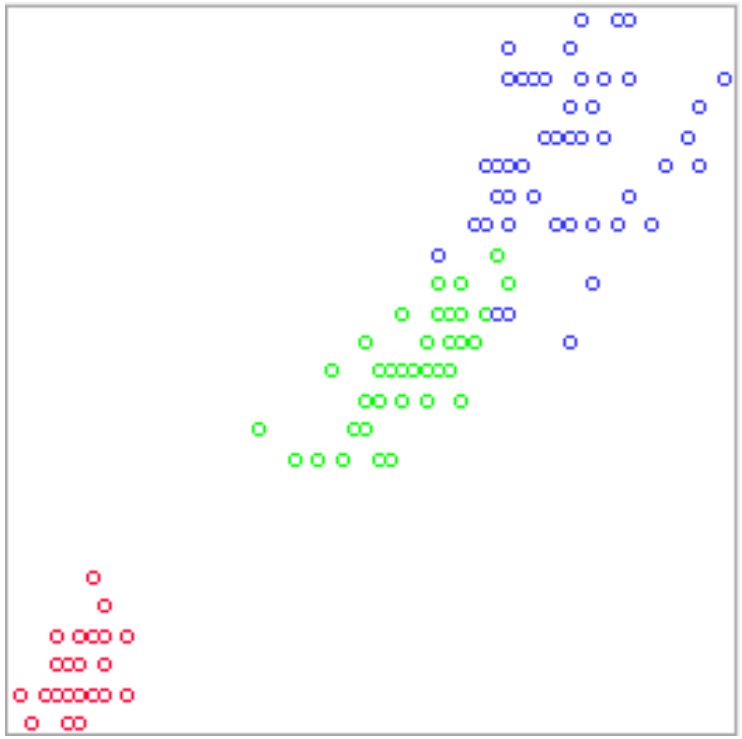
Remark: kNN induces a partition of the data space in regions; the regions are not explicitly computed but are implicitly determined by the similarity measure (and the value of k)

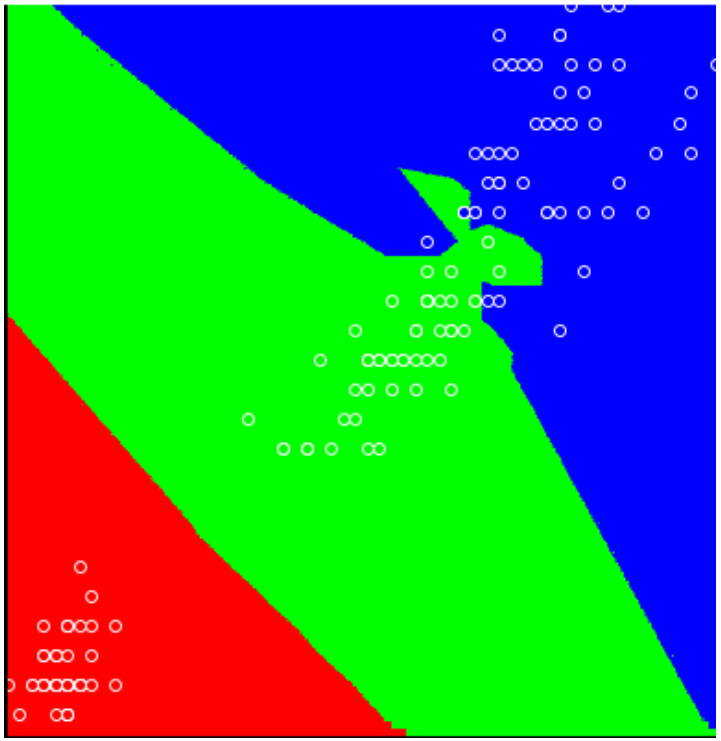
Instance based classifiers

1NN = Nearest Neighbor with one closest neighbor (based on the normalized Euclidean distance)

Illustration of the geometric boundaries. Dataset: iris2D ("petal length" and "petal width").

Plot: Weka->Visualization->BoundaryVisualizer





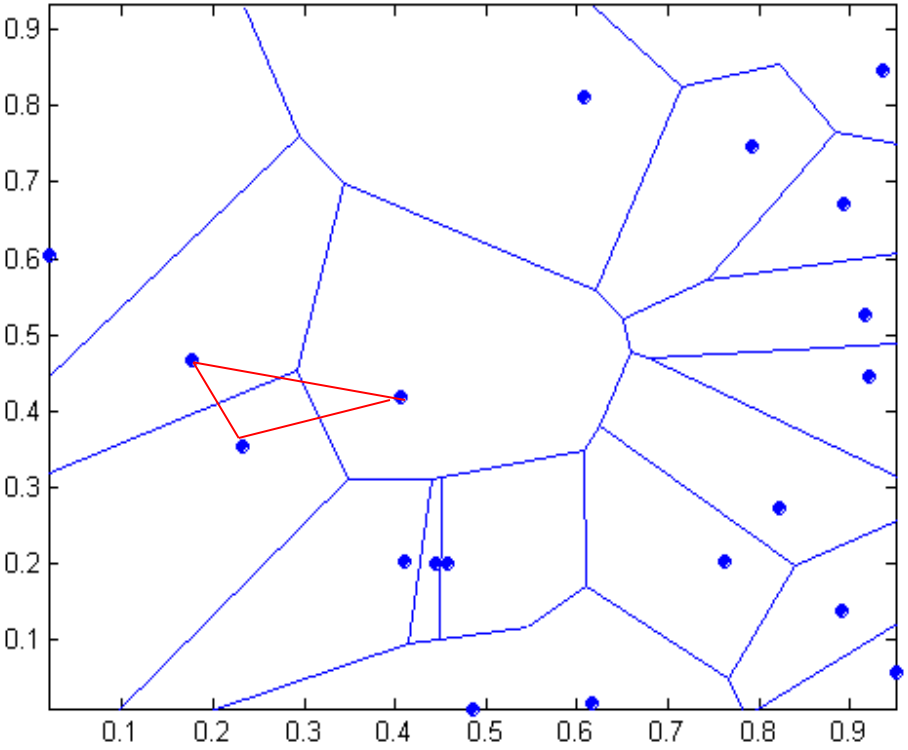
Instance based classifiers

1NN = Nearest Neighbor with one closest neighbor (based on the normalized Euclidean distance)

1NN induces a partitioning of the data space (e.g. In 2D this corresponds to a Voronoi diagram)

Remark:

Each instance in the training dataset has associated a region containing all data which are in the neighborhood of that training instance



[Tan, Steinbach, Kumar: Introduction to Data Mining, slides, 2004]

Similarity/ dissimilarity measures

Let us consider two entities (e.g. data vectors, time series etc) A and B

- A similarity measure S(A,B) is a number which is higher if A and B are more similar
- A dissimilarity measure D(A,B) is a number which is higher if A and B are less similar (or more different)

The choice of an appropriate measure depends on:

- The type of attributes
- The number of attributes
- The distribution of data

Similarity/ dissimilarity measures

Numerical attributes

Most popular dissimilarity measures:

- Euclidean distance
- Manhattan distance

$$d_p(A,B) = \sqrt[p]{\sum_{i=1}^{n}(a_i - b_i)^p} \quad \text{(Minkowski, } L_p\text{)}$$

$$d_E(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \quad \text{(Euclidean, p = 2)}$$

$$d_M(A,B) = \sum_{i=1}^{n}|a_i - b_i| \quad \text{(Manhattan, p = 1)}$$

$$d_\infty(A,B) = \max_{i=\overline{1,n}}|a_i - b_i| \quad (p = \infty)$$

Remarks:

- The Euclidean distance is invariant with respect to rotations
- If not all attributes have the same importance then weights should be included in the distance (e.g. $w_i(a_i-b_i)^2$ instead of $(a_i-b_i)^2$)

The weights can be estimated by using preprocessing methods; instead of using explicit weights the data can be normalized/ standardized

Similarity/ dissimilarity measures

Practical issues – dimensionality curse:

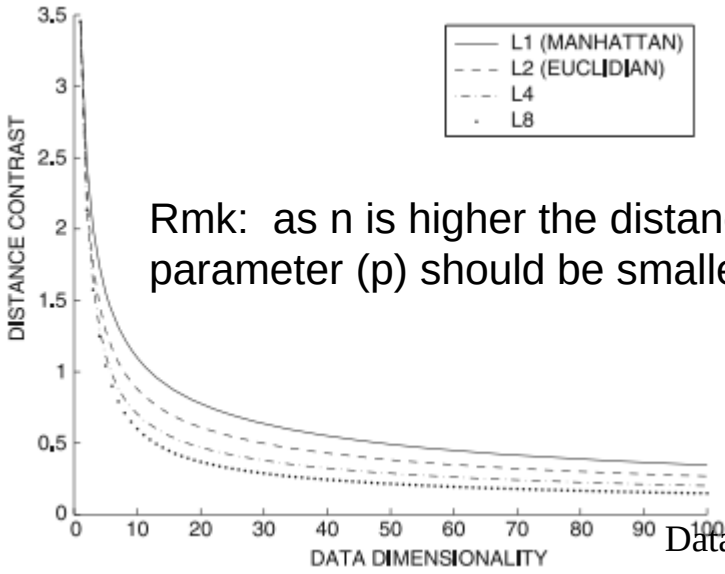
- These distances lose their discriminative power as the size (n) increases =>
for high-dimensional data the distance-based classifiers may be qualitatively ineffective

$$d_p(A,B) = \sqrt[p]{\sum_{i=1}^{n}(a_i - b_i)^p} \quad \text{(Minkowski, } L_p\text{)}$$

$$d_E(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \quad \text{(Euclidean, p = 2)}$$

$$d_M(A,B) = \sum_{i=1}^{n}|a_i - b_i| \quad \text{(Manhattan, p = 1)}$$

$$d_\infty(A,B) = \max_{i=\overline{1,n}}|a_i - b_i| \quad (p = \infty)$$



Rmk: as n is higher the distance parameter (p) should be smaller

$$\text{Distance contrast} : \frac{d_{\max} - d_{\min}}{\sigma}$$

$$d_{\max}, d_{\min} = \text{largest and smallest distance}$$

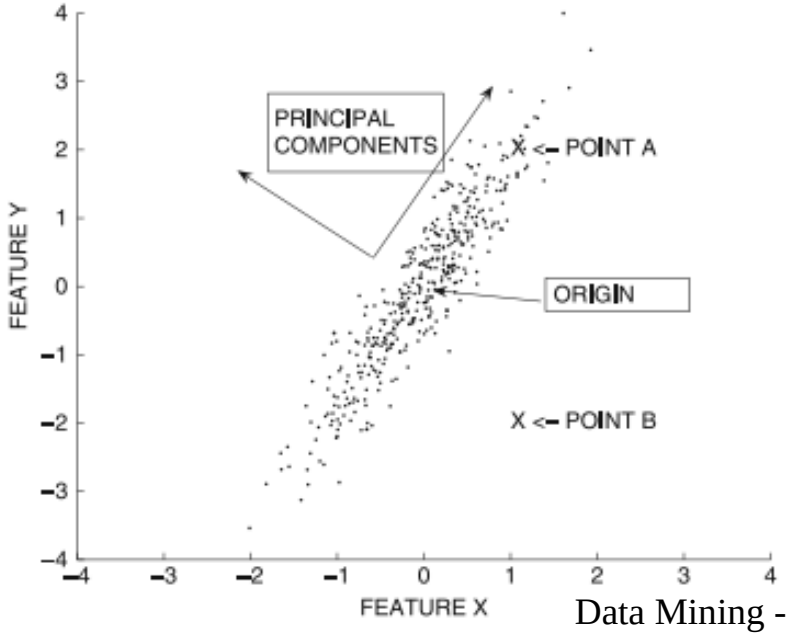
$$\sigma = \text{standard deviation of distances}$$

Aggarwal, Data Mining Textbook, 2015

Similarity/ dissimilarity measures

Practical issues – impact of data distribution

Q: Which is closer to the origin? Point A or point B?



Aggarwal, Data Mining Textbook, 2015

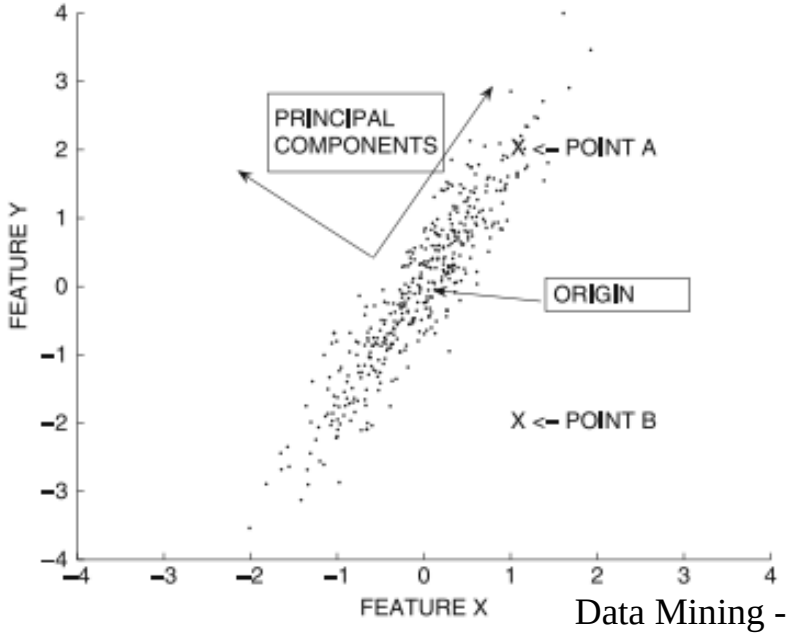
Similarity/ dissimilarity measures

Practical issues – impact of data distribution

Q: Which is closer to the origin? Point A or point B?

A: The Euclidean distances d(O,A) and d(O,B) are equal. However taking into account the data distribution A is more in agreement with the data distribution than B, thus it should be considered that A is "closer" to O than B

New question: how can be quantified the data distribution in the distance?



Mahalanobis distance

$$d_{Mah}(A,B) = \sqrt{(A-B)^T C^{-1}(A-B)}$$

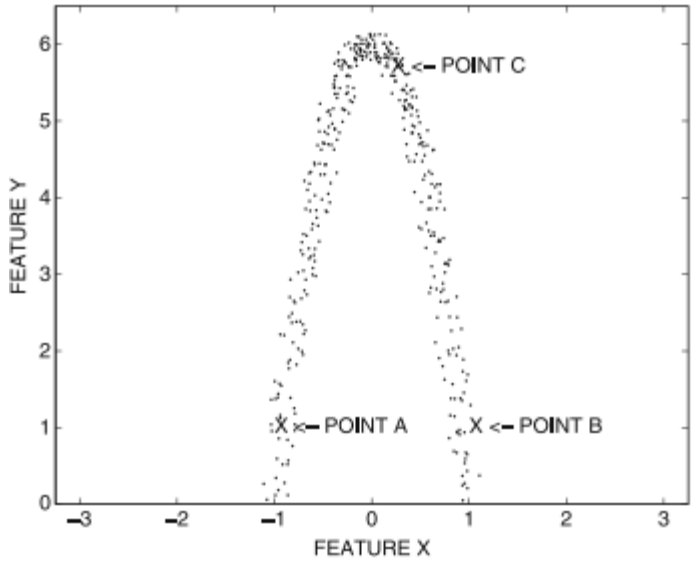
C^{-1} = inverse of the covariance matrix

Aggarwal, Data MiningTextbook, 2015

Similarity/ dissimilarity measures

Practical issues – impact of data distribution

Q: Is the distance between A and B smaller than the distance between B and C?



Aggarwal, Data Mining Textbook, 2015

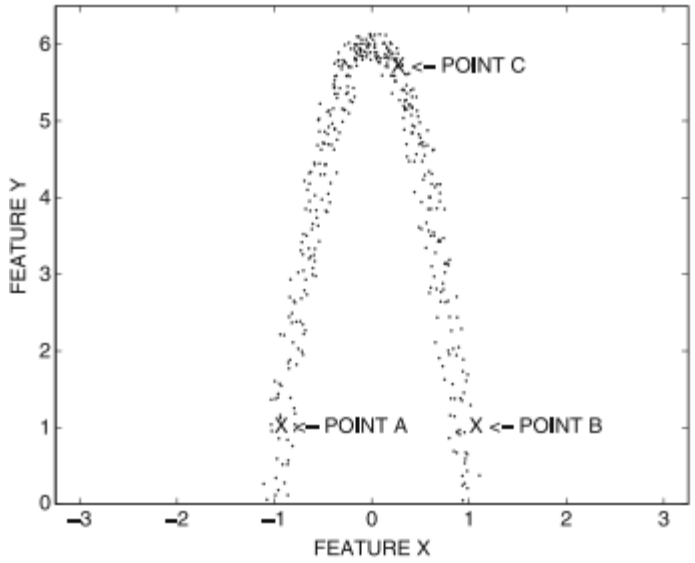
Similarity/ dissimilarity measures

Practical issues – impact of data distribution

Q: Is the distance between A and B smaller than the distance between B and C?

A: Yes, if we ignore the data distribution and we use the Euclidean distance

However, the data distribution provides a context for the problem to be solved and in this context d(A,B)>d(B,C)



Geodesic distance:

- Construct a graph by using the points as nodes and by defining edges between nearest neighbors (each point is connected to its first k nearest neighbors)
- Compute the distance between two points as the shortest path in the graph.

Aggarwal, Data Mining Textbook, 2015

Similarity/ dissimilarity measures

Numerical attributes – similarity measure

- Cosine measure: $sim(A,B)=A^TB/(||A||\ ||B||)$ (scalar product between A and B divided by the product of the norms)

Remark:

- In the case of normalized data vectors ($||A||=||B||=1$) the similarity between the vectors is maximal when the Euclidean distance between them is minimal:

$$d_E^2(A,B) = (A-B)^T(A-B) = A^TA - 2A^TB + B^TB =$$
$$= 2(1\text{-}A^TB) = 2(1\text{-}sim(A,B))$$

Similarity/ dissimilarity measures

Nominal attributes

Approach 1: Transformation of nominal attributes in numerical ones (by binarization) and use similarity/dissimilarity measures for binary vectors:

- Dissimilarity: Hamming distance = Manhattan distance: $d_H(A,B)=d_M(A,B)$
- Similarity: Jaccard coefficient

$$J(A,B)=\frac{\sum_{i=1}^{n} a_i b_i}{\sum_{i=1}^{n}(a_i^2+b_i^2-a_i b_i)}=\frac{card(S_A \cap S_B)}{card(S_A \cup S_B)}$$

Remark: S_A and S_B are the subsets of the global set of n attributes which correspond to the membership vectors A and B, respectively

Similarity/ dissimilarity measures

Nominal attributes

Approach 2: Use local similarity measures (between attribute values)

$$S(A,B) = \sum_{i=1}^{n} S(a_i, b_i)$$

$$S(a_i, b_i) = \begin{cases} 1 & \text{if } a_i = b_i \\ 0 & \text{if } a_i \neq b_i \end{cases}$$

Remark: the similarities which are unusual should be considered more significant than those which are frequent

$$S(a_i, b_i) = \begin{cases} 1/f^2(a_i) & \text{if } a_i = b_i \\ 0 & \text{if } a_i \neq b_i \end{cases}$$

$$f(a_i) = \text{frequency of the value } a_i \text{ in the dataset (for i - th attribute)}$$

Similarity/ dissimilarity measures

Mixed attributes: the measures corresponding to numerical and nominal attributes should be aggregated (using specific weights)

$$S(A,B) = \lambda S_{numerical}(A,B) + (1-\lambda) S_{nominal}(A,B)$$

Other types of data:

- Strings (e.g. text or biological sequences) - use of the edit distance
- Concepts (e.g. nodes in an ontology) – path-based distances in trees/ graphs
- Graphs (e.g. social or biological networks) – amount of similar small structures (patterns)

Nearest Neighbour: choice of k

The performance of kNN is sensitive to the number (k) of neighbours

Extreme cases:

- k=1 - the classifier is not robust (errors in the dataset could mislead the classification result)
- k=N - it is equivalent to ZeroR being based only on the distribution of the data in classes (the sensitivity of the classifier to the underlying data is lost)

How to choose k?

- Trial-and-error approach: try different values and choose the one providing the best performance

Nearest Neighbour: computational cost

In an naïve implementation the classification step requires the computation of N distances (or similarities) for a dataset with N elements (d-dimensional vectors) and the selection of the smallest k distances ➔ O(Nd+kN)

When N is large this could be costly (as it should be done for each new data to classify)

Possible approaches:

- Create indexing data structures allowing to find the k nearest neighbours in an efficient way
- Identify small clusters in the data set and replace each such cluster with a single prototype
- Use instance (prototype) selection = select the most representative instances from the dataset (not easy to decide which ones are representative)

Next lecture

- Probabilistic Classification: Naïve Bayes
- Classifiers based on decision functions
  - Neural Networks
  - Support Vector Machines