C | A | U Christian-Albrechts-Universität zu Kiel Technische Fakultät

Inf-KDDM:
Knowledge Discovery and Data Mining

Winter Term 2019/20

Lecture 5: Classification

Lectures: Prof. Dr. Matthias Renz

Exercises: Christian Beth

Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

The classification problem

- Given:
  - a dataset of instances $D=\{t_1,t_2,\ldots,t_n\}$ and
  - a set of classes $C=\{c_1,\ldots,c_k\}$

classification is the task of learning a *target function*/ mapping $f{:}D \rightarrow C$ that assigns each t_i to a c_j.

- The mapping or target function is known informally as a *classification model*.

ID	Age	Car type	Risk
1	23	Family	high
2	17	Sport	high
3	43	Sport	high
4	68	Family	low
5	32	Truck	low

Predictor attributes: Age, Car type

Class attribute: risk={high, low}

The classification problem

Classification vs Prediction

- Classification
  - predicts categorical (discrete, unordered) class labels
  - Constructs a model (classifier) based on a training set
  - Uses this model to predict the class label for new unknown-class instances

- Prediction
  - is similar, but may be viewed as having infinite number of classes (cf. Regression)

A simple classifier

ID	Age	Car type	Risk
1	23	Family	high
2	17	Sport	high
3	43	Sport	high
4	68	Family	low
5	32	Truck	low

A simple classifier:

- if Age > 50 then Risk= low;
- if Age $\leq$ 50 and Car type =Truck then Risk=low;
- if Age $\leq$ 50 and Car type $\neq$ LKW then Risk = high.

Applications

- Credit approval
  - Classify bank loan applications as e.g. safe or risky.
- Fraud detection
  - e.g., in credit cards
- Churn prediction
  - E.g., in telecommunication companies
- Target marketing
  - Is the customer a potential buyer for a new computer?
- Medical diagnosis
- Character recognition
- …

Classification techniques

- Typical classification approach:
  - Create specific model by evaluating training data (or using domain experts’ knowledge).
    - Assess the quality of the model
  - Apply model developed to new data.
- Classes must be predefined!!!
- Many techniques
  - Decision trees
  - Naïve Bayes
  - kNN
  - Neural Networks
  - Support Vector Machines
  - ….

Classification technique (detailed)

- **Model construction:** describing a set of predetermined classes
  - The set of tuples used for model construction is the **training set**
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label
  - The model is represented as classification rules, decision trees, or mathematical formula
- **Model evaluation:** estimate accuracy of the model
  - The set of tuples used for model evaluation is the **test set**
  - The class label of each tuple/sample in the test set is known in advance.
  - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
  - Test set is independent of training set, otherwise over-fitting will occur
- **Model usage:** for classifying future or unknown objects
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known.

predefined class values

Class attribute: tenured={yes, no}

Training set

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

known class label attribute

Test set

NAME	RANK	YEARS	TENURED	PREDICTED
Maria	Assistant Prof	3	no	no
John	Associate Prof	7	yes	no
Franz	Professor	3	yes	yes

known class label attribute

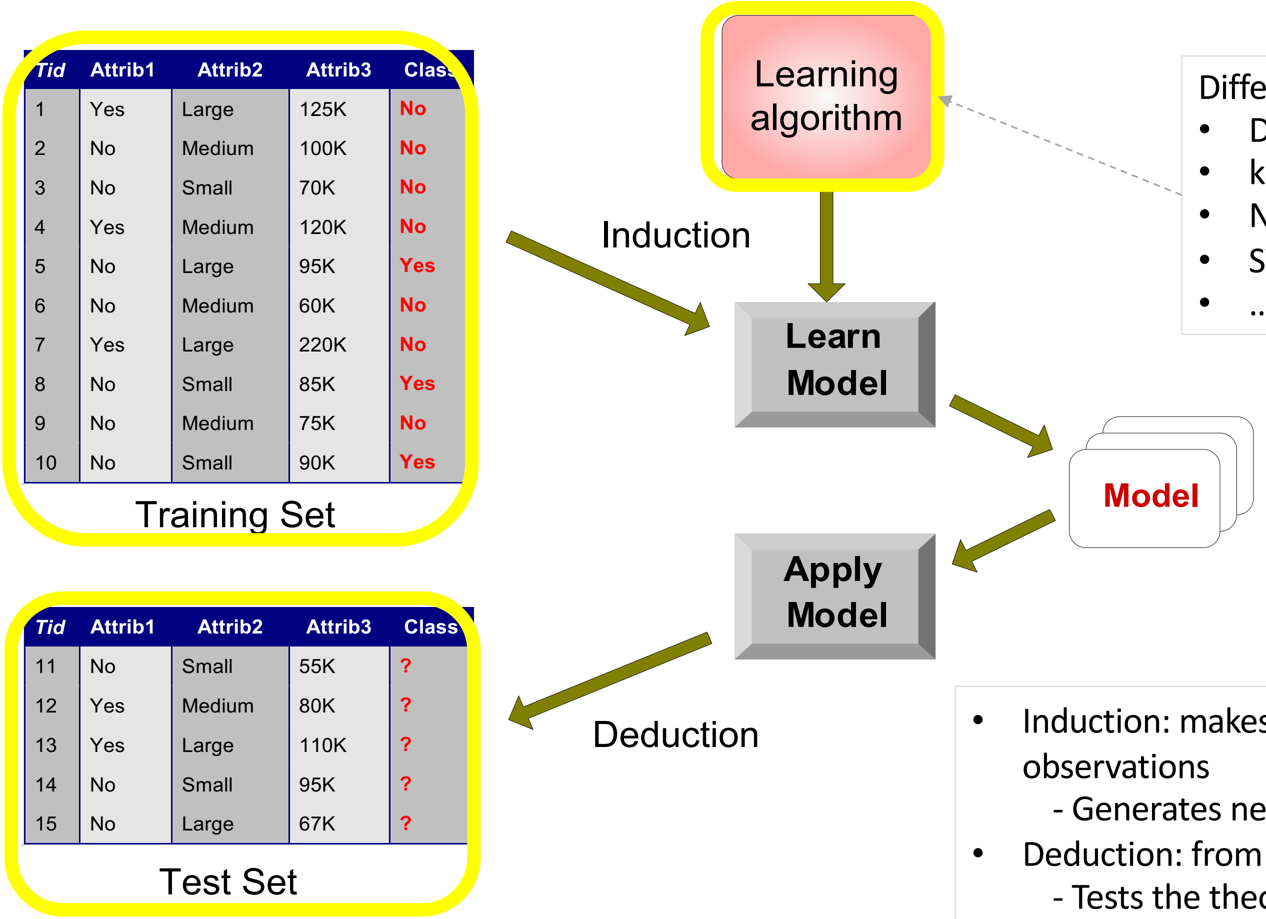
predicted class value by the model

NAME	RANK	YEARS	TENURED	PREDICTED
Jeff	Professor	4	?	yes
Patrick	Associate Prof	8	?	yes
Maria	Associate Prof	2	?	no

unknown class label attribute

predicted class value by the model

General approach for building a classification model



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	?

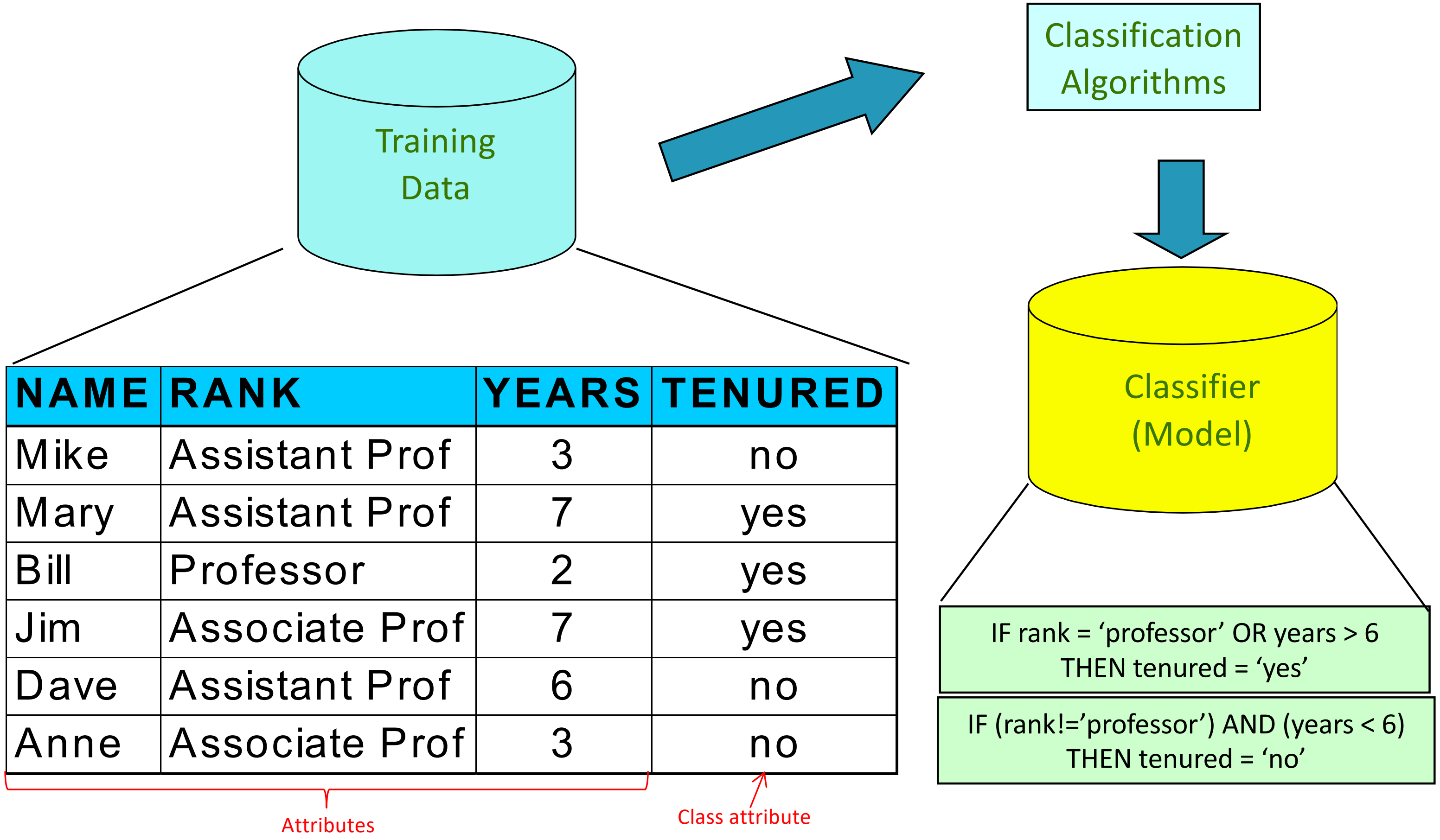
Test Set

Different classification techniques (or classifiers)

- Decision trees
- kNN
- Neural networks
- SVMs
- …

- Induction: makes broad generalizations from specific observations
  - Generates new theory emerging from the data
- Deduction: from general to specific
  - Tests the theory

Model construction

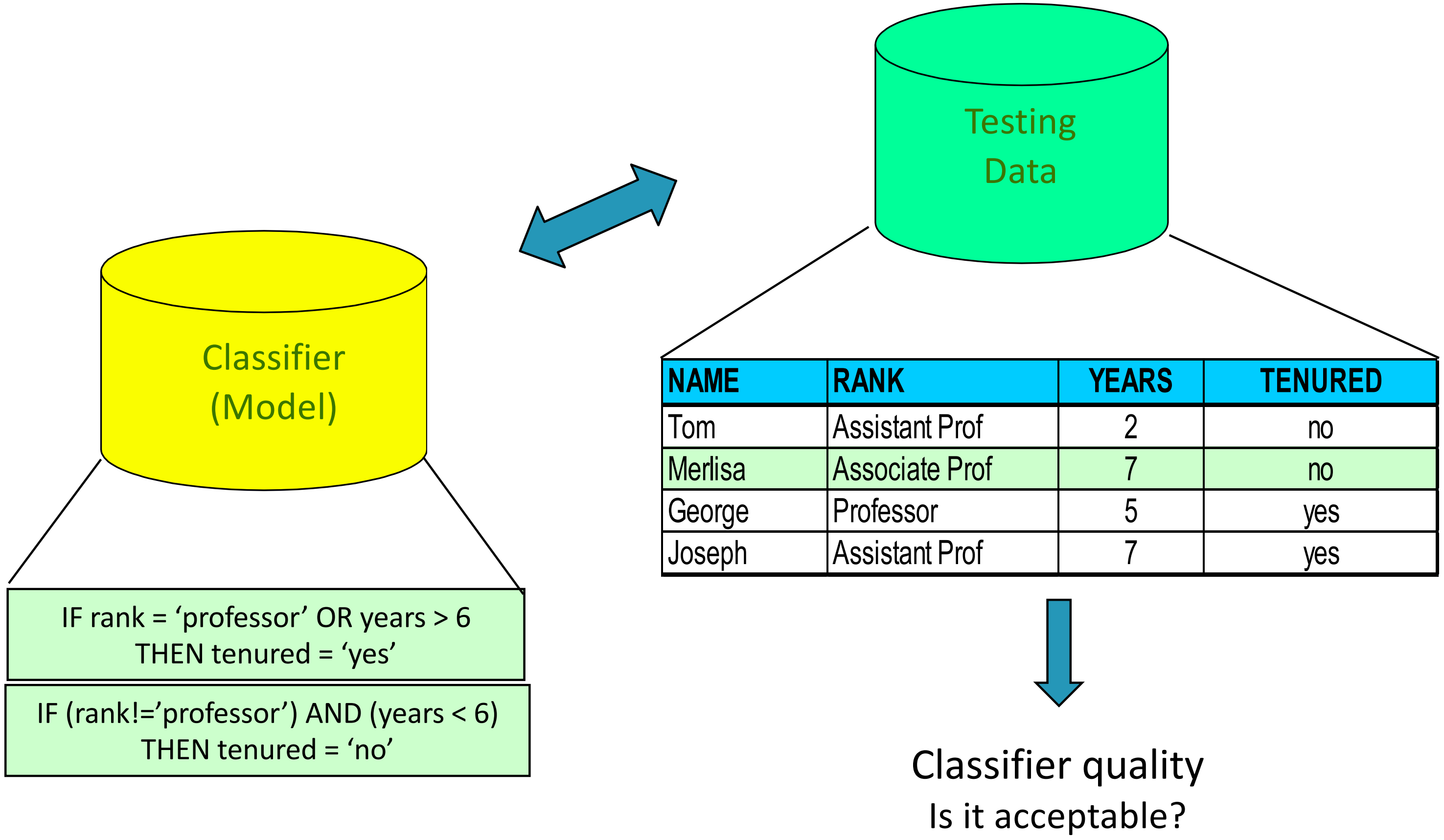


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor' OR years > 6
THEN tenured = 'yes'

IF (rank!='professor') AND (years < 6)
THEN tenured = 'no'

Model evaluation



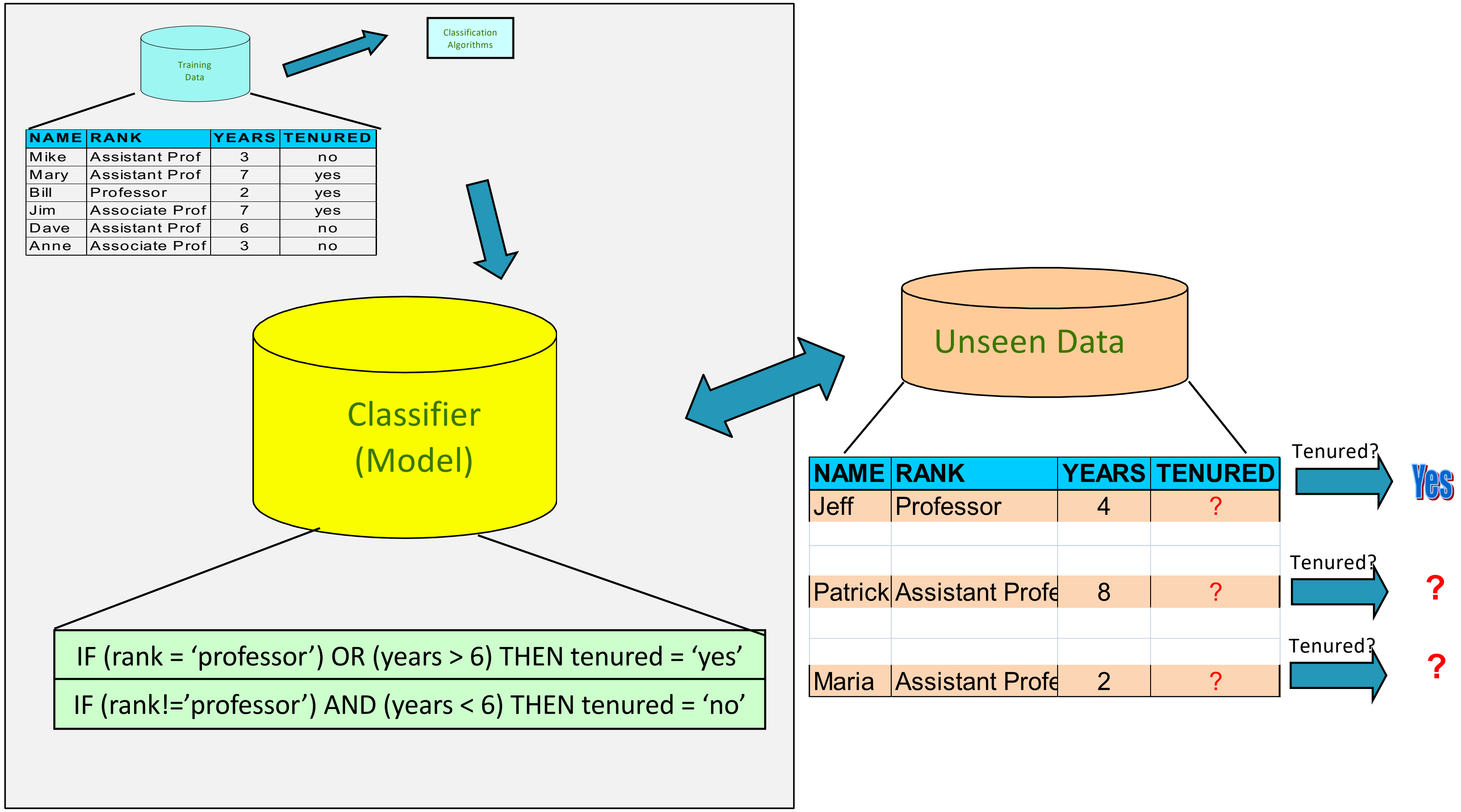
IF rank = 'professor' OR years > 6
THEN tenured = 'yes'

IF (rank!='professor') AND (years < 6)
THEN tenured = 'no'

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classifier quality
Is it acceptable?

Model usage for prediction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF (rank = ‘professor’) OR (years > 6) THEN tenured = ‘yes’

IF (rank!=’professor’) AND (years < 6) THEN tenured = ‘no’

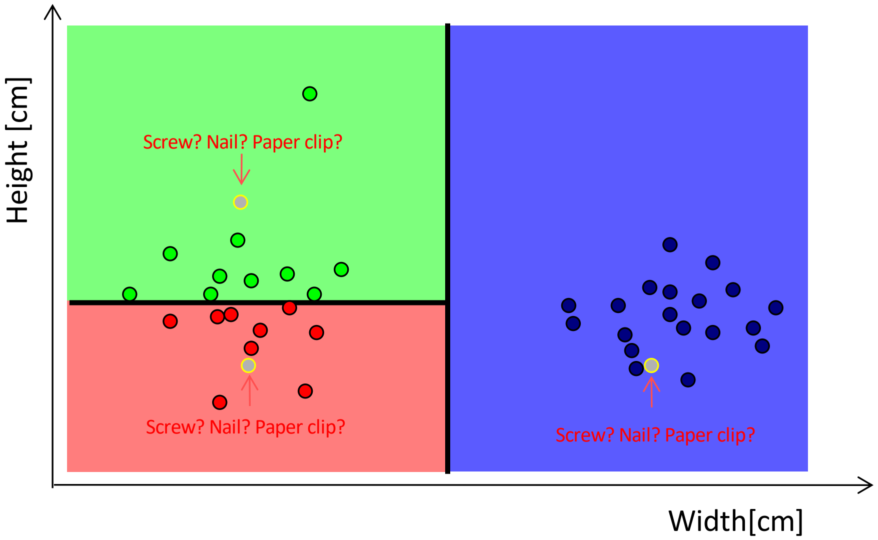
NAME	RANK	YEARS	TENURED
Jeff	Professor	4	?
Patrick	Assistant Profe	8	?
Maria	Assistant Profe	2	?

A supervised learning task

- Classification is a supervised learning task
  - Supervision: The training data (observations, measurements, etc.) are accompanied by *labels* indicating the *class* of the observations
  - New data is classified based on the training set

- Clustering is an unsupervised learning task
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc., the goal is to group the data into groups of similar data (clusters)

Supervised learning example



Classification model

● Screw

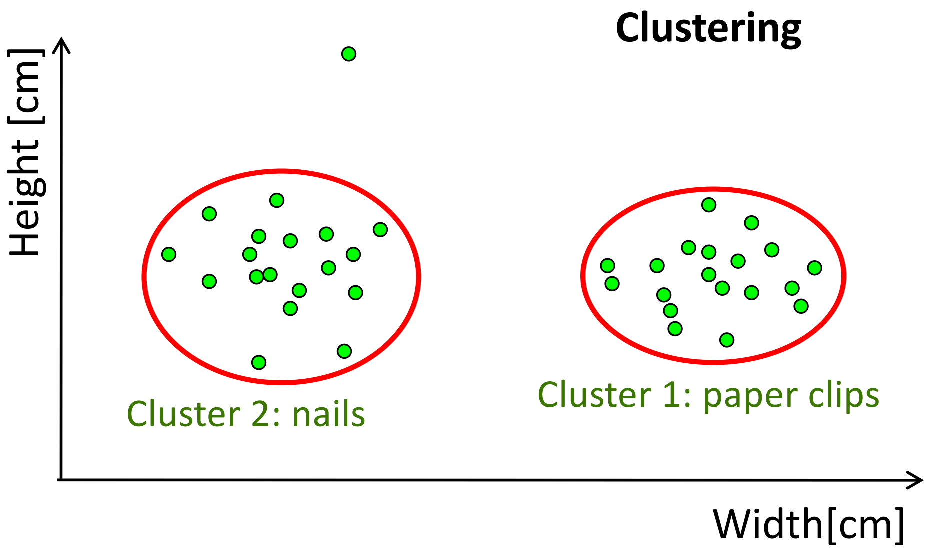
● Nails

● Paper clips

○ New object (unknown class)

Question:
What is the class of a new object???
Is it a screw, a nail or a paper clip?

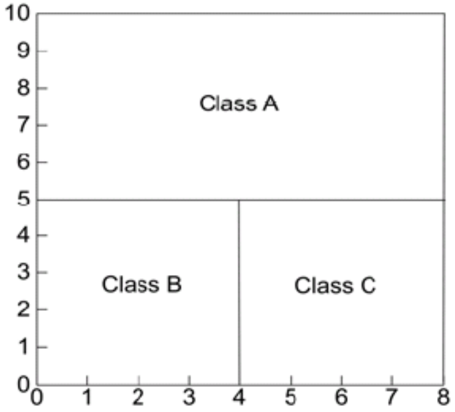
Unsupervised learning example

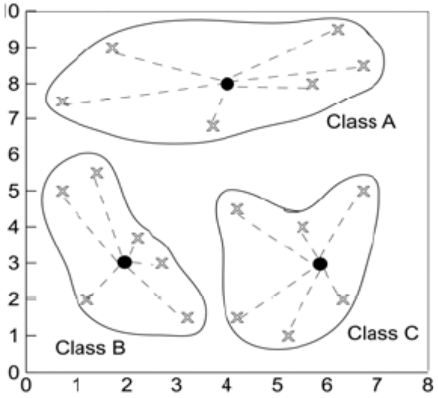


Question:
Is there any structure in data (based on their characteristics, i.e., width, height)?

Classification techniques

- Statistical methods
  - Bayesian classifiers etc
- Partitioning methods
  - Decision trees etc
- Similarity based methods
  - K-Nearest Neighbors etc





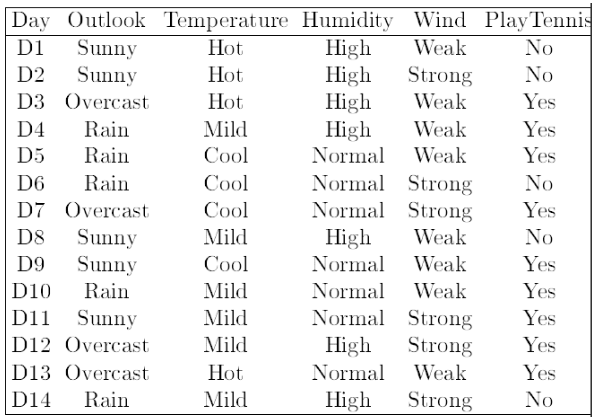
Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

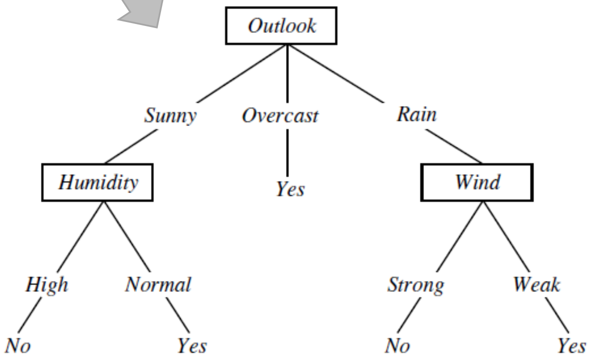
Decision tree (DTs) classifiers

- One of the most popular classification methods
- DTs are included in many commercial systems nowadays
- Easy to interpret, human readable, intuitive
- Simple and fast methods
- Many algorithms have been proposed
  - ID3 (Quinlan 1986)
  - C4.5 (Quinlan 1993)
  - CART (Breiman et al 1984)
  - …

Training set



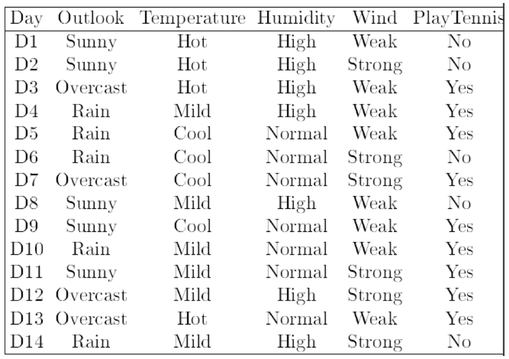
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



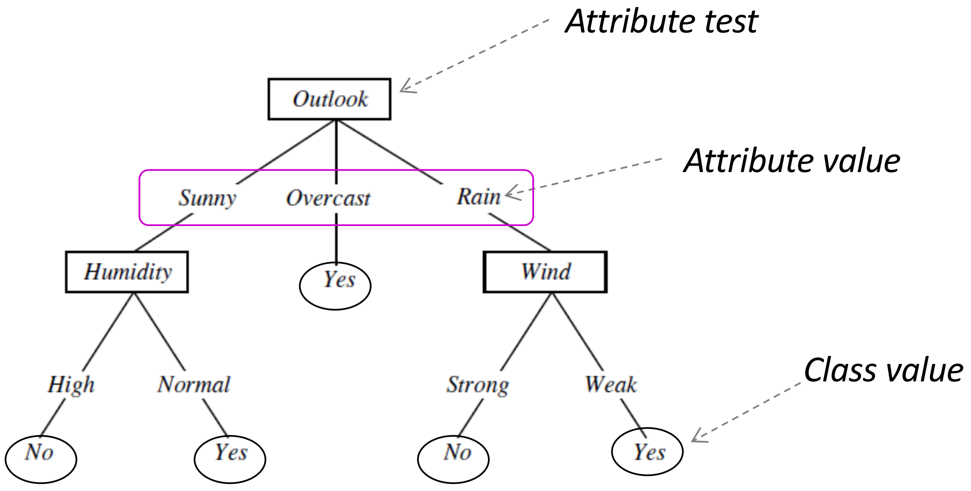
Representation 1/2

- Representation
  - Each *internal node* specifies a test of some predictor attribute
  - Each *branch* descending from a node corresponds to one of the possible values for this attribute
  - Each *leaf node* assigns a class label
- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

Training set



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



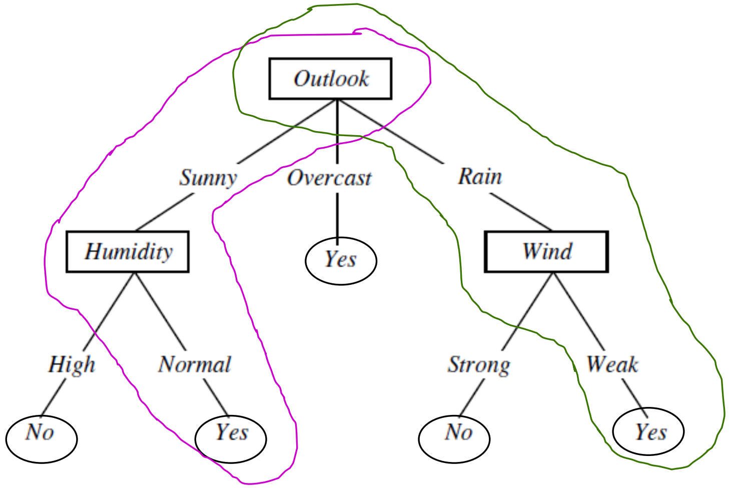
Attribute test

Attribute value

Class value

Representation 2/2

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of the instances
  - Each path from the root to a leaf node, corresponds to a conjunction of attribute tests
  - The tree corresponds to a disjunction of these conjunctions
- We can “translate” each path into IF-THEN rules (human readable)



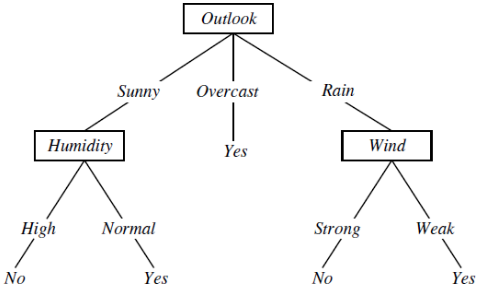
IF ((Outlook = Sunny) ^ (Humidity = Normal)),
THEN (Play tennis=Yes)

IF ((Outlook = Rain) ^ (Wind = Weak)),
THEN (Play tennis=Yes)

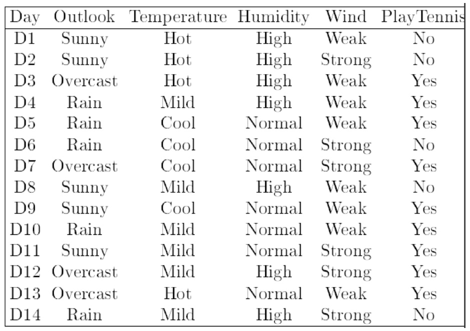
The basic decision tree learning algorithm

Basic algorithm (ID3, Quinlan 1986)

- The tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root node
- The question is “*which attribute should be tested at the root*?”
  - Attributes are evaluated using some statistical measure, which determines how well each attribute alone classifies the training examples.
  - The *best splitting attribute* is selected and used as the *test attribute* at the root.
- For each possible value of the test attribute, a descendant of the root node is created and the instances are mapped to the appropriate descendant node.
- The procedure is repeated for each descendant node, so instances are partitioned recursively.



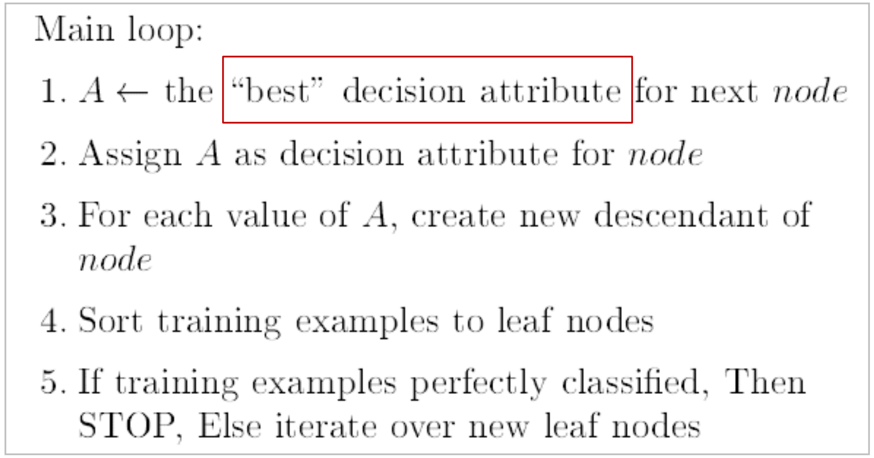
Training set



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

The basic decision tree learning algorithm

- Pseudocode

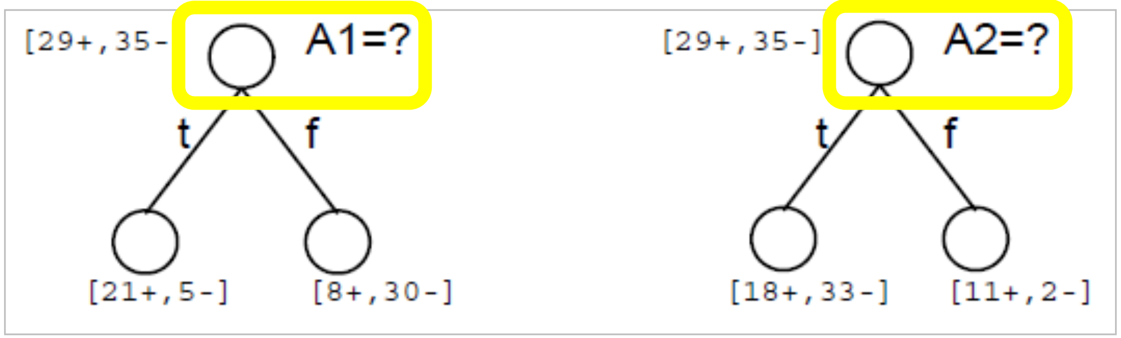


```
Main loop:
 1. $A \leftarrow$ the "best" decision attribute for next node
 2. Assign $A$ as decision attribute for node
 3. For each value of $A$, create new descendant of
    node
 4. Sort training examples to leaf nodes
 5. If training examples perfectly classified, Then
    STOP, Else iterate over new leaf nodes
```

- When do we stop partitioning?
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – *majority voting* for classifying the leaf

Which attribute is the best?

- Which attribute to choose for splitting? A1 or A2?

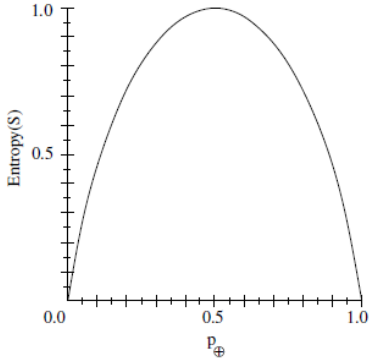


- The goal is to select the attribute that is most useful for classifying examples.
- By useful we mean that the resulting partitioning is as *pure* as possible
  - A partition is pure if all its instances belong to the same class.
- Different attribute selection measures
  - Information gain, gain ratio, gini index, …
  - all based on the degree of impurity of the parent (before splitting) vs the children nodes (after splitting)

Entropy for measuring impurity of a set of instances

- Let S be a collection of positive and negative examples for a binary classification problem, C={+, -}.
  - p_+: the percentage of positive examples in S
  - p_-: the percentage of negative examples in S
- Entropy measures the impurity of S:

$$Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$



  - Entropy = 0, when all members belong to the same class
  - Entropy = 1, when there is an equal number of positive and negative examples
- Examples :
  - Let S: [9+,5-] $Entropy(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$
  - Let S: [7+,7-] $Entropy(S) = -\frac{7}{14}\log_2(\frac{7}{14}) - \frac{7}{14}\log_2(\frac{7}{14}) = 1$
  - Let S: [14+,0-] $Entropy(S) = -\frac{14}{14}\log_2(\frac{14}{14}) - \frac{0}{14}\log_2(\frac{0}{14}) = 0$

in the general case
(*k-classification problem)*

$$Entropy(S) = \sum_{i=1}^{k} - p_i \log_2(p_i)$$

Attribute selection measure: Information gain

- Used in ID3
- It uses entropy, a measure of pureness of the data
- The information gain *Gain(S, A)* of an attribute *A* relative to a collection of examples *S* measures the entropy reduction in *S* due to splitting on *A*:

$$Gain(S,A) = \boxed{Entropy(S)} - \boxed{\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}$$

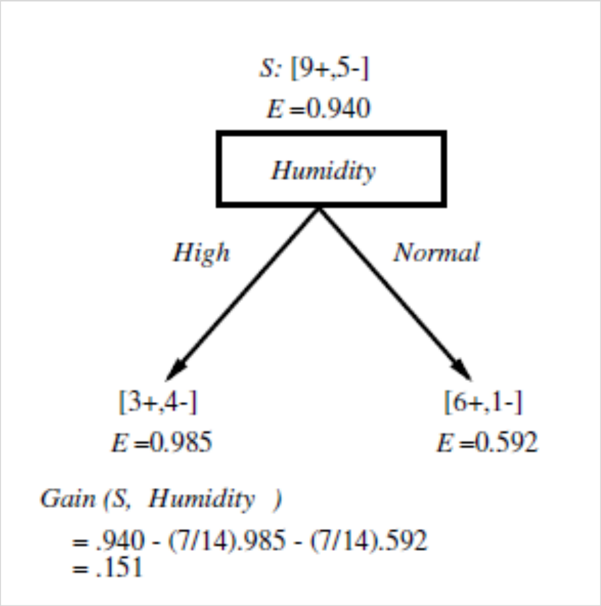
Before splitting

After splitting on A

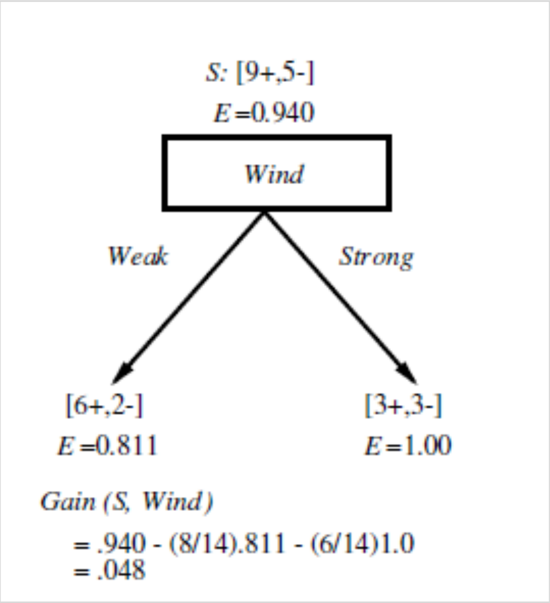
- Gain measures the expected reduction in entropy due to splitting on *A*
- The attribute with the higher entropy reduction is chosen for splitting

Information Gain example 1

- “Humidity” or “Wind”? Which attribute to choose for splitting?



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

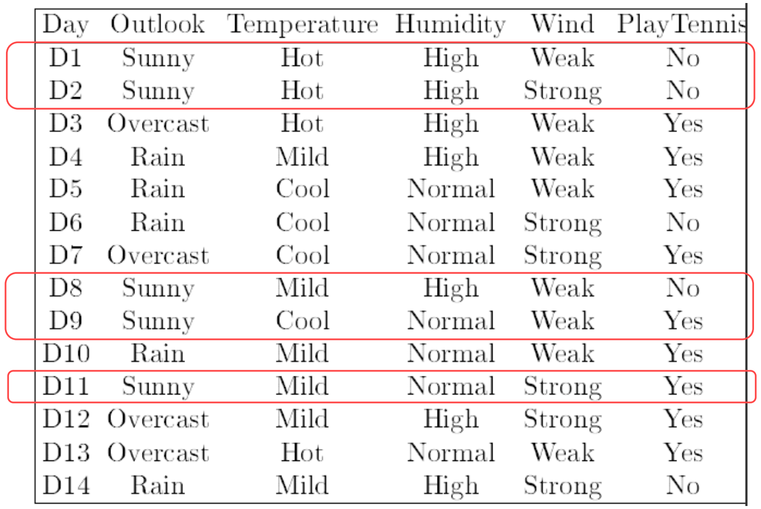


Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

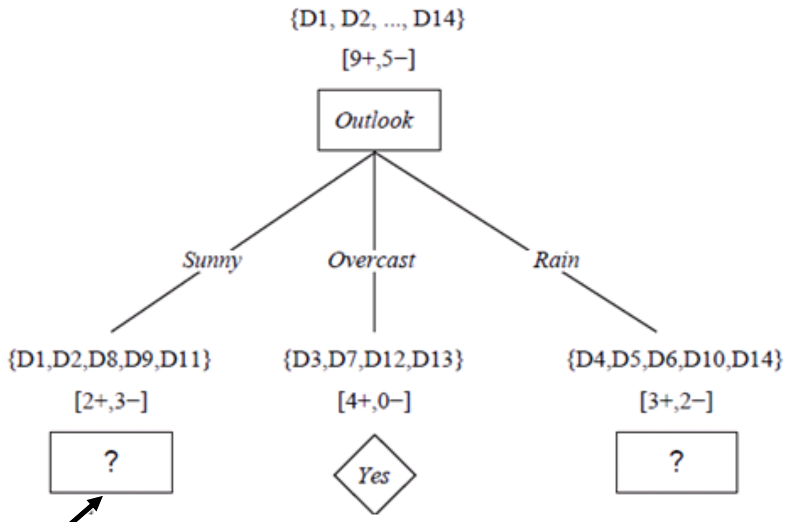
Which attribute is chosen?

Information Gain example 2

Training *set*



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Which attribute should we choose for splitting here?

$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}\ , Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}\ , Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Which attribute is chosen?

Attribute selection measure: Gain ratio

- Information gain is biased towards attributes with a large number of values
  - Consider the attribute ID (unique identifier)
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem, which normalizes the gain by split information:

$$\text{GainRatio}(S,A) = \frac{\text{Gain}(S,A)}{\text{SplitInfo}(S,A)}$$

Measures the information w.r.t. classification

Measures the information generated by splitting S into |Values(A)|partitions

$$\text{SplitInfo}(S,A) = -\sum_{v \in Values(A)} P_v \bullet \log_2(P_v) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \bullet \log_2\left(\frac{|S_v|}{|S|}\right)$$

  - High split info: partitions have more or less the same size (uniform)
  - Low split info: few partitions hold most of the tuples (peaks)
  - If an attribute produces many splits → high SplitInfo()→low GainRatio().
- The attribute with the maximum gain ratio is selected as the splitting attribute

Example: Split information

- Example:
  - Humidity={High, Low}

$$SplitInformation(S, Humidity) = -\frac{7}{14} \times \log_2(\frac{7}{14}) - \frac{7}{14} \times \log_2(\frac{7}{14}) = 1$$

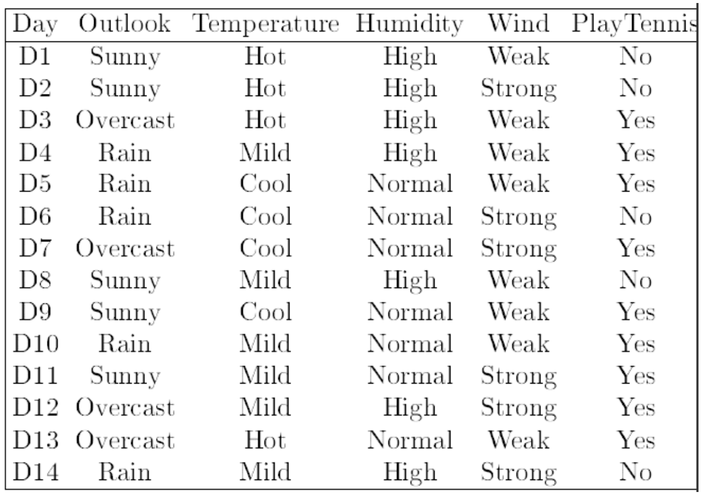
  - Wind={Weak, Strong}

$$SplitInformation(S, Wind) = -\frac{8}{14} \times \log_2(\frac{8}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) = 0.9852$$

  - Outlook = {Sunny, Overcast, Rain}

$$SplitInformation(S, Outlook) = -\frac{5}{14} \times \log_2(\frac{5}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{5}{14} \times \log_2(\frac{5}{14}) = 1.5774$$

Training set



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute selection measure: Gini Index 1/2

- Used in CART
- Let a dataset S containing examples from k classes. Let p_j be the probability of class j in S. The Gini Index of S is given by:

$$\mathrm{Gini}(S) = 1 - \sum_{j=1}^{k} p_j^2$$

  - Gini index considers a binary split for each attribute
- If *S* is split based on attribute A into two subsets S_1 and S_2:

$$Gini(S, A) = \frac{|S_1|}{|S|} Gini(S_1) + \frac{|S_2|}{|S|} Gini(S_2)$$

- Reduction in impurity:

$$\Delta Gini(S, A) = Gini(S) - Gini(S, A)$$

- The attribute A that provides the smallest G*ini*(*S,A*) (or the largest reduction in impurity) is chosen to split the node

Attribute selection measure: Gini Index 2/2

- How to find the binary splits?
  - For discrete-valued attributes, we consider all possible subsets that can be formed by values of A (next slides)
  - For numerical attributes, we find the split points (next slides)

Gini index example for discrete-valued attributes 1/2

- Let *D* has 14 instances
  - 9 of class *buys_computer = "yes"*
  - 5 in *buys_computer = "no"*
- The Gini Index of *D* is:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{k} p_j^2 \Rightarrow \text{Gini}(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Let the attribute *"Income"* = {*low, medium, high*} .
- To generate the binary splits for *"Income"*, we check all possible subsets:
  - (*{low,medium} and {high}*)
  - (*{low,high} and {medium}*)
  - (*{medium,high} and {low}*)

Gini index example for discrete-valued attributes 2/2

- For each subset, we check the Gini Index:
- For example, (*{low,medium} and {high}*) split result in D_1 (#10 instances) and D_2(#4 instances)

$$Gini_{\{low,medium\}\,and\,\{high\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_2)$$

$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$
$$= 0.450$$

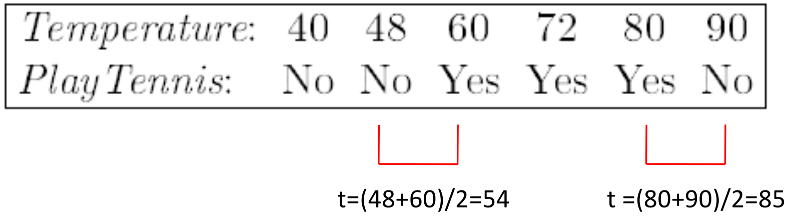
- For the remaining binary split partitions:

$$Gini_{\{low,high\}\,and\,\{medium\}}(D) = 0.315$$
$$Gini_{\{medium,high\}\,and\,\{low\}}(D) = 0.300$$

- So, the best binary split for income is on *({medium, high} and {low})*

Dealing with continuous attributes 1/2

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* t for A, (A ≤ t)
  - Sort the value A in increasing order
  - Identify adjacent examples that differ in their target classification
    - Typically, every such pair suggests a potential split threshold t= $(a_i+a_{i+1})/2$
  - Select threshold t that yields the best value of the splitting criterion.



Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

t=(48+60)/2=54

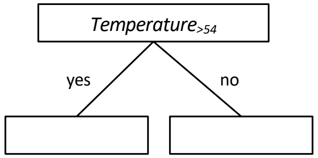
t =(80+90)/2=85

- 2 potential thresholds:Temperature$_{>54}$, Temperature $_{>85}$
- Compute the attribute selection measure (e.g. information gain) for both
- Choose the best (Temperature$_{>54}$ here)

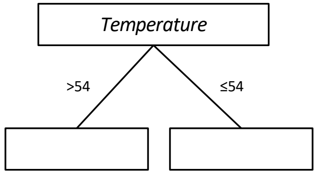
Dealing with continuous attributes 2/2

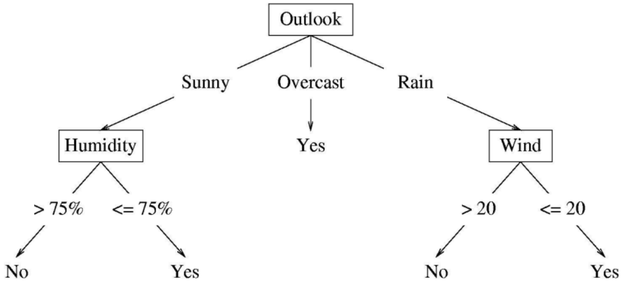
- Let t be the threshold chosen from the previous step
- Create a boolean attribute based on A and threshold t with two possible outcomes: yes, no
  - S_1 is the set of tuples in S satisfying (A >t), and S_2 is the set of tuples in S satisfying (A ≤ t)

How it looks



or



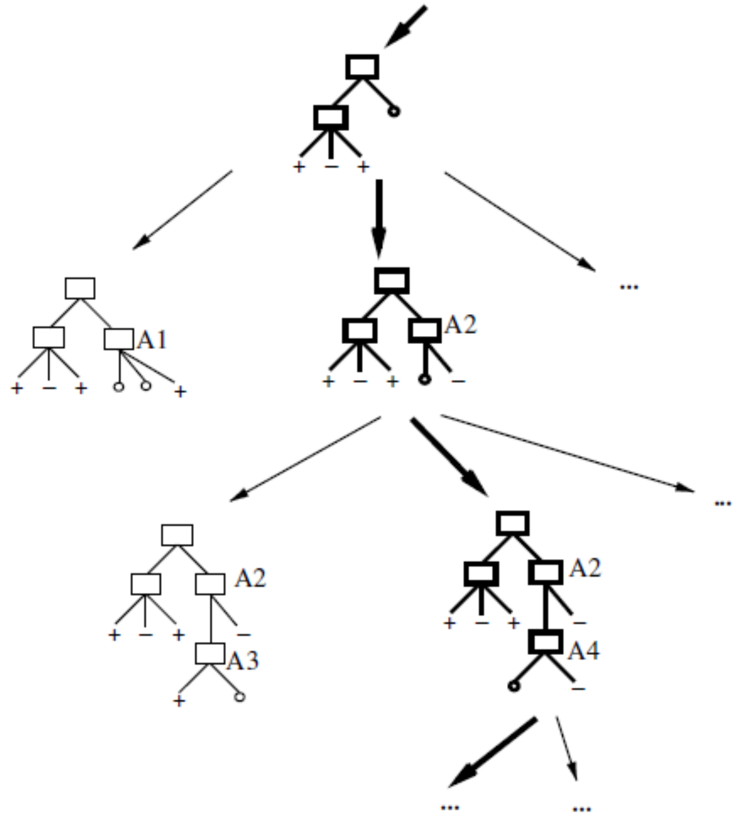


An example of a tree for the play tennis problem when attributes Humidity and Wind are continuous

Comparing Attribute Selection Measures

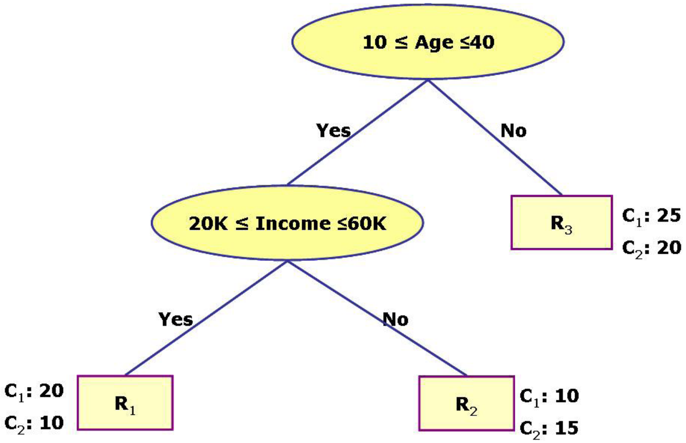
- The three measures, are commonly used and in general, return good results but
  - Information gain Gain(S,A):
    - biased towards multivalued attributes
  - Gain ratio GainRatio(S,A) :
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions
- Several other measures exist

Hypothesis search space (by ID3)

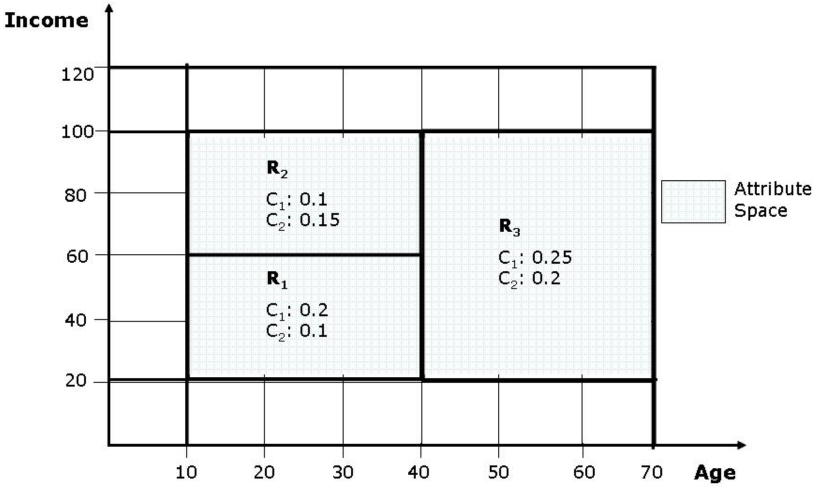


- Hypothesis space is complete
  - – Solution is surely in there
- Greedy approach
- No back tracking
  - – Local minima
- Outputs a single hypothesis

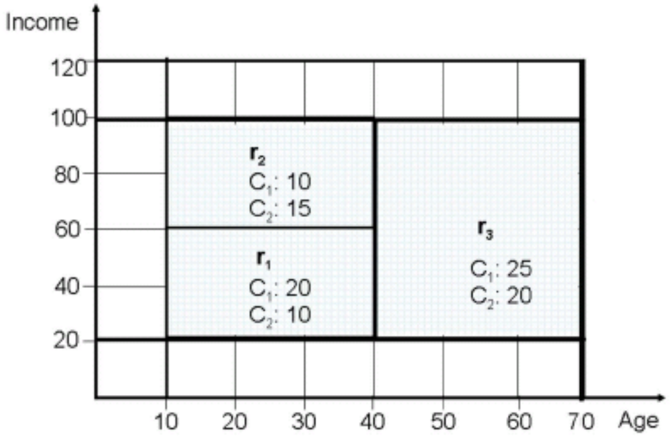
Partition-based methods



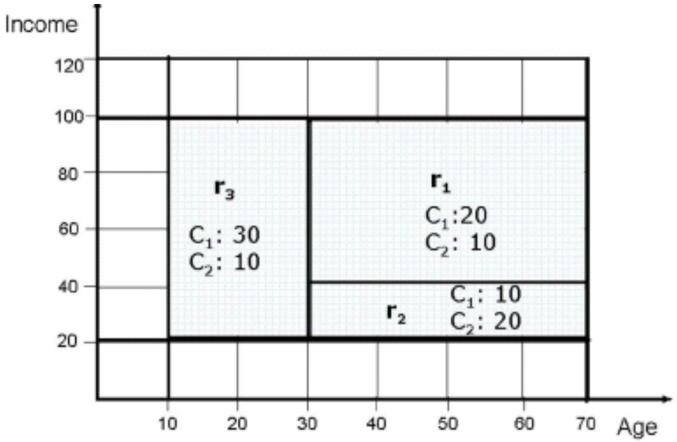
- DTs partition the space into rectangular regions
- Decision regions: axis parallel hyper-rectangles
- Decision boundary: the border line between two neighboring regions of different classes



Comparing DTs/ partitionings



DT_1



DT_2

When to consider decision trees

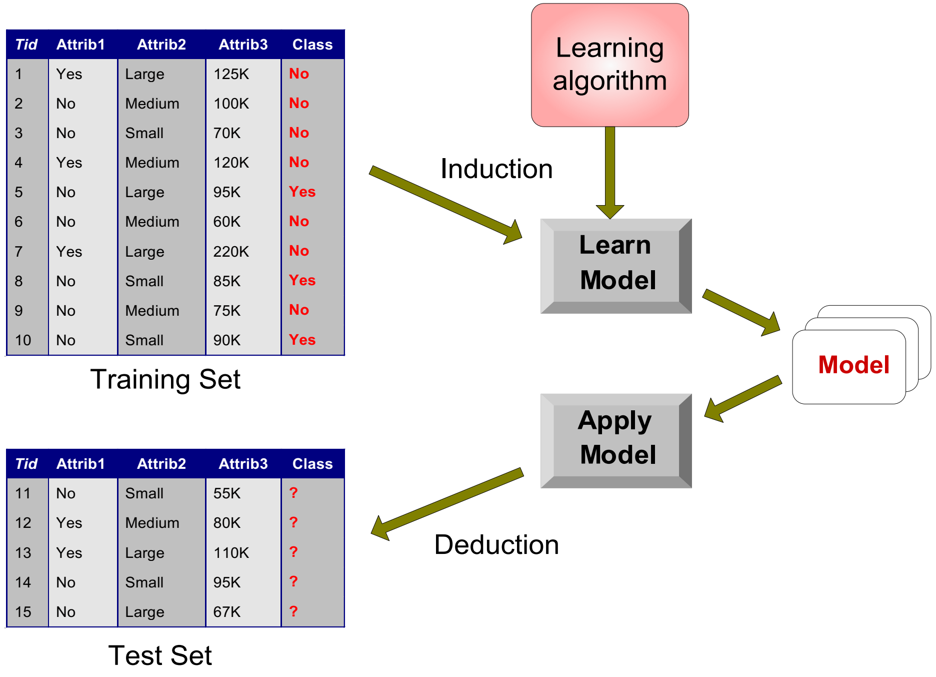
- Instances are represented by attribute-value pairs
  - Instances are represented by a fixed number of attributes, e.g. outlook, humidity, wind and their values, e.g. (wind=strong, outlook =rainy, humidity=normal)
  - The easiest situation for a DT is when attributes take a small number of disjoint possible values, e.g. wind={strong, weak}
  - There are extensions for numerical attributes also, e.g. temperature, income.
- The class attribute has discrete output values
  - Usually binary classification, e.g. {yes, no}, but also for more class values, e.g. {pos, neg, neutral}
- The training data might contain errors
  - DTs are robust to errors: both errors in the class values of the training examples and in the attribute values of these examples
- The training data might contain missing values
  - DTs can be used even when some training examples have some unknown attribute values

Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

Training vs generalization errors

- The errors of a classifier are divided into
  - Training errors (or resubstitution error or apparent error):
    - errors commited in the training set
  - Generalization errors:
    - the expected error of the model on previously unseen examples
- A good classifier must
  1. Fit the training data &
  2. Accurately classify records never seen before
- i.e., a good model ➔ low training error & low generalization error



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

Model overfitting

- Model overfitting
  - A model that fits the training data well (low training error) but has a poor generalization power (high generalization error)
- Overfitting: Consider an hypothesis *h*
  - $error_{train}(h)$: the error of *h* in the training set
  - $error_{D}(h)$: the error of *h* in the entire distribution *D* of data (i.e., including instances beyond the training set)
  - Hypothesis *h* overfits training data if there is an alternative hypothesis *h'* in *H* such that:

$$error_{train}(h) < error_{train}(h')$$

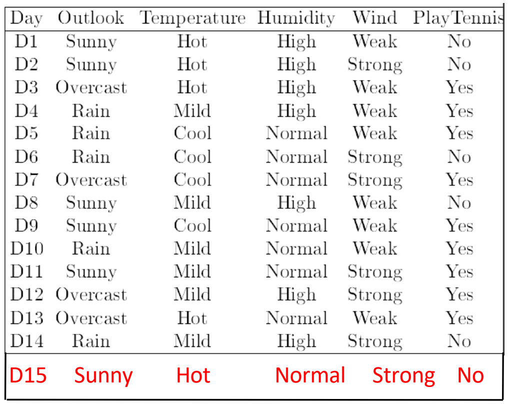
and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

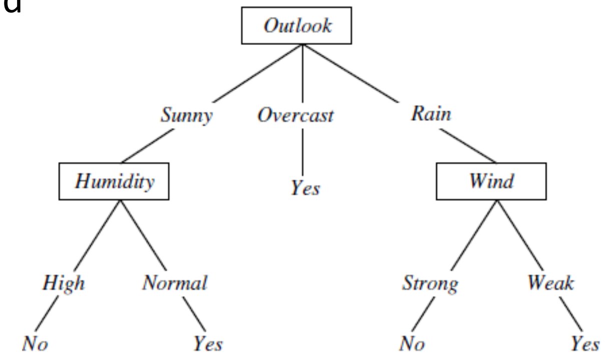
Decision trees overfitting

- An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Very good performance in the training (already seen) samples
  - Poor accuracy for unseen samples
- Example
  - Let us add a *noisy/outlier* training example (D_{15}) to the training set
  - How the earlier tree (built upon training examples D_1-D_{14}) would be effected?

Training set



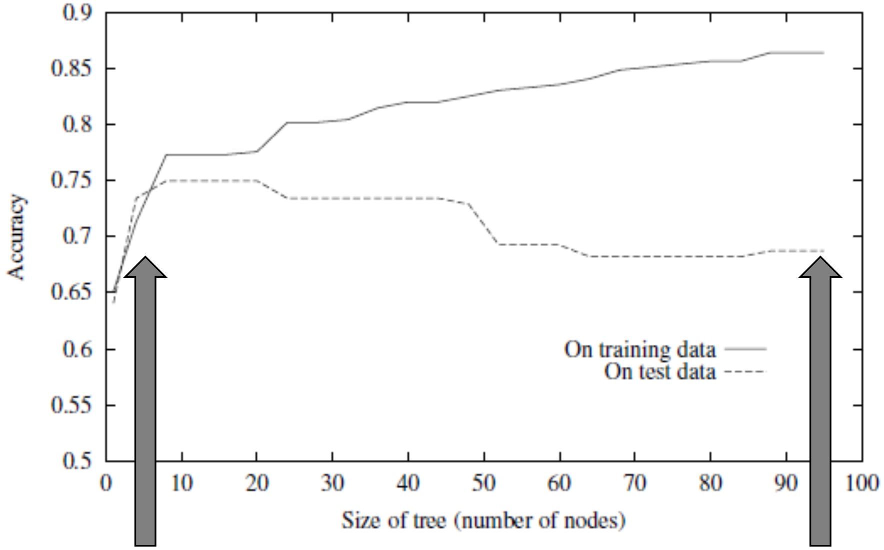
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No
D15	Sunny	Hot	Normal	Strong	No



Underfitting & Overfitting

- The training error can be decreased by increasing the model complexity
- But, a complex, tailored to the training data model, will also have a high generalization error

How the error in both training and test data evolves with the tree complexity



The model has yet to learn the true structure from the training data.

Model underfitting

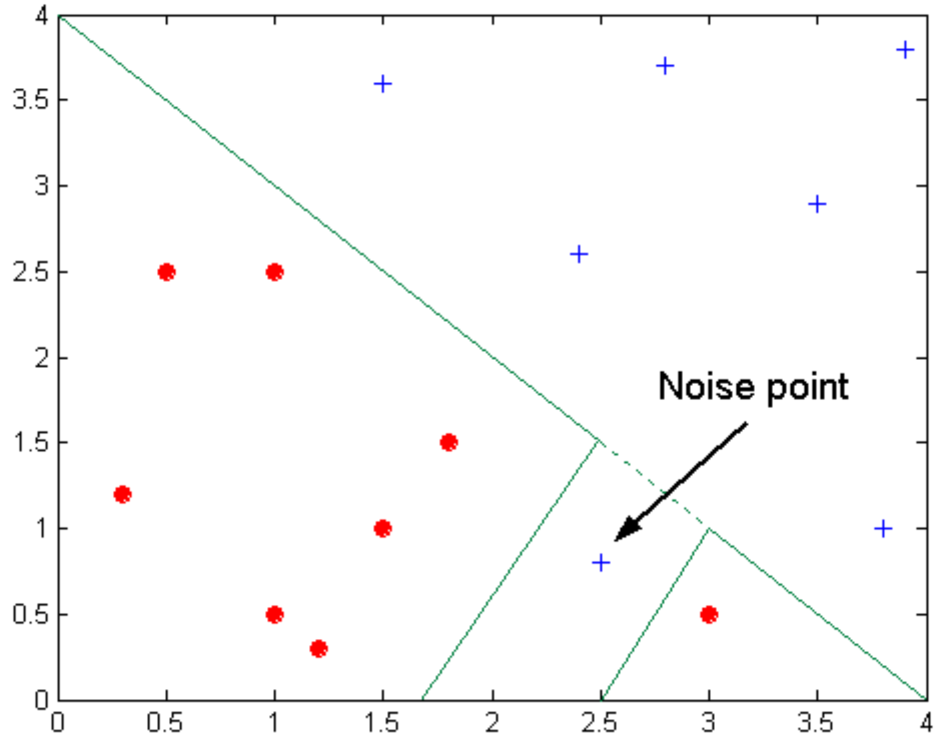
Model overfitting

The model overspecializes to the training data

Potential causes of model overfitting

- Overfitting due to presence of noise
- Overfitting due to lack of representative samples

Overfitting due to presence of noise

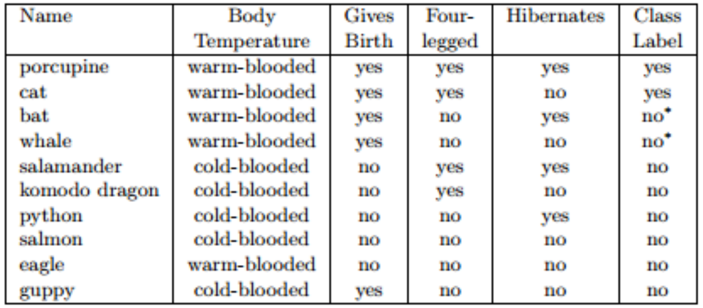


The decision boundary is distorted by the noise point.

Overfitting due to presence of noise – an example

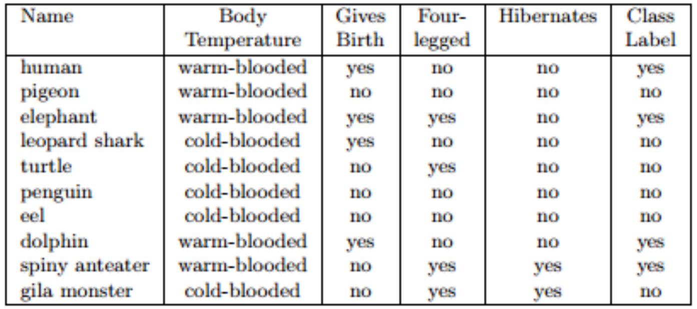
Training set

(stands for missclassified instances)*

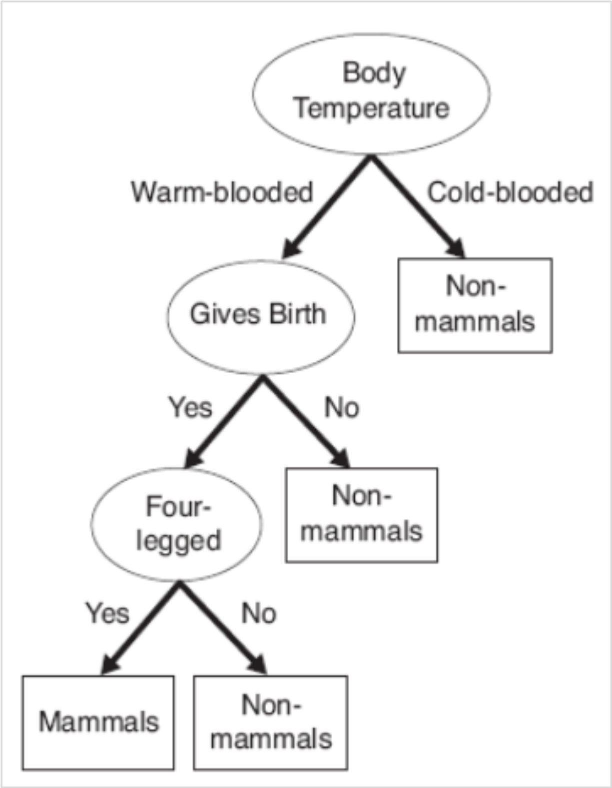


Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

Test set



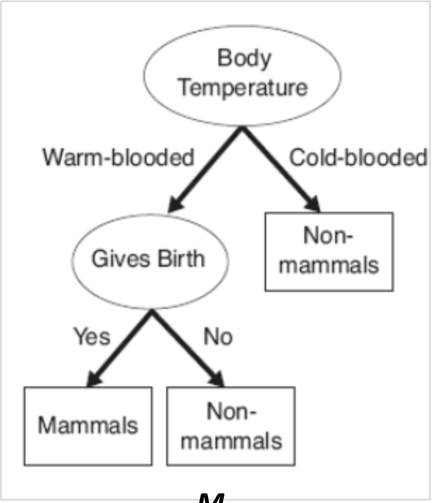
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no



***M*₁**

Training error: 0

Test error: 30%

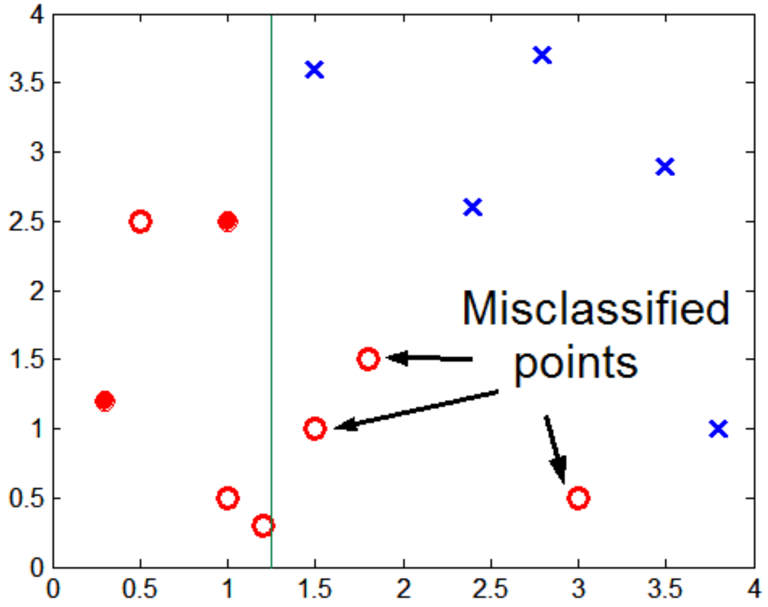


***M*₂**

Training error: 20%

Test error: 10%

Overfitting due to lack of representative samples



- Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region
  - Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

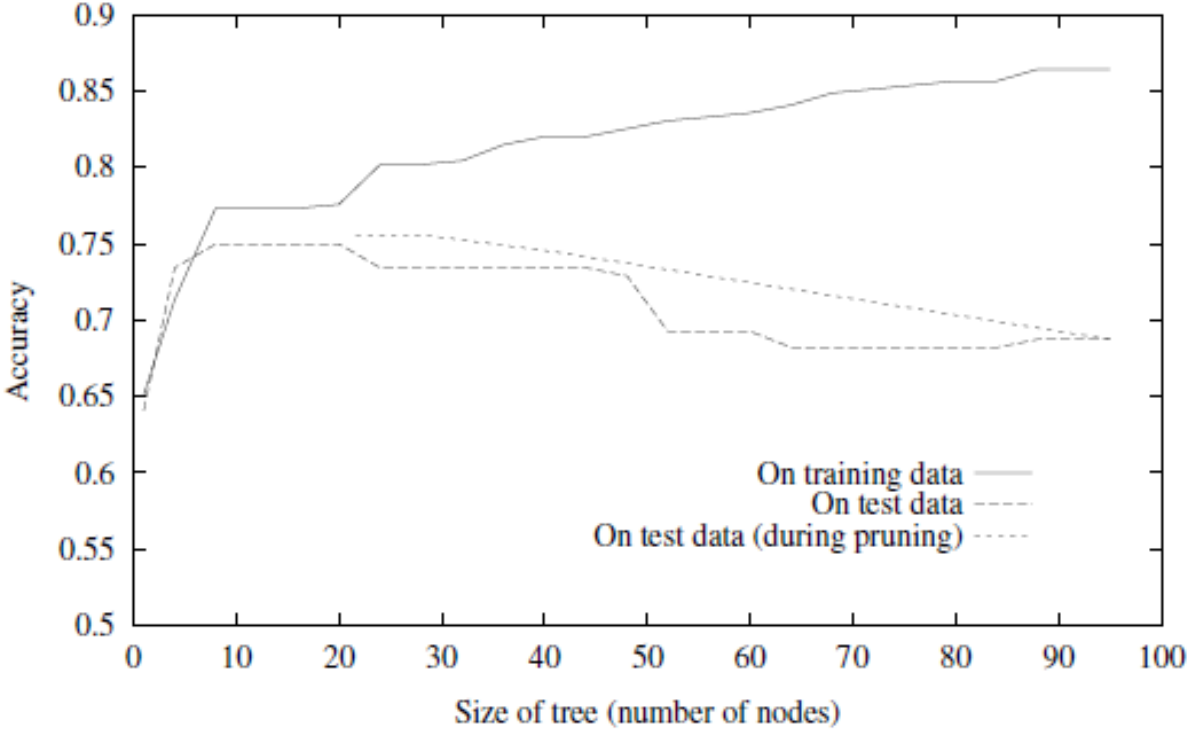
Avoiding overfitting in decision trees

- Overfitting results in decision trees that are more complex than necessary
- The training error no longer provides a good estimate of how well the tree will perform on previously unseen records

  → Generalization error is very important
- Two approaches to avoid overfitting in decision trees
  - Pre-pruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Post-pruning: Remove decision nodes from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide whether pruning node is effective

Effect of prunning

- How the error in both training and test data evolves with the tree complexity; with and without pruning



Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

Lazy vs Eager learners

- Eager learners
  - Construct a classification model (based on a training set)
  - Learned models are ready and eager to classify previously unseen instances
  - e.g., decision trees
- Lazy learners
  - Simply store training data and wait until a previously unknown instance arrives
  - No model is constructed.
  - known also as instance based learners, because they store the training set
  - e.g., k-NN classifier

Eager learners

- Do lot of work on training data
- Do less work on classifying new instances

Lazy learners

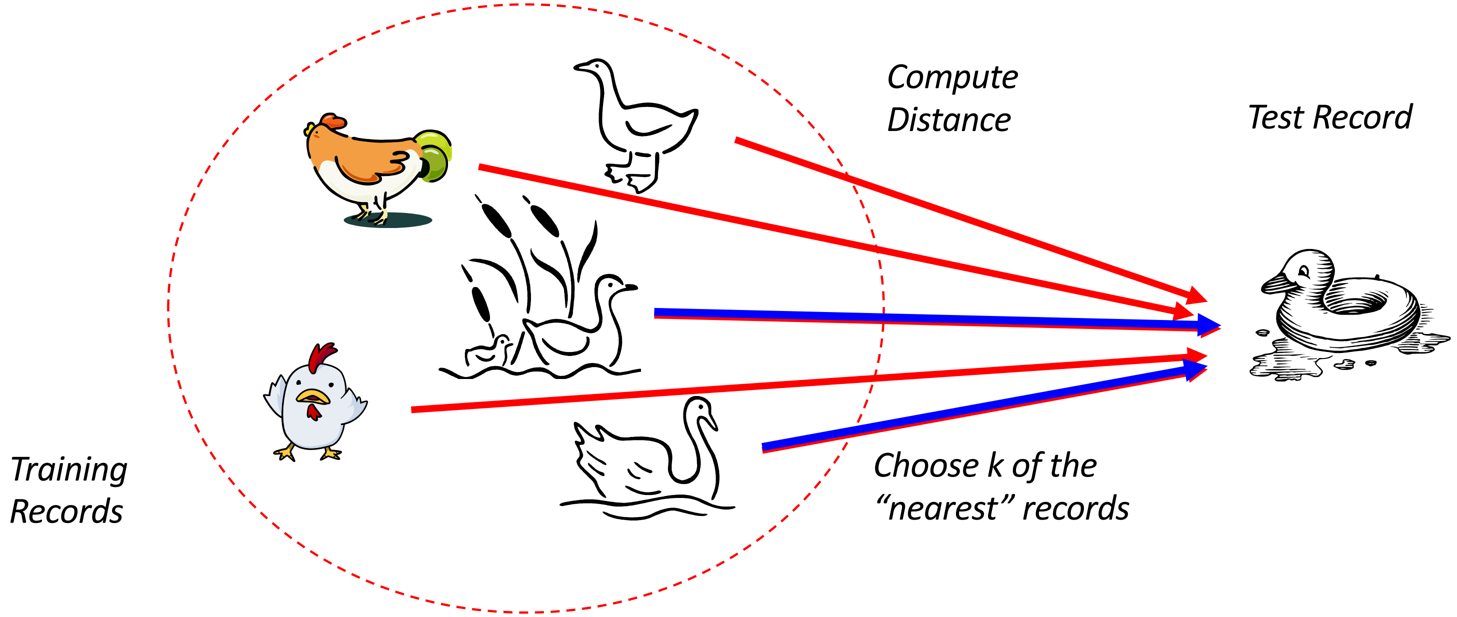
- Do less work on training data
- Do more work on classifying new instances

Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

Lazy learners/ Instance-based learners: k-Nearest Neighbor classifiers

- Nearest-neighbor classifiers compare a given unknown instance with training tuples that are similar to it
  - Basic idea: *If it walks like a duck, quacks like a duck, then it's probably a duck*



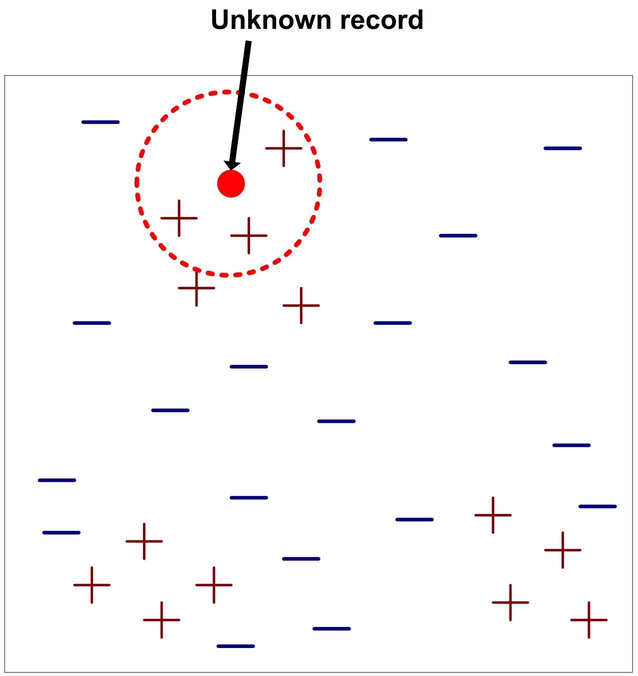
k-Nearest Neighbor classifiers

Input:

- A training set D (with known class labels)
- A distance metric to compute the distance between two instances
- The number of neighbors k

Method: Given a new unknown instance *X*

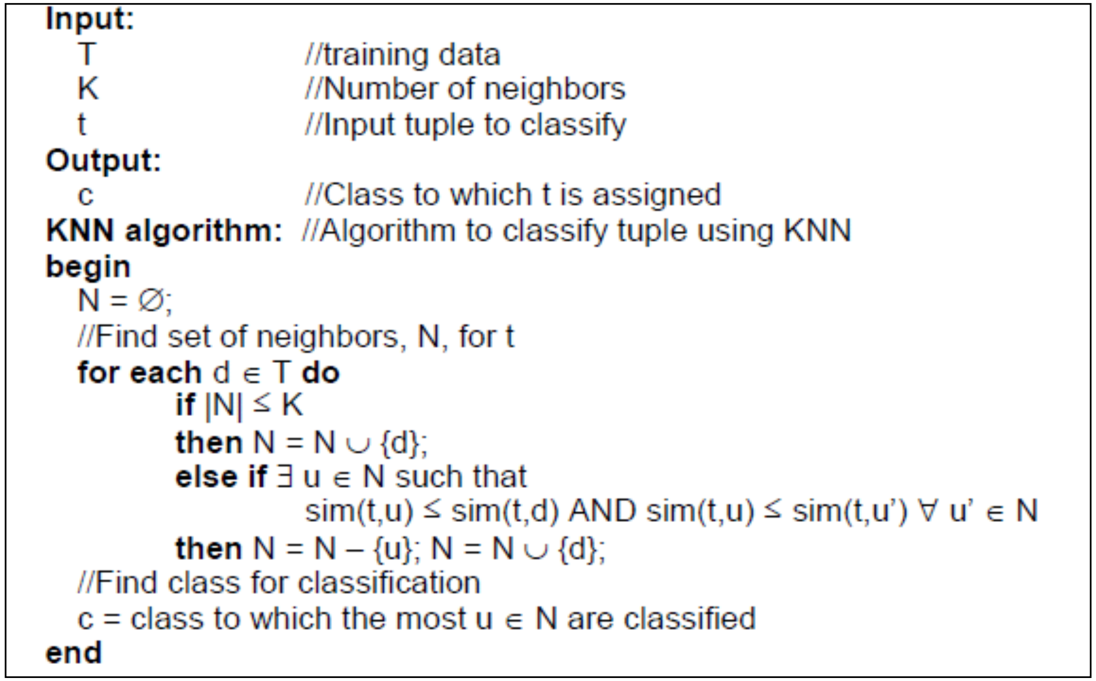
- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



It requires O(|D|) for each new instance

kNN algorithm

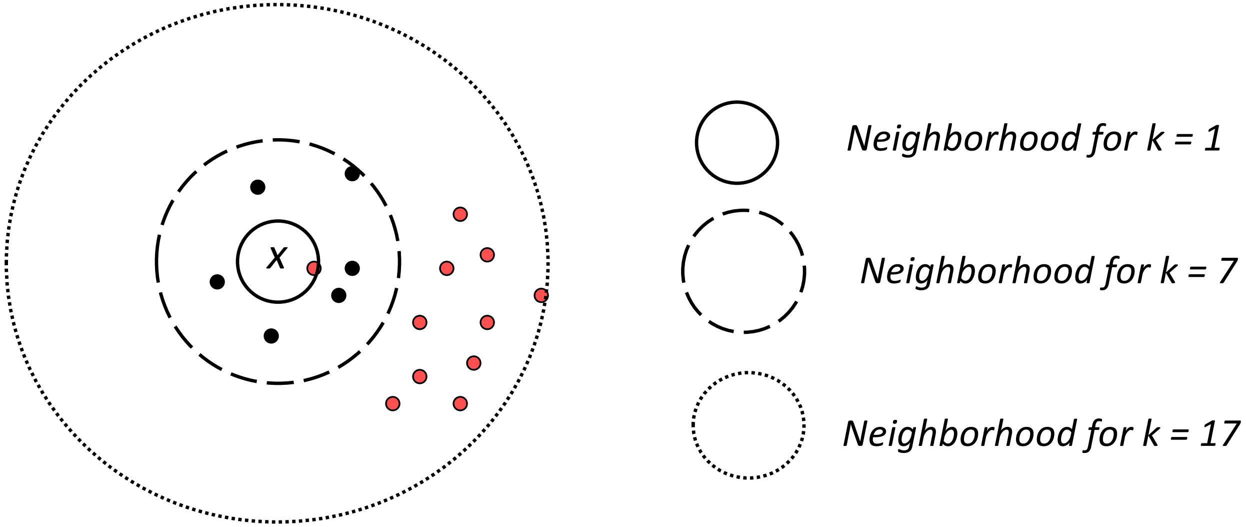
- Pseudocode:



```
Input:
  T                 //training data
  K                 //Number of neighbors
  t                 //Input tuple to classify
Output:
  c                 //Class to which t is assigned
KNN algorithm:      //Algorithm to classify tuple using KNN
begin
  $N = \emptyset$;
  //Find set of neighbors, N, for t
  for each $d \in T$ do
        if $|N| \leq K$
        then $N = N \cup \{d\}$;
        else if $\exists\, u \in N$ such that
                $sim(t,u) \leq sim(t,d)$ AND $sim(t,u) \leq sim(t,u') \;\forall\, u' \in N$
        then $N = N - \{u\}$; $N = N \cup \{d\}$;
  //Find class for classification
  c = class to which the most $u \in N$ are classified
end
```

Definition of k nearest neighbors

- too small k: high sensitivity to outliers
- too large k: many objects from other classes in the resulting neighborhood
- average k: highest classification accuracy, usually 1 << k < 10



x: unknown instance

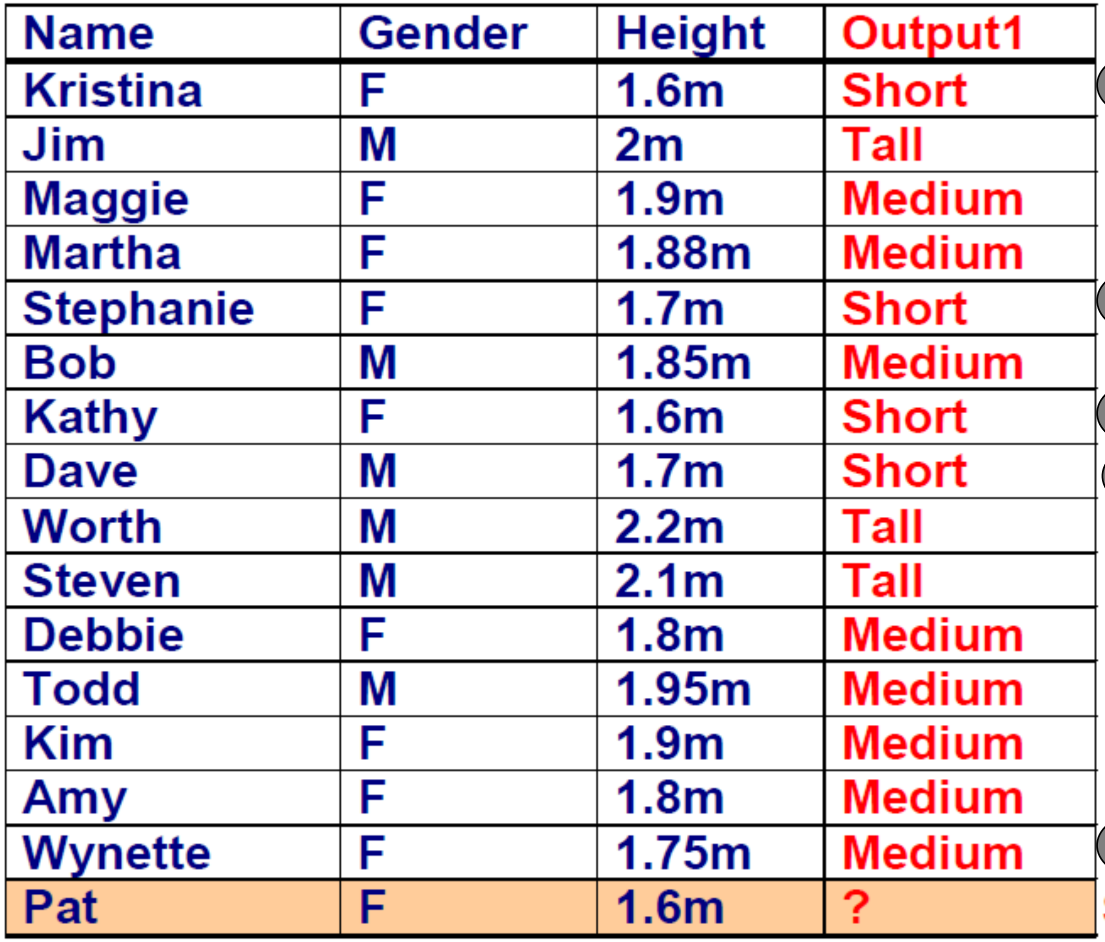
Nearest neighbor classification

- "Closeness" is defined in terms of a distance metric
  - e.g. Euclidean distance

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

- The k-nearest neighbors are selected among the training set
- The class of the unknown instance X is determined from the neighbor list
  - If k=1, the class is that of the closest instance
  - Majority voting: take the majority vote of class labels among the neighbors
    - Each neighbor has the same impact on the classification
    - The algorithm is sensitive to the choice of k
  - Weighted voting: Weigh the vote of each neighbor according to its distance from the unknown instance
    - weight factor, $w = 1/d^2$

Nearest neighbor classification: example



Name	Gender	Height	Output1	
Kristina	F	1.6m	Short	1
Jim	M	2m	Tall	
Maggie	F	1.9m	Medium	
Martha	F	1.88m	Medium	
Stephanie	F	1.7m	Short	3
Bob	M	1.85m	Medium	
Kathy	F	1.6m	Short	2
Dave	M	1.7m	Short	4
Worth	M	2.2m	Tall	
Steven	M	2.1m	Tall	
Debbie	F	1.8m	Medium	
Todd	M	1.95m	Medium	
Kim	F	1.9m	Medium	
Amy	F	1.8m	Medium	
Wynette	F	1.75m	Medium	5
Pat	F	1.6m	?	Short

Nearest neighbor classification issues I

- Different attributes have different ranges
  - e.g., height in [1.5m-1.8m]; income in [$10K -$1M]
  - Distance measures might be dominated by one of the attributes
  - Solution: normalization

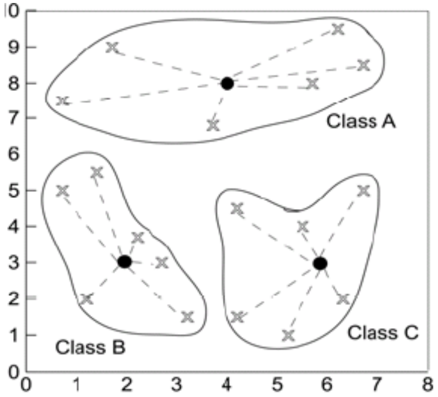
- k-NN classifiers are lazy learners
  - No model is built explicitly, like in eager learners such as decision trees
  - Classifying unknown records is relatively expensive
  - Possible solutions:
    - Use index structures to speed up the nearest neighbors computation
    - Partial distance computation based on a subset of attributes

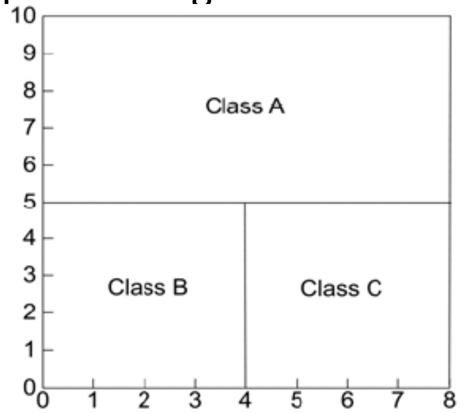
Nearest neighbor classification issues II

- The “curse of dimensionality”
  - Ratio of (D_{max_d} – D_{min_d}) to D_{min_d} converges to zero with increasing dimensionality *d*
    - D_{max_d}: distance to the nearest neighbor in the d-dimensional space
    - D_{min_d}: distance to the farthest neighbor in the d-dimensional space
  - This implies that:
    - all points tend to be almost equidistant from each other in high dimensional spaces
    - the distances between points cannot be used to differentiate between them
  - Possible solutions:
    - Dimensionality reduction (e.g., PCA)
    - Work with a subset of dimensions instead of the complete feature space

k-NN classifiers: overview

- (+-) Lazy learners: Do not require model building , but testing is more expensive
- (-) Classification is based on local information in contrast to e.g. DTs that try to find a global model that fits the entire input space: Susceptible to noise
- (+) Incremental classifiers
- (-) The choice of distance function and k is important
- (+) Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries, in contrary to e.g. decision trees that result in axis parallel hyper rectangles





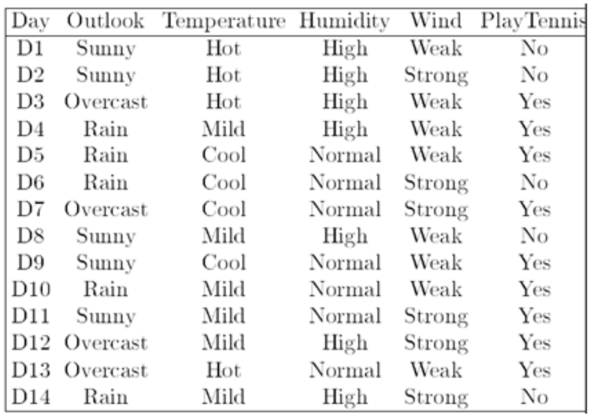
Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

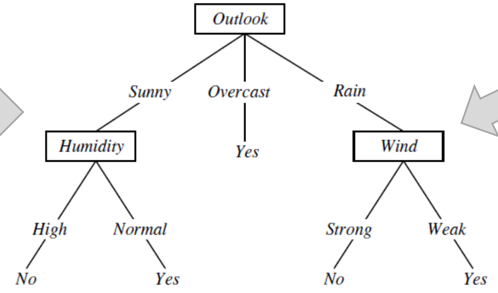
True vs predicted class labels

- The quality of a classifier is evaluated over a *test set*, different from the training set
  - For each instance in the test set, we know its true class label
  - Compare the predicted class label (by the classifier) with the true class of the test instances

Training set



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Test set

true class label

Day	Outlook	Temperature	Humidity	Wind	Play Teniss	Prediction
D16	Overcast	Cool	Normal	Weak	Yes	Yes
D17	Overcast	High	Normal	Weak	Yes	No
D18	Sunny	Hot	Normal	Weak	No	No
D19	Overcast	Cool	Normal	Weak	No	Yes

predicted class labels

Confusion matrix

- Terminology
  - Positive tuples: tuples of the main class of interest (e.g., “Play tennis = yes”)
  - Negative tuples: all other tuples
- A useful tool for analyzing how well a classifier performs is the *confusion matrix*
  - For an *m*-class problem, the matrix is of size *mxm*
- An example of a matrix for a 2-class problem:

Predicted class

Actual/ true class		yes	no	totals
	yes	TP (true positive)	FN (false negative)	P
	no	FP(false positive)	TN (true negative)	N
	Totals	P’	N’	

Classifier evaluation measures

- Accuracy
- Error rate
- Sensitivity
- Specificity
- Precision
- Recall
- F–measure
- F_β-measure
- ...

Classifier evaluation measures 1/3

Predicted class

Actual class

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P'	N'	

- Accuracy/ Recognition rate:

% of test set instances correctly classified

$$accuracy(M) = \frac{TP + TN}{P + N}$$

Predicted class

Actual class

classes	buy_computer = yes	buy_computer = no	total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
total	7366	2634	10000

➔Accuracy(M)=95.42%

- Error rate/ Missclassification rate: error_rate(M)=1-accuracy(M)

$$\text{error\_rate}(M) = \frac{FP + FN}{P + N}$$

➔Error_rate(M)=4.58%

Limitations of accuracy and error rate

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

!!! Accuracy and error rate are more effective when the class distribution is relatively *balanced*

Classifier evaluation measures 2/3

Predicted class

Actual class

	C$_1$	C$_2$	totals
C$_1$	TP (true positive)	FN (false negative)	P
C$_2$	FP(false positive)	TN (true negative)	N
Totals	P’	N’	

If classes are *imbalanced*:

- Sensitivity/ True positive rate/ recall:

  % of positive tuples that are correctly recognized

$$sensitivity(M) = \frac{TP}{P}$$

- Specificity/ True negative rate : % of negative tuples that are correctly recognized

$$specificity(M) = \frac{TN}{N}$$

Predicted class

Actual class

classes	buy_computer = yes	buy_computer = no	total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
total	7366	2634	10000

➔Accuracy(M)=95.42%

➔sensitivity(M)=99.34%

➔specificity(M)=86.27%

Classifier evaluation measures 3/3

- Precision: % of tuples labeled as positive which are actually positive

$$precision(M) = \frac{TP}{TP + FP}$$

- Recall: % of positive tuples labeled as positive

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

  - Precision biased towards TP and FP
  - Recall biased towards TP and FN
  - Higher precision → less FP
  - Higher recall → less FN

Predicted class

Actual class

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P'	N'	

Recall the definition of precision/recall in IR:

- *Precision: % of selected items that are correct*
- *Recall: % of correct items that are selected*

Predicted class

Actual class

classes	buy_computer = yes	buy_computer = no	total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
total	7366	2634	10000

→precision(M)=94.41%

→recall(M)=99.34%

Classifier evaluation measures 3/3

- F-measure/ F_1 score/F-score combines both

$$F(M) = \frac{2 * precision(M) * recall(M)}{precision(M) + recall(M)}$$

It is the harmonic mean of precision and recall

Predicted class

Actual class

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P'	N'	

More on harmonic mean: http://mathworld.wolfram.com/HarmonicMean.html

- F_β-measure is a weighted measure of precision and recall

$$F_\beta(M) = \frac{(1+\beta^2) * precision(M) * recall(M)}{\beta^2 * precision(M) + recall(M)}$$

Common values for β:

- $\beta=1 \rightarrow F_1$
- $\beta=0.5$

- For our example, F(M)=2*94.41%*99.34%/(94.41%+99.34%)=96.81%

Evaluation setup

- How to create the training and test sets out of a dataset?
  - We don’t want to make unreasonable assumptions about our population
- Many approaches
  - Holdout
  - Cross-validation
  - Bootstrap
  - ….

Evaluation setup 1/5

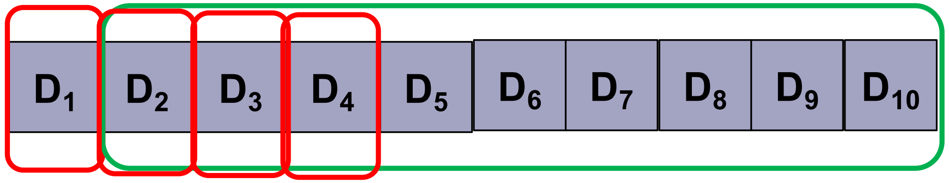
- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - (+) It takes no longer to compute
  - (-) it depends on how data are divided

Training set
Test set

- Random sampling: a variation of holdout
  - Repeat holdout *k* times, accuracy is the *avg* accuracy obtained

Evaluation setup 2/5

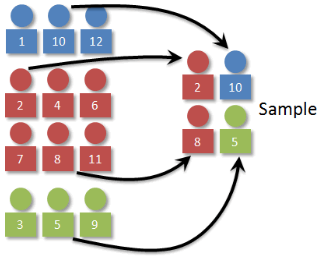
- Cross-validation (*k*-fold cross validation, k = 10 usually)
  - Randomly partition the data into *k mutually exclusive* subsets D_1, …, D_k each approximately equal size
  - Training and testing is performed k times
    - At the *i*-th iteration, use D_i as test set and rest as training set



  - Each point is in a test set 1 time and in a training set k-1 times
  - Accuracy is the avg accuracy over all iterations
  - (+) Does not rely so much on how data are divided
  - (-) The algorithm should re-run from scratch k times
- Leave-one-out: k-folds with k = #of tuples, so only one sample is used as a test set at a time;
    - for small sized data
- Stratified cross-validation: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data
    - Stratified 10 fold cross-validation is recommended!!!

Evaluation setup 3/5

- Stratified sampling vs random sampling
  - Stratified sampling creates a mini-reproduction of the population in terms of the class labels. E.g., if 25% of the population belongs to the class “blue”, 25% to class “green” and 50% to class “red” then 25% of the sample is drawn randomly from class “blue”, 25% from class “green” and 50% from class “red”.



Source: https://faculty.elgin.edu/dkernler/statistics/ch01/images/strata-sample.gif

- Stratified cross-validation: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data
  - Stratified 10 fold cross-validation is recommended!!!

Evaluation setup 4/5

- Bootstrap: Samples the given training data uniformly with replacement
  - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
  - Works well with small data sets
- Several boostrap methods, and a common one is .632 boostrap
  - Suppose we are given a data set of #d tuples. The data set is sampled #d times, with replacement, resulting in a training set of #d samples (known also as *bootstrap sample*):
  - The data tuples that did not make it into the training set end up forming the test set.
  - Each sample has a probability 1/d of being selected and (1-1/d) of not being chosen. We repeat d times, so the probability for a tuple to not be chosen during the whole period is (1-1/d)^d.
    - For large d: $\left(1-\frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$
  - So on average, 36.8% of the tuples will not be selected for training and thereby end up in the test set; the remaining 63.2% will form the train test

Evaluation setup 5/5

- Repeat the sampling procedure k times → k bootstrap datasets
- Report the overall accuracy of the model:

$$acc_{boot}(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times acc(M_i)_{testSet_i} + 0.368 \times acc(M_i)_{train\_set})$$

Accuracy of the model obtained by bootstrap sample i when it is applied on test set i.

Accuracy of the model obtained by bootstrap sample i when it is applied over all labeled data

Evaluation summary

- Evaluation measures
  - accuracy, error rate, sensitivity, specificity, precision, F-score, F_{β}…
- Train – test splitting
  - Holdout, cross-validation, bootstrap,…
- Other parameters
  - Speed (model building time, model testing time)
  - Robustness to noise, outliers and missing values
  - Scalability for large data sets
  - Interpretability (by humans)

Outline

- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers
- Things you should know from this lecture

Things you should know from this lecture

- Decision tree classifiers
  - Attribute selection measures
  - Algorithm for decision tree induction
- Lazy vs Eager classifiers
- kNN classifiers