1

Decision Trees

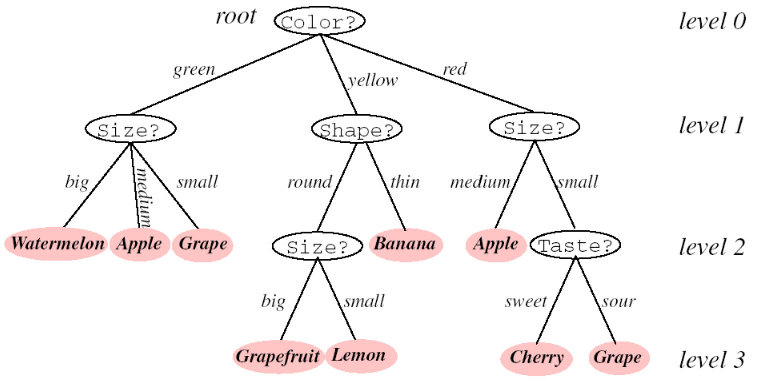
2

Decision Trees: Introduction

- Classifiers
  - Supervised Classifiers
    - Linear Classifiers
      - Perceptron, Least Squares Methods
      - Linear SVM
    - Nonlinear Classifiers
      - Part I: Multi Layer Neural Networks
      - Part II: Pol. Class., RBF, Nonlinear SVM
    - **Nonmetric Methods - Decision Trees**
    - AdaBoost
  - Unsupervised Classifiers

Decision Trees: Introduction

Example: Learning to classify fruits



Note, that same attributes (inner nodes) and class leafs (outer nodes) can appear in different places in the tree.

Decision Trees: Agenda

- **Definition**
- Mechanism
  - Splitting Functions
  - Hypothesis Space and Bias
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - Avoiding overfitting through pruning
- Ensemble Methods and Random Forests
- Application

Decision Trees: Definition

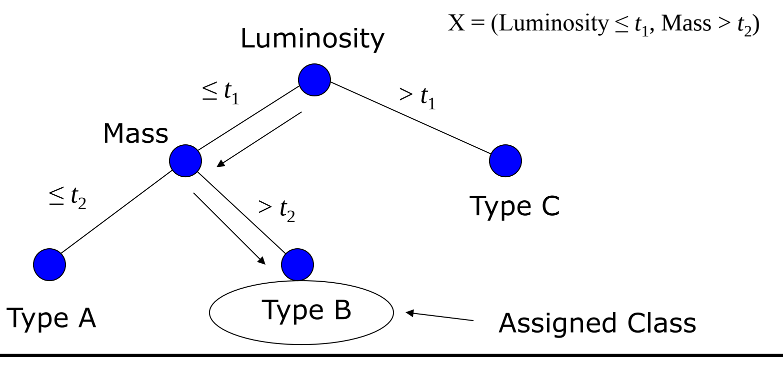
- **Decision Tree learning:** algorithm approximates a target concept using a tree representation, where each internal node corresponds to an attribute, and every terminal node corresponds to a class.

- **Two types of nodes:**
  - **Internal node:** Splits into different branches according to the different values the corresponding attribute can take.
  - **Terminal Node** (Leaf)**:** Decides the class assigned to the example.

Classifying Examples

Classification of an example X:

1. Start at the root of the tree.
2. Check the value of that attribute on X. Follow the branch corresponding to that value and jump to the next node.
3. Continue until a terminal node is reached.
4. Take that class as the best prediction.



7

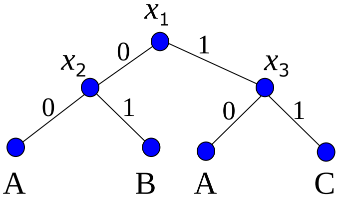
Representation

- Decision trees adopt a DNF (Disjunctive Normal Form) representation.
- Every branch from the root of the tree to a terminal node with a fixed class is a conjunction of attribute values.
- Different branches ending in that class form a disjunction.

  ➔ The axioms from the logic can be used, for generation and optimizing the trees. E.g. each logic expression can be transformed to a DNF

  ➔ Each knowledge represented as combination of logical statements (if … then … and … or …) can be modeled by a decision tree.

For class A:
($\sim x_1$ & $\sim$x2) OR (x_1 & $\sim$x3)



8

Appropriate Problems for Decision Trees

- Attributes are both numeric and nominal.
- Target function takes on a discrete number of values.
- A DNF representation is effective in representing the target concept.
- Training Data may have errors.
- Some examples may have missing attribute values.

Decision Trees: Agenda

- Definition
- **Mechanism**
  - **Splitting Functions**
  - Hypothesis Space and Bias
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - Avoiding overfitting through pruning
- Ensemble Methods and Random Forests
- Application

Mechanism

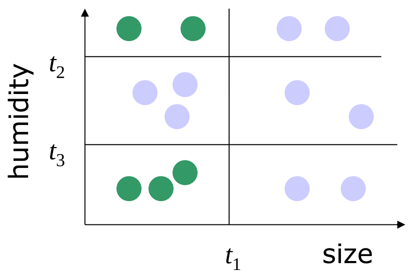
There are different ways to construct trees from data.
We will concentrate on the top-down, greedy search approach:

Basic idea:

1. Choose the best attribute a^* to place at the root of the tree.
2. Separate training set D into subsets $\{D_1, D_2, \ldots, D_k\}$ where each subset D_i contains examples having the same value for a^* .
3. Recursively apply the algorithm on each new subset until examples have the same class or there are few of them.

Illustration

Mushroom sample:



Class ω_P : poisonous

Class ω_N: not-poisonous

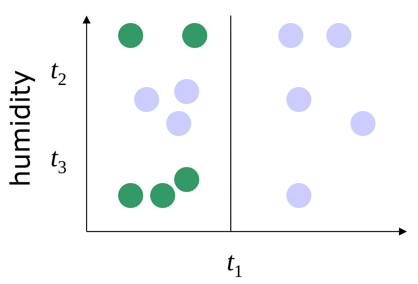
Attributes:

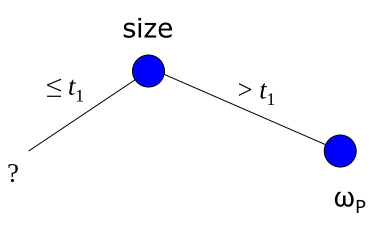
Size has two values: $> t_1$ or $\leq t_1$

Humidity has three values: $> t_2$, $(> t_3 \text{ and } \leq t_2)$, $\leq t_3$

Illustration

Suppose we choose size as the best attribute:



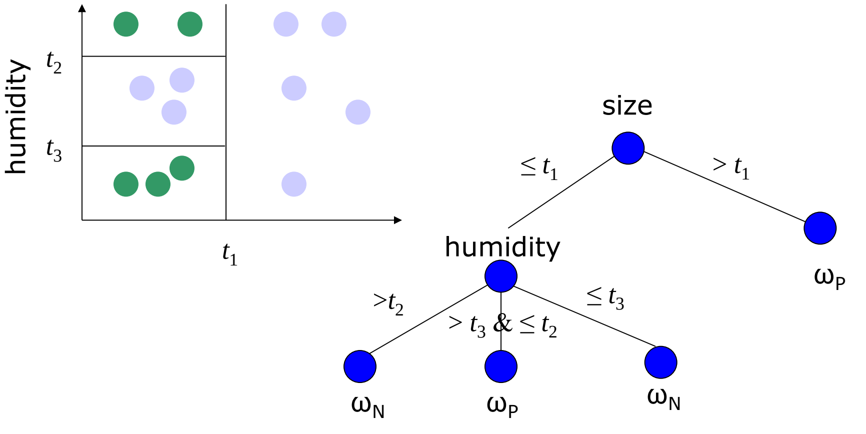


Class ω_P: poisonous

Class ω_N: not-poisonous

Illustration

Then humidity as the next best attribute:



Formal Mechanism

1. Create a root for the tree.
2. Stop-splitting rule:
   - If all examples are of the same class return that class.
   - If the number of examples is below a threshold or if no attributes are available return majority class.
3. Find the best attribute a^*.
4. For each possible range of values in S_v for a^* .
   - Add a branch below a^* labeled $v \in S_v$.
   - Recursively apply the algorithm to S_v.

Splitting Functions

What attribute is the best to split the data?

e.g. from information theory:

A measure of **impurity** or **entropy** for a subset S_v, associated with a node v is defined:

$$H(S_v) = -\sum_{i=1}^{M} P(\omega_i | S_v) \log_2 (P(\omega_i | S_v)),$$

where M is the number of classes (events), $P(\omega_i | S_v)$ denotes the probability that a vector in the subset S_v belongs to class ω_i .

Entropy

There are two possible complete events (classes) **A** and **B** (Example: flipping a biased coin).

P(A) = 1/2
P(B) = 1/2

=> $H(x)$ = 1 bit

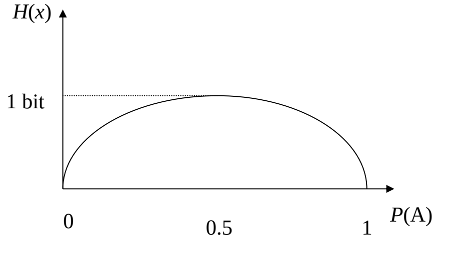
P(A) = 1/256,
P(B) = 255/256

=> $H(x)$ = 0.0369 bit

P(A) = 7/16
P(B) = 9/16

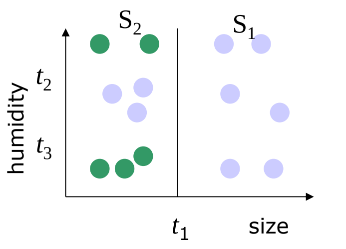
=> $H(x)$ = 0.989 bit

Entropy:



Splitting based on Entropy

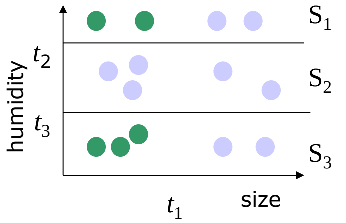
Mushroom sample:



Size divides the sample in two.

S_1 = { 6P, 0NP}
S_2 = { 3P, 5NP}

$H(S_1) = 0$
$H(S_2) = -(3/8)\log_2(3/8) - (5/8)\log_2(5/8)$
$= 0.9544$



Humidity divides the sample in three.

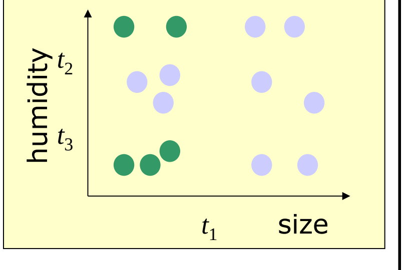
S_1 = { 2P, 2NP}
S_2 = { 5P, 0NP}
S_3 = { 2P, 3NP}

$H(S_1) = 1$ ➔ largest entropy ("*impurity"*)
$H(S_2) = 0$ ➔ no "*impurity"*
$H(S_3) = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5)$
$= 0.9710$ ➔ in between

Information Gain

Information gain IG
(decrease in node impurity)
over attribute a: $IG(a)$:

$$IG(a) = H(S) - \sum_v \frac{S_v}{S} H(S_v)$$



- $H(S)$ is the entropy of all samples.
- $H(S_v)$ is the entropy of one subsample after partitioning S based on all possible values of attribute a.
- $v = 1, \ldots, N$ (number of sub-nodes).

➔ The goal now becomes to adopt, from the set attributes, the one that performs the split leading to the highest IG.

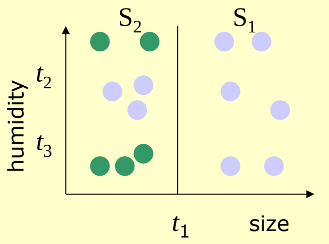
$$a^* = \arg\max_{a \in A} IG(a)$$

19

Example

$$IG(a) = H(S) - \sum_v \frac{S_v}{S} H(S_v)$$

H(S) = - (9/14)$\log_2$(9/14) - (5/14)$\log_2$(5/14) = 0.9403



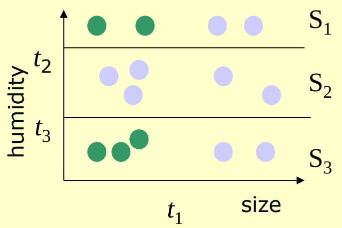
$H(S_1) = 0$

$H(S_2) = -(3/8)\log_2(3/8) - (5/8)\log_2(5/8) = 0.9544$

$S_1/S = 6/14$

$S_2/S = 8/14$

$IG(\text{size}) = 0.9403 - 0 - (8/14)\, 0.9544 = \underline{0.3949}$



$H(S_1) = 1$,

$H(S_2) = 0$

$H(S_3) = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.9710$

$S_1/S = 4/14$

$S_2/S = 5/14$

$S_3/S = 5/14$

$IG(\text{hum}) = 0.9403 - 4/14 - (5/14)\, 0.9710 = \underline{0.\ 3078}$

➔ a^* = size

20

Formal Mechanism

1. Create a root node for the tree.
2. Stop-splitting rule:
   - If all examples are of the same class return that class.
   - If the number of examples is below a threshold or if no attributes are available return majority class.
3. Compute the best attribute: $a^* = \arg\max_{a \in A} IG(a)$
4. For each possible range of values in S_v for a^*
   - Add a branch below a^* labeled $v \in S_v$.
   - Recursively apply the algorithm to S_v.

Decision Trees: Agenda

21

- Definition
- Mechanism
  - Splitting Functions
  - **Hypothesis Space and Bias**
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - Avoiding overfitting through pruning
- Ensemble Methods and Random Forests
- Application

Hypothesis Space

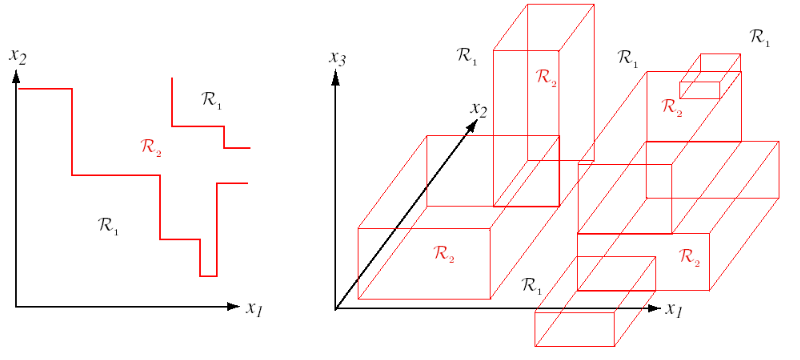
22

- We search over the hypothesis space of all possible decision trees.
- We keep only one hypothesis at a time, instead of having several (greedy search).
- We don't do backtracking in the search. We choose locally the best alternative and continue growing the tree.
- We prefer shorter trees than larger trees.
- We prefer trees where attributes with highest Information Gain are placed on the top.

Hypothesis Space

Decision Tress create decision boundaries with portions perpendicular to the feature axes.

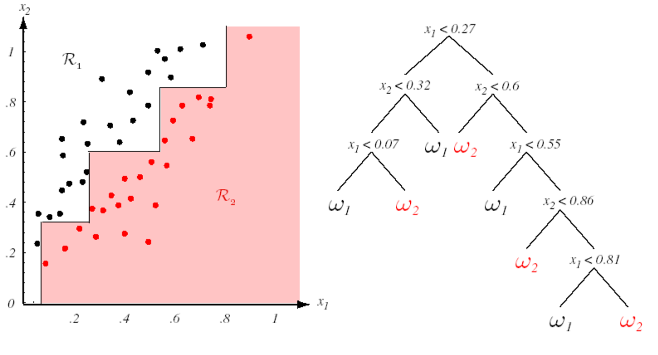
With a sufficiently large tree, any decision boundary can be approximated arbitrarily well in this way.



Hypothesis Space

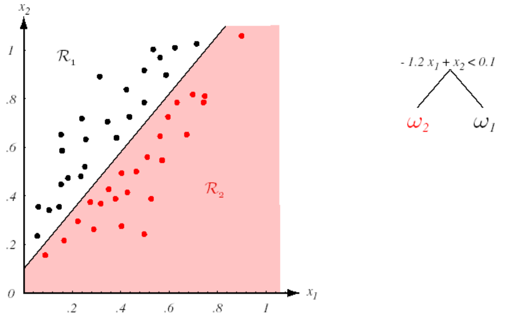
If the class of node decisions does not match the form of the training data, a very complicated decision tree will result.

Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction.



Hypothesis Space

If, however, "proper" decision forms are used (here, linear combinations of the features), the tree can be quite simple.



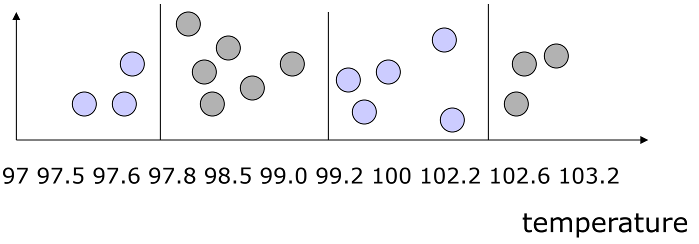
Decision Trees: Agenda

- Definition
- Mechanism
  - Splitting Functions
  - Hypothesis Space and Bias
- **Issues in Decision-Tree Learning**
  - **Numeric and missing attributes**
  - Avoiding overfitting through pruning
- Ensemble Methods and Random Forests
- Application

Discretizing Continuous Attributes

Example: attribute temperature.

1) Order all values in the training set.
2) Consider only those cut points where there is a change of class.
3) Choose the cut point that maximizes information gain.



Missing Attribute Values

Example: $\mathbf{X}$ = (luminosity > T_1, mass = ?)

We are at a node n in the decision tree.
Different approaches:

1) Assign the most common value for that attribute in node n.
2) Assign the most common value in n among examples with the same classification as $\mathbf{X}$.
3) Assign a probability to each value of the attribute based on the frequency of those values in node $\boldsymbol{n}$. Each fraction is propagated down the tree.

Decision Trees: Agenda

29

- Definition
- Mechanism
  - Splitting Functions
  - Hypothesis Space and Bias
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - **Avoiding overfitting through pruning**
- Ensemble Methods and Random Forests
- Application

Short vs. Long Hypotheses

30

❖ We described a top-down, greedy approach to construct decision trees denotes a preference of short hypotheses over long hypotheses.

➔ Why is this the right thing to do?

Occam's Razor:
Prefer the simplest hypothesis that fits the data.

Back since William of Occam (1320).
Great debate in the philosophy of science.

Issues in Decision Tree Learning

Practical issues while building a decision tree can be enumerated as follows:

1) How deep should the tree be?
2) How do we handle continuous attributes?
3) What is a good splitting function?
4) What happens when attribute values are missing?
5) How do we improve the computational efficiency?

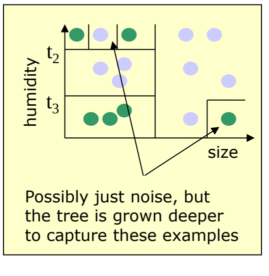
Issues in Decision Tree Learning

1) How deep should the tree be?

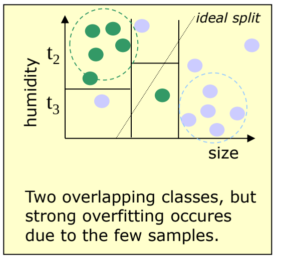
A tree *over fits* the data if we let it grow deep enough so that it begins to capture "aberrations" in the data that harm the predictive power on unseen examples:

Causes?

a) Random errors or noise: Examples have incorrect class label or incorrect attribute values.



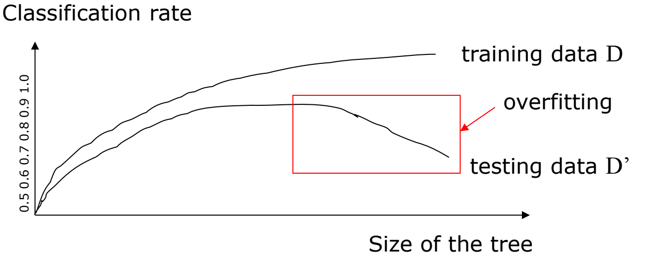
b) Coincidental patterns: Examples seem to deviate from a pattern due to the small size of the sample.



Overfitting the Data: Definition

33

Assume a hypothesis space H. We say a hypothesis h in H overfits a dataset D if there is another hypothesis h' in H where h has better classification accuracy than h' on D but worse classification accuracy than h' on additional set D'.



➔ Overfitting is a serious problem that can cause strong performance degradation.

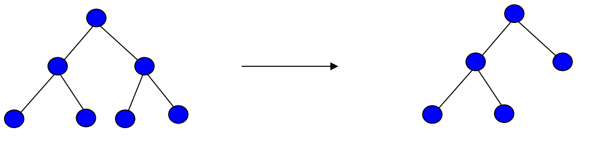
Solutions for Overfitting the Data

34

There are two main classes of solutions:

1) Stop the tree early before it begins to overfit the data.
   - In practice this solution is hard to implement because it is not clear what is a good stopping point.

2) Grow the tree until the algorithm stops even if the overfitting problem shows up. Then prune the tree as a post-processing step.
   + This method has found great popularity in the machine learning community.



a) Grow the tree to learn the training data

b) Prune tree to avoid overfitting the data

Pruning

Characteristics of pruning methods

- Use of a validation set
- Tends to under- or overprune
- Bottom-up or top-down tree traversal
- Computational complexity

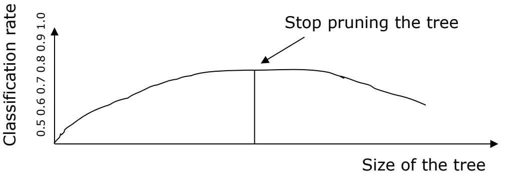
Three exemplary pruning approaches

A. Reduced Error Pruning
B. Error-Based Pruning
C. Rule Post-Pruning

A. Reduced Error Pruning

Main Idea

Remove nodes of the tree as long as the classification rate on the validation data increases.



Formal Mechanism

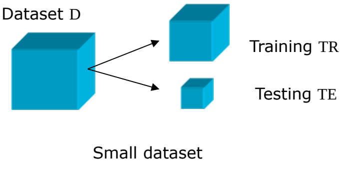
1) Consider all internal nodes in the tree.
2) For each node check if removing it (along with the subtree below it) and assigning the most common class to it improves accuracy on the validation set.
3) Pick the node n^* that yields the best performance and prune its subtree.
4) Go back to (2) until no more improvements are possible.

A. Reduced Error Pruning

Advantages:
- Computational complexity is linear in the number of inner nodes.
- Leads to the smallest version of the most accurate subtree with respect to the validation set.

Disadvantages:
- All evidence of the training set is neglected during the pruning process.
- Tends to overprune if validation set is not large enough.
- If the original data set is small, separating examples away for validation may result in a very small training set.

➔ **Threesfold Cross Validation**:
- share data in parts A, B and C
- train A,B against C; A,C against B and C,B against A.
- test on separate Test-Data

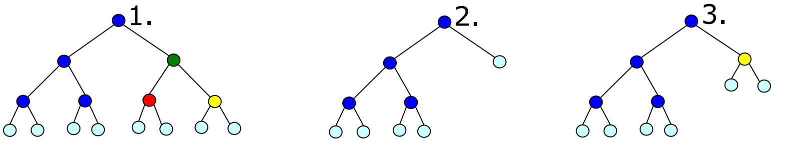


B. Error-Based Pruning

Core Idea

Estimate the error rate on unseen samples based on the training samples.

- Assume training errors are binomial distributed.
- Calculate the error rate on unseen samples as upper bound of confidence interval.
- Compare the errors at each inner node of:
  1. the subtree (sum of errors in all leaves),
  2. pruning the subtree,
  3. replacing the subtree (take subtree of the inner node with most frequent outcome)



B. Error-Based Pruning

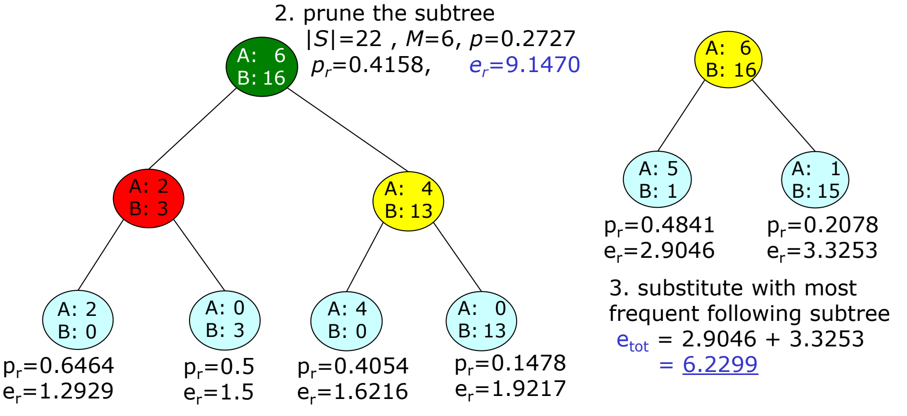
- p_r : upper bound of the confidence interval
- S : set of N samples reaching a node
- M : number of errors in a node using the majority class
- e_r : estimate the number of errors on unseen data as $e_r = p_r|S|$
- p : probability of an error in the node estimated as $p = |S|/M$.

Calculate p_r so that: $1\text{-}CF = P(p \leq p_r)$
Assuming the errors are binomial distributed the above solution is equivalent to solve for p_r in:

$$CF = \begin{cases} (1-p_r)^N & , for\ M = 0 \\ \sum_{i=0}^{M} \binom{N}{i} p_r^{\,i} (1-p_r)^{N-i} & , for\ M > 0 \end{cases}$$

Here $N = |S|$, the number of samples in the set and M, the number of errors made in the node. There exist a variety of algorithms to solve this equation for p_r (Matlab: *binofit(M,N,CF)*).

B. Error-Based Pruning



1. do not prune the subtree
e_{tot} = 1.2929 + 1.5 + 1.6216 + 1.9217 = 6.3362

CF = 25%

Choose to substitute the green inner node with the yellow inner node!!!

B. Error-Based Pruning

Advantages:
- Allows to remove „intermediate" tests wich appear useless.
- Has often a good performance in practice.

Disadvantages:
- The parameter *CF* determines if EBP over- or underprune.
- Strong assumption that errors are binomial distributed.
- Computationally less efficient than reduced error pruning.

C4.5 is an algorithm for decision trees that uses error-based pruning with *CF*=25%.

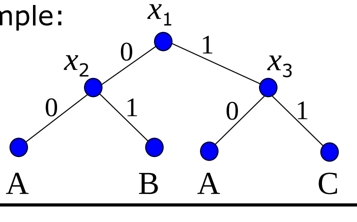
C. Rule Post-Pruning

Core Idea:

1) Convert the tree into a rule-based system.
2) Prune every single rules first by removing redundant conditions using propositional logic.
3) Sort rules by accuracy.

Advantages:
- ❖ The language is more expressive
- ❖ Improves on interpretability
- ❖ Pruning is more flexible
- ❖ In practice this method yields high accuracy performance

Example:



Rules:
~x1 & ~x2 -> Class A
~x1 & x2 -> Class B
x1 & ~x3 -> Class A
x1 & x3 -> Class C

↓

Possible rules for pruning (based on validation set):
~x1 -> Class A
~x1 & x2 -> Class B
~x3 -> Class A
x1 & x3 -> Class C

↓

Test the different rules and select the most efficient ones.

43

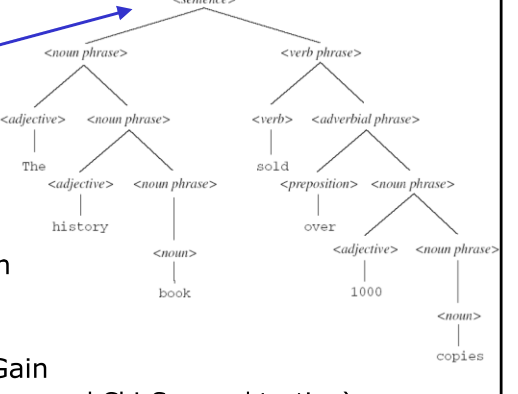
Summary

- The generalization performance is not as good as margin maximized classifiers, but
  - Computationally dramatically cheap!!!   (binary search!)
- Decision-tree induction is a popular approach to classification that enables us to interpret the output hypothesis.
  - Easy to understand,
  - Easy to implement,
  - Easy to use.
- The hypothesis space is powerful: all possible DNF formulas.
- Overfitting is an important issue in decision-tree induction. Different methods exist to avoid overfitting like reduced-error pruning and rule post-processing.
- Techniques exist to deal with continuous attributes and missing attribute values.

44

What we haven't discussed

- It's easy to have real-valued outputs too - these are called Regression Trees.
- Rule based Methods.
- Other trees, here derivation trees e.g. for definition of a grammar.



- Recognitions with Strings.
- Bayesian Decision Trees can take a different approach to preventing over-fitting.
- Alternatives to Information Gain for splitting nodes (MaxP-chance and Chi-Squared testing).

Decision Trees: Agenda

45

- Definition
- Mechanism
  - Splitting Functions
  - Hypothesis Space and Bias
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - Avoiding overfitting through pruning
- **Ensemble Methods and Random Forests**
- Application

Ensemble Methods

46

Main Idea

To increase the predictive performance of a base learning technique, ensemble methods combine the output of several learned models instead of learning a single model.

1. Use a base procedure (e.g. decision trees) and perturb the algorithm and/or the learning data to learn several models.
2. Combine the prediction (e.g. mean or majority prediction) of all learned models to the final prediction of the ensemble.

Some variants of ensemble methods used with decision trees are **bagging**, **boosting** and **random-sub-space** methods.

Ensemble Methods

Bagging: (bootstrap aggregating)

- For each classifier select randomly *n* training samples from the training set.
- Better accuracy than boosting when data is noisy.
- Classifiers can be learned in parallel.

Boosting

- Adjust weights for each training sample when a new classifier is trainined.
- Good accuracy but susceptible to noise.
- Classifiers can not be learned in parallel.

Random subspace

- For each classifier select randomly *n* attributes of all available.
- Accuracy lies between bagging and boosting.
- Poor accuracy if attributes are uncorrelated.

Random Forests

Main Idea

Combine the response of several decision trees to improve accuracy and generalization.

Random forests belong to the ensemble methods. The base procedure of learning a decision tree is perturbed using bagging and/or random subspace methods. Further possibilities of perturbing the learning of a decision tree are:

- Randomly generate decision functions when searching for the best split.
- Use only a subset of the training data to choose the best split.
- Select one of the *n*-best decision functions and not the best.

Advantages of randomization:

- Handle larger data sets
- Search larger function space

Random Forests

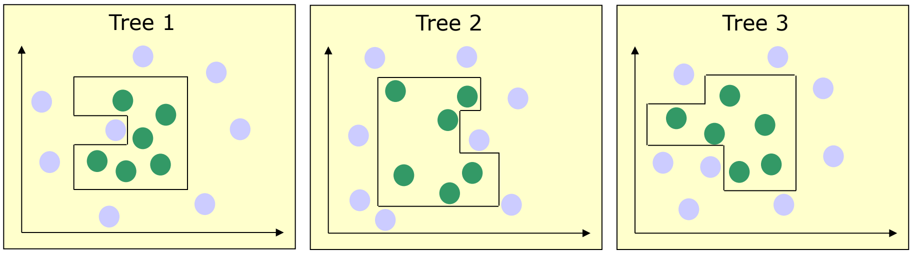
Formal Learning

1) Randomly select the training data for one tree.
2) Learn the tree based on the training data.
   a) Create a root node for the tree.
   b) If a stopping rule holds do not split the samples.
   c) Generate randomly a set of decision functions.
   d) Select the best decision function using the samples reaching the node.
   e) Assigning a new node to each outcome of the best function.
   f) Recursively apply b), c), d) and e) to each node.
3) Repeat 1) and 2) for every tree in the forest.

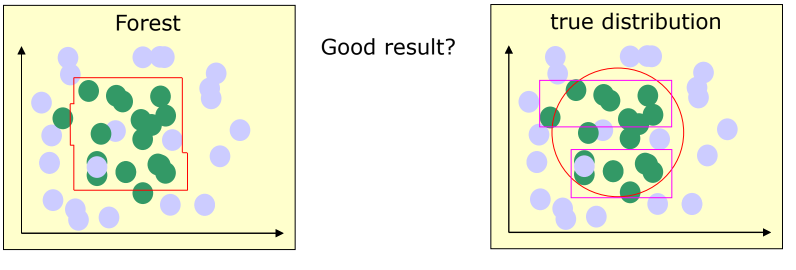
Formal Application

1) Recursively classify a new sample with each tree.
2) Return the class predicted by the majority of the trees.

Random Forests



... combine using majority class ...



51

Decision Trees: Agenda

- Definition
- Mechanism
  - Splitting Functions
  - Hypothesis Space and Bias
- Issues in Decision-Tree Learning
  - Numeric and missing attributes
  - Avoiding overfitting through pruning
- Ensemble Methods and Random Forests
- **Application**
  - C4.5, See5, CART
  - Spam, Expert Systems, Multiclass Classifiers

52

Decision Trees: Application

Spam detection (by Trevor Hastie, Stanford University)

goal: predict whether an email message is spam or good.

- Data from 4601 email messages.
- Input features: relative frequencies in a message of 57 of the most commonly occurring words and punctuation marks in all the training the email messages.
- For this problem not all errors are equal; we want to avoid filtering out good email, while letting spam get through is not desirable but less serious in its consequences.
- We coded spam as 1 and email as 0.

Decision Trees: Application

53

Spam detection – DT Training

- 48 quantitative predictors—the percentage of words in the email that match a given word. Examples include *business*, *address*, *internet*, *free*, and *george*.
- 6 quantitative predictors—the percentage of characters in the email that match a given character. The characters are ch;, ch(, ch[, ch!, ch$, and ch#.
- The average length of uninterrupted sequences of capital letters: CAPAVE.
- The length of the longest uninterrupted sequence of capital letters: CAPMAX.
- The sum of the length of uninterrupted sequences of capital letters: CAPTOT.

Decision Trees: Application

54

Spam detection – DT Training

- A test set of size 1536 was randomly chosen, leaving 3065 observations in the training set.
- A full tree was grown on the training set, with splitting continuing until a minimum bucket size of 5 was reached.
- This bushy tree was pruned back using cost-complexity pruning, and the tree size was chosen by 10-fold cross-validation.
- We then compute the test error and ROC curve on the test data.

Decision Trees: Application

Spam detection – Training

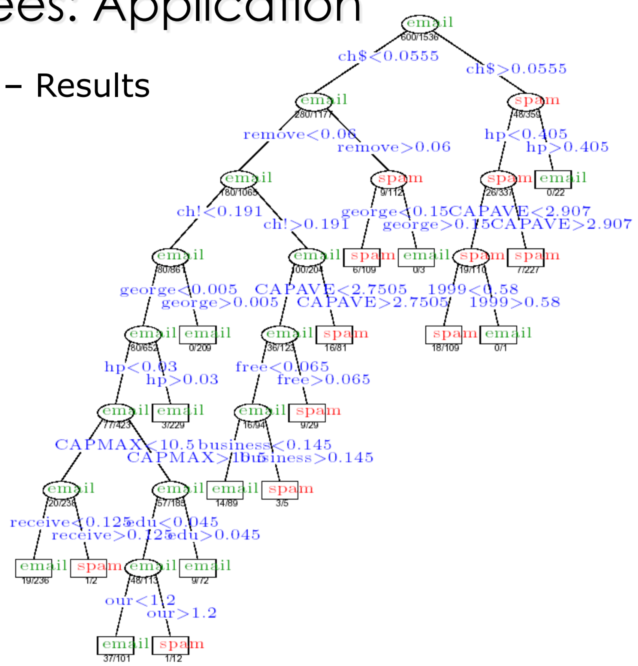
- 39% of the training data were spam. Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam and email.

	george	you	your	hp	free	hpl
spam	0.00	2.26	1.38	0.02	0.52	0.01
email	1.27	1.27	0.44	0.90	0.07	0.43

	!	our	re	edu	remove
spam	0.51	0.51	0.13	0.01	0.28
email	0.11	0.18	0.42	0.29	0.01

Decision Trees: Application

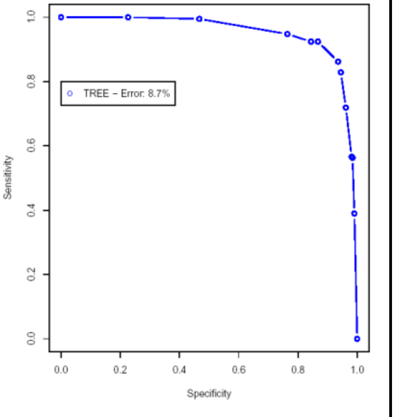
Spam detection – Results



Decision Trees: Application

Spam detection – Results

- ROC curve for pruned tree on SPAM data
- Overall error rate on test data: 8.7%.
- Sensitivity
(detection rate: DR)
proportion of true spam identified
- Specificity 1- FAR (false alarm rate))
proportion of true email identified.

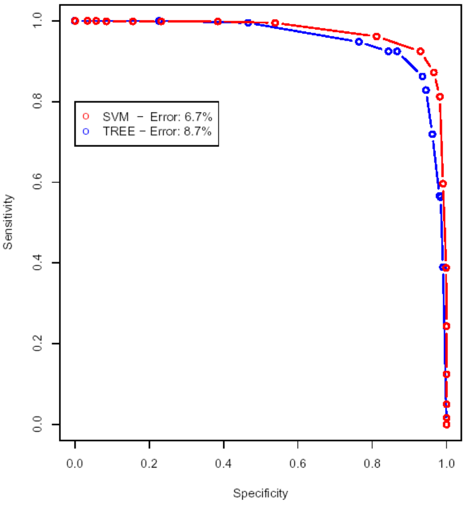


We may want specificity to be high, and suffer some spam

➔ Specificity : 95% ⇒ Sensitivity : 79%

Decision Trees: Application

- Spam detection – DT vs. SVM



- Comparing ROC curves on the test data is a good way to compare classifiers.

➔ SVM dominates DT here.

➔ But DT much faster!

Decision Trees: Literature

- L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.
- C4.5 : Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan
- Learning Classification Trees, Wray Buntine, Statistics and Computation (1992), Vol 2, pages 63-73
- Kearns and Mansour, On the Boosting Ability of Top-Down Decision Tree Learning Algorithms, STOC: ACM Symposium on Theory of Computing, 1996“