Lecture 3: Decision Trees

Cognitive Systems - Machine Learning

Part I: Basic Approaches of Concept Learning

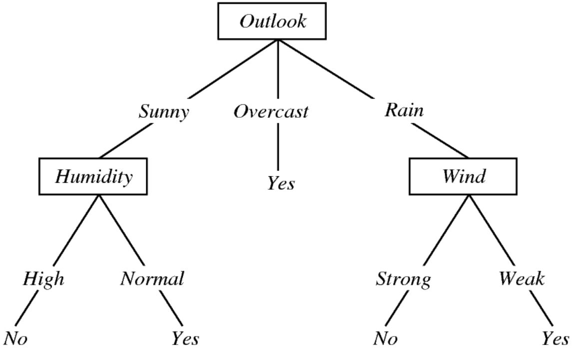
ID3, Information Gain, Overfitting, Pruning

last change November 17, 2011

Decision Tree Representation

- classification of instances by sorting them down the tree from the root to some leaf node
  - **node** $\approx$ test of some attribute
  - **branch** $\approx$ one of the possible values for the attribute
- decision trees represent a **disjunction of conjunctions of constraints on the attribute values of instances**
  - i.e., $(\ldots \wedge \ldots \wedge \ldots) \vee (\ldots \wedge \ldots \wedge \ldots) \vee \ldots$
- equivalent to a set of if-then-rules
  - each branch represents one if-then-rule
    - **if-part**: conjunctions of attribute tests on the nodes
    - **then-part**: classification of the branch

Decision Tree Representation



- This decision tree is equivalent to:

  if ($Outlook = Sunny$) $\wedge$ ($Humidity = Normal$) then *Yes*;
  if ($Outlook = Overcast$) then *Yes*;
  if ($Outlook = Rain$) $\wedge$ ($Wind = Weak$) then *Yes*;

Appropriate Problems

- Instances are represented by attribute-value pairs, e.g. (Temperature, Hot)
- Target function has discrete output values, e.g. *yes* or *no* (concept/classification learning)
- **Disjunctive descriptions** may be required
- Training data may **contain errors**
- Training data may contain **missing attribute values**

$\Rightarrow$ last three points make Decision Tree Learning more attractive than CANDIDATE-ELIMINATION

ID3

- learns decision trees by constructing them **top-down**
- employs a **greedy search algorithm without backtracking** through the space of all possible decision trees
  $\Rightarrow$ finds a short tree (wrt path length) but not neccessarily the best decision tree
- **key idea:**
  - selection of the next attribute according to a statistical measure
  - all examples are considered at the same time (simultaneous covering)
  - recursive application with reduction of selectable attributes until each training example can be classified unambiguously

ID3

Algorithm for Concept Learning

ID3(*Examples*, *Target_attribute*, *Attributes*)

- Create a *Root* for the tree
- If all examples are **positive**, Return single-node tree *Root*, with *label* $= +$
- If all examples are **negative**, Return single-node tree *Root*, with *label* $= -$
- If *Attributes* is empty, Return single-node tree *Root*, with *label* $=$ most common value of *Target_attribute* in *Examples*
- **otherwise**, Begin
  - $A \leftarrow$ attribute in *Attributes* that best classifies *Examples*
  - decision attribute for *Root* $\leftarrow A$
  - **For each possible value** v_i **of** A
    - Add new branch below *Root* with $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* with v_i for A
    - If $Examples_{v_i}$ is empty
      Then add a leaf node with label = most common value of
      Target_attribute in *Examples*
      Else add ID3($Examples_{v_i}$, *Target_Attribute*, $Attributes - \{A\}$)
- Return *Root*

The best classifier

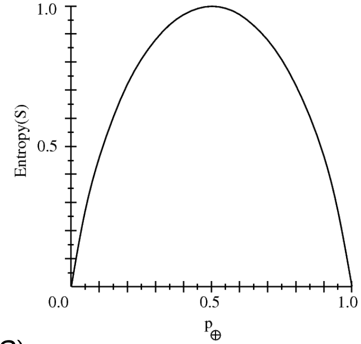
- **central choice**: Which attribute classifies the examples best?
- ID3 uses the **information gain**
  - statistical measure that indicates how well a given attribute separates the training examples according to their target classification
  - $$Gain(S, A) \equiv \underbrace{Entropy(S)}_{\text{original entropy of S}} - \underbrace{\sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)}_{\text{relative entropy of S}}$$
  - interpretation:
    - denotes the reduction in entropy caused by partitioning S according to A
    - alternative: number of saved yes/no questions (i.e., bits)

$\Rightarrow$ attribute with $\max_A Gain(S, A)$ is selected!

Entropy

- statistical measure from information theory that **characterizes (im-)purity** of an arbitrary collection of examples S
- for binary classification: $H(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$
- for n-ary classification: $H(S) \equiv \sum_{i=1}^{n} -p_i \log_2 p_i$
- **interpretation:**
  - specification of the minimum number of bits of information needed to encode the classification of an arbitrary member of S
  - alternative: number of yes/no questions

Entropy



- **minimum** of $H(S)$
  - for minimal impurity $\rightarrow$ point distribution
  - $H(S) = 0$
- **maximum** of $H(S)$
  - for maximal impurity $\rightarrow$ uniform distribution
  - for binary classification: $H(S) = 1$
  - for n-ary classification: $H(S) = \log_2 n$

Illustrative Example

- example days

Day	*Outlook*	*Temp.*	*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

Illustrative Example

- entropy of S

$$S = \{D1, ..., D14\} = [9+, 5-]$$
$$H(S) = -\frac{9}{14} \cdot \log_2 \frac{9}{14} - \frac{5}{14} \cdot \log_2 \frac{5}{14} = 0.940$$

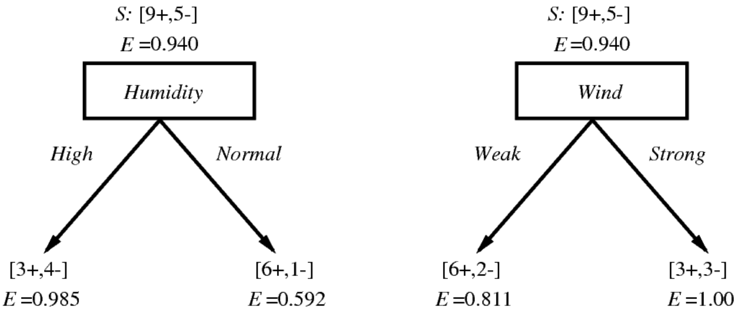
- information gain (e.g. *Wind*)

$$S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\} = [6+, 2-]$$
$$S_{Strong} = \{D2, D6, D7, D11, D12, D14\} = [3+, 3-]$$

$$\begin{aligned} Gain(S, Wind) &= H(S) - \sum_{v \in Wind} \frac{|S_v|}{|S|} \cdot H(S_v) \\ &= H(S) - \frac{8}{14} \cdot H(S_{Weak}) - \frac{6}{14} \cdot H(S_{Strong}) \\ &= 0.940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} 1.000 \\ &= 0.048 \end{aligned}$$

Illustrative Example

Which attribute is the best classifier?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Illustrative Example

- informations gains for the four attributes:

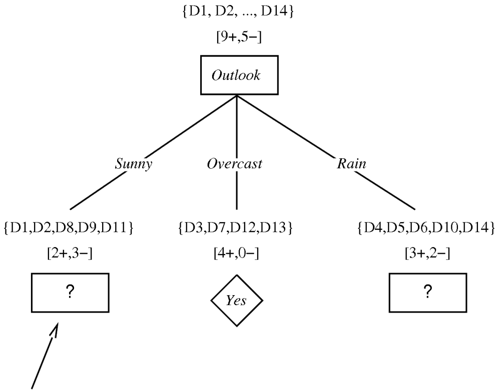
  $Gain(S, Outlook) = 0.246$
  $Gain(S, Humidity) = 0.151$
  $Gain(S, Wind) = 0.048$
  $Gain(S, Temperature) = 0.029$

⇒ *Outlook* is selected as best classifier and is therefore *Root* of the tree

⇒ now branches are created below the root for each possible value

  - because every example for which $Outlook = Overcast$ is positive, this node becomes a leaf node with the classification *Yes*
  - the other descendants are still ambiguous ($H(S) \neq 0$)
  - hence, the decision tree has to be further elaborated below these nodes

Illustrative Example



Which attribute should be tested here?

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

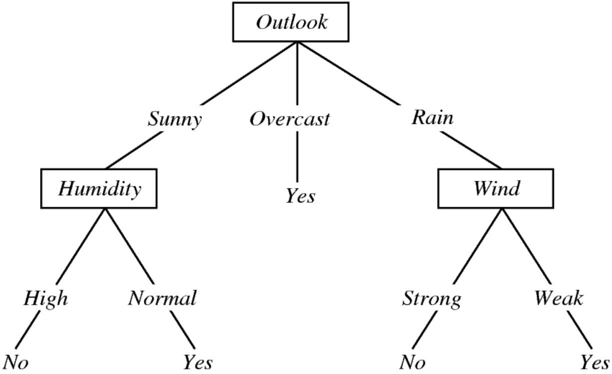
$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

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

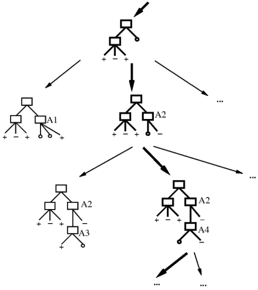
$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Illustrative Example

- Resulting decision tree



Hypothesis Space Search



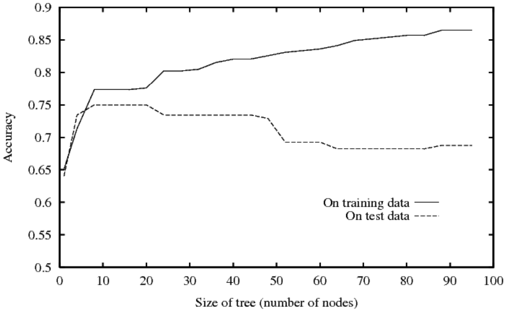
- $H \approx$ *complete* space of finite discrete functions, relative to the available attributes (i.e. all possible decision trees)
- **capabilites and limitations:**
  - returns just one single consistent hypothesis
  - performs greedy search (i.e., $\max_A Gain(S, A)$)
  - susceptible to the usual risks of hill-climbing without backtracking
  - uses all training examples at each step $\Rightarrow$ simultaneous covering

Inductive Bias

- as mentioned above, ID3 searches
  - *complete* space of possible hypotheses (wrt instance space), but not *completely* ⇒ Preference Bias
- **Inductive bias**: Shorter trees are prefered to longer trees. Trees that place high information gain attributes close to the root are also preferred.
- Why prefer shorter hypotheses?
  - **Occam's Razor**: Prefer the simplest hypothesis that fits the data! (aka W. Ockham)
  - see Minimum Description Length Principle (Bayesian Learning)
  - e.g., if there are two decision trees, one with 500 nodes and another with 5 nodes, the second one should be prefered

  ⇒ better chance to avoid overfitting

Overfitting



- Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training data, but h' has smaller error than h over the entire distribution of instances.

Overfitting – Example

Day	*Outlook*	*Temp.*	*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**

Wrong classification in last example (noise)

⇒ Resulting tree is more complex and has different structure

⇒ Tree still fits training set but wrong classification of unseen examples

Overfitting

- **reasons for overfitting:**
  - noise in the data
  - number of training examples is too small to produce a representative sample of the target function
- **how to avoid overfitting:**
  - **stop the tree growing earlier**, before it reaches the point where it perfectly classifies the training data, i.e. create a leaf and assign the most common concept
  - allow overfitting and then **post-prune** the tree (more successful in practice!)
- **how to determine the perfect tree size:**
  - separate validation set to evaluate utility of post-pruning
  - apply statistical test to estimate whether expanding (or pruning) produces an improvement

Training Set, Validation Set and Test Set

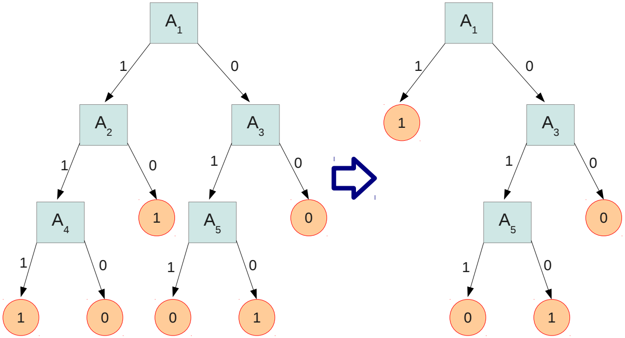
- **Training Set**
  - used to form the learned hypothesis
- **Validation Set**
  - separated from training set
  - used to evaluate the accuracy of learned hypothesis over subsequent data
  - (in particular) used to evaluate the impact of pruning this hypothesis
- **Test Set**
  - separated from training set
  - used **only** to evaluate the accuracy of learned hypothesis over subsequent data
  - no more learning / adjustment of the parameters after applying the test set

Post-Pruning

Prune the tree after it has been generated to avoid overfitting.

Two approaches:

1. Reduced Error Pruning
2. Rule Post-Pruning

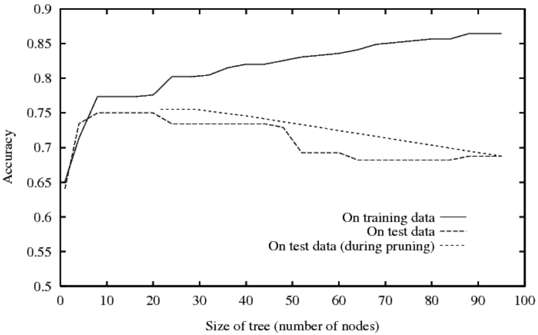


Reduced Error Pruning

- each of the decision nodes is considered to be a candidate for pruning
- **pruning** a decision node consists of removing the subtree rooted at the node, making it a leaf node and assigning the most common classification of the training examples affiliated with that node
- nodes are removed only if the resulting tree performs **not worse** than the original tree over the validation set
- pruning starts with the node whose removal most increases accuracy and continues until further pruning is harmful

Reduced Error Pruning

- **effect of reduced error pruning:**



- any node added to coincidental regularities in the training set is likely to be pruned
- the stronger the pruning (less number of nodes), the better is the fitting to the test set
- the validation set used for pruning is distinct from both the training and test sets

Rule Post-Pruning

- rule post-pruning involves the following steps:
  1. Infer the decision tree from the training set (Overfitting allowed!)
  2. Convert the tree into a set of rules
  3. Prune each rule by removing any preconditions that result in improving its estimated accuracy
  4. Sort the pruned rules by their estimated accuracy
- one method to estimate rule accuracy is to use a separate validation set
- pruning rules is more precise than pruning the tree itself

Alternatives to Information Gain

- natural bias in information gain favors attributes with many values over those with few values
- e.g. attribute *Date*
  - very large number of values (e.g. March 21, 2005)
  - inserted in the above example, it would have the highest information gain, because it perfectly separates the training data
  - but the classification of unseen examples would be impossible
- alternative measure: *GainRatio*
  - $GainRatio(S, A) = \frac{Gain(S,A)}{SplitInformation(S,A)}$

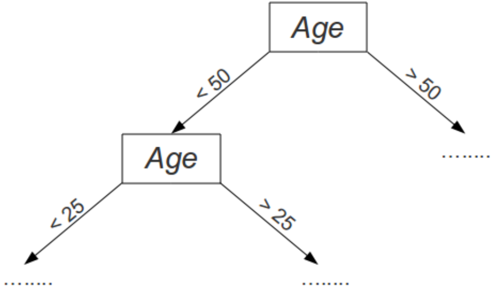
    $$SplitInformation(S, A) \equiv -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

  - $SplitInformation(S, A)$ is sensitive to how broadly and uniformly A splits S (entropy of S with respect to the values of A)

  $\Rightarrow$ *GainRatio* penalizes attributes such as *Date*

Real-Valued Attributes

- Decision tree learning with real valued attributes is possible
- Discretization of data by split in two ranges (ID3) estimating number of best disciminating ranges (CAL5)
- One attribute must be allowed to occur more than one time on a path in the decision tree!



General ML-Methods
Determining the Generalization Error

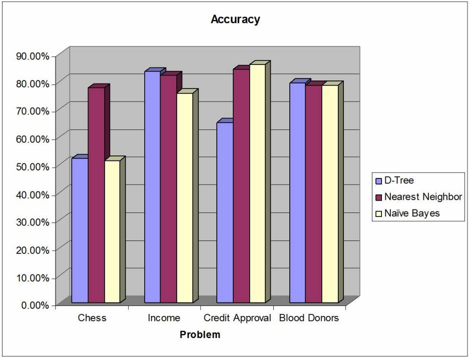
- Simple method: Randomly select part of the data from the training set for a test set
- Unbiased estimate of error because hypothesis is chosen independently of test cases
- But: the estimated error may still vary from the true error!
- Estimate the confidence interval in which the true error lies with a certain probability (see Mitchell, chap. 5)

General ML-Methods
Cross Validation

- For many learning algorithms, it is useful to provide an additional validation set (e.g. for parameter fitting)
- k-fold cross validation:
  - partition training set with m examples into k disjoint subsets of size $\frac{m}{k}$
  - run k times with a different subset as validation set each times (using the combined other subsets as training set)
  - Calculate the mean of your estimates over the k runs
  - Last run with complete training set and parameters fixed to the estimates

Compare Learners

- Which learner obtains better results in some domain?
- Compare whether generalization error of one is significantly lower than of the other
- Use similar procedure to *k*-fold cross-validation to obtain data for inference statistical comparison (see Mitchell, chap. 5)



Source: Michael Wurtz

`https://sites.google.com/site/eecs349michaelwurtz/`

General ML-Methods
Bagging and Boosting

- Bagging (bootstrap aggregation, Breimann, 1996):
  - Calculate M classifiers (e.g. decision trees) over different bootstrap samples
  - Prediction by majority vote
- Boosting (Freund & Schapire, 1996)
  - Additionally introduce weights for each classifyer which are iterativly adjusted (due to classification failure/success)
- see diploma thesis of Jörg Mennicke: Classifier Learning for Imbalanced Data with Varying Misclassification Costs - A Comparison of kNN, SVM, and Decision Tree Learning (2006)

The Problem of Imbalanced Data

- In realistic settings occurance of different classes might be imbalanced (e.g. cancer screening, quality control)
- Undersampling (remove examples for the over-represented class), oversampling (clone data for the under-represented class)
- Estimate is worse for the class which occurs more seldomly, this might be the class with higher misclassification costs (e.g. decide no cancer if true class is cancer)
- Instead of over-/, undersampling, introduce different costs for misclassifications and calculate weighted error measure!
- see e.g.: Tom Hecker and Jörg Mennicke, Diagnosing Cancerous Abnormalities with Decision Tree Learning, Student Project in cooperation with Fraunhofer IIS, 2005.

Summary

- practical and intuitively understandable method for concept learning
- able to learn disjunctive, discrete-valued concepts
- noise in the data is allowed
- **ID3** is a simultaneous covering algorithm based on information gain that performs a greedy top-down search through the space of possible decision trees
- **Inductive Bias**: Short trees are preferred (Ockham's Razor)
- overfitting is an important issue and can be reduced by pruning

Learning Terminology

ID3

Supervised Learning	unsupervised learning

Approaches:

Concept / Classification	Policy Learning
symbolic	statistical / neuronal network
inductive	analytical

Learning Strategy:
$\Rightarrow$ **learning from examples**