Computational Learning Theory
Learning with Decision Trees

Dimitris Diochnos
School of Computer Science
University of Oklahoma

Outline

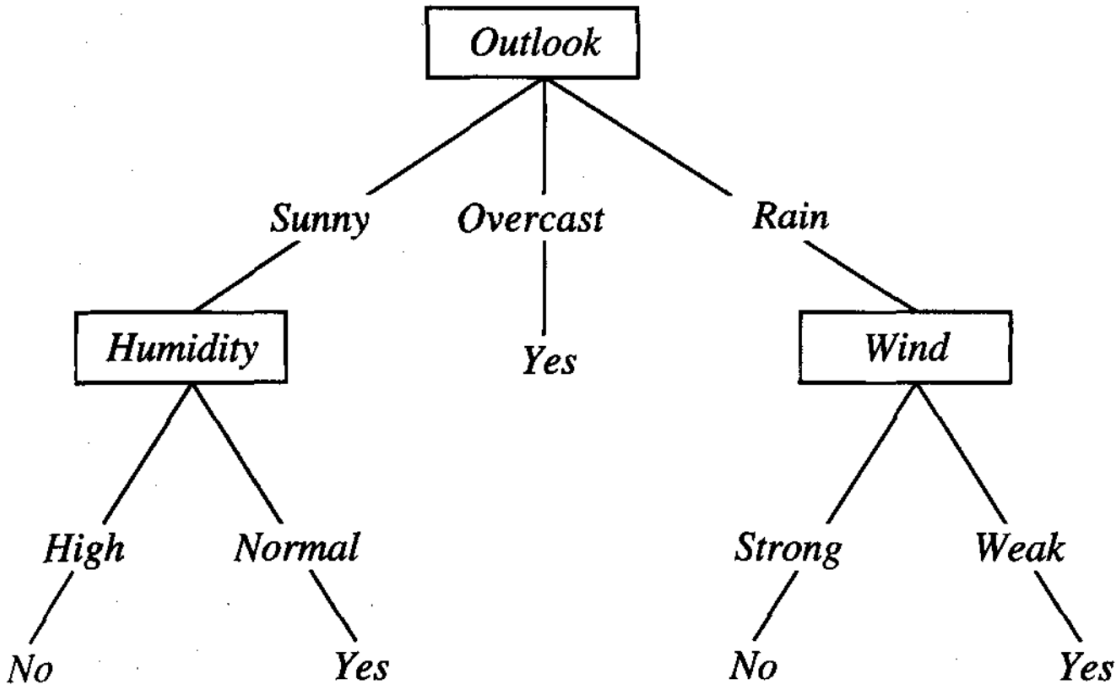
1. Learning with Decision Trees

Table of Contents

1 Learning with Decision Trees

What Decision Trees are?

- The material is based primarily on Tom Mitchell's book [4, Ch. 3]



- We can also view the decision tree in disjunctive normal form (DNF).

$$\begin{aligned} & (\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Low}) \\ \vee \quad & (\text{Outlook} = \text{Overcast}) \\ \vee \quad & (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak}) \end{aligned}$$

How do we Construct a Decision Tree?

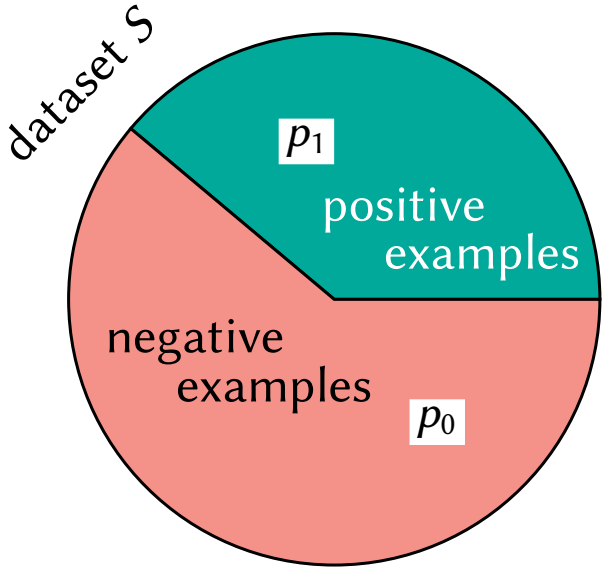
Q: Which attribute should we put in the root?

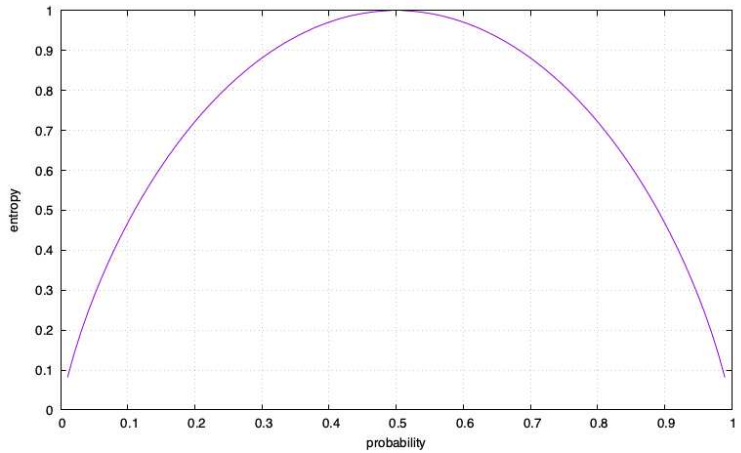
- Our answer is based on the *entropy* function.

Entropy. The entropy is calculated as shown below.

- If we have two classes ($\{0, 1\}$), then:

$$\begin{aligned} \text{Entropy}(S) &\equiv -p_0 \log_2 p_0 - p_1 \log_2 p_1 \\ &= -p_0 \log_2 p_0 - (1 - p_0) \log_2 (1 - p_0) \\ &= -(1 - p_1) \log_2 (1 - p_1) - p_1 \log_2 p_1 \end{aligned}$$





How do we Construct a Decision Tree? (cont'd)

- In general, if we have k classes, the entropy of S

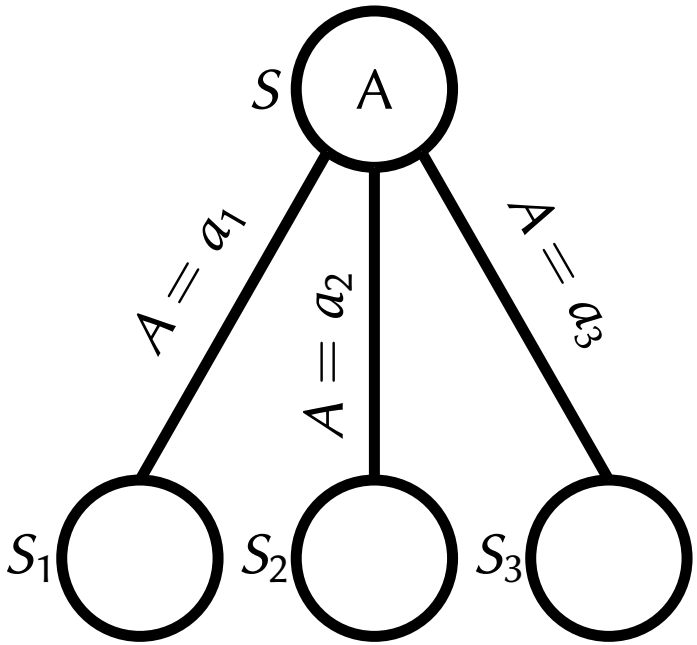
$$\text{Entropy}(S) \equiv -\sum_{i=1}^{k} p_i \log_2 p_i \,.$$

- If $p_i = 0$, then set $p_i \log_2 p_i = 0$ (consistent with $\lim_{p \to 0^+} p \log_2 p$)
- Ultimately we care about the *Information Gain*.

$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$

Figure: figure here

How do we Construct a Decision Tree? (cont'd)



$$S = S_1 \cup S_2 \cup S_3$$

1. Associate training examples to each node that we construct
   - At the root we assign all the training examples S
2. Iterate over the attributes that we have not used and identify which attribute yields the largest information gain.
   - Use that attribute A as a test for that node.
3. The recursive process stops when a node becomes "pure".
   - pure node: all the examples associated with the node have the same label.

ID3 Algorithm

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

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

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

Comments

Problems Appropriate for Decision Trees

- Disjunctive descriptions may be required
- Training data may contain errors
- Training data may be missing attribute values

Bias in Decision Trees

- Shorter trees are preferred
- Select trees that place attributes with highest information gain closest to the root.
- No backtracking.

Origins of Algorithms

- ID3 (Iterative Dichotomizer 3) – due to Quinlan [5].
- CART (Classification and Regression Trees) – due to Breiman [1].

Example: Play Tennis

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

- What is the information gain if we use the attribute Wind to split this set of training examples?
- Values(Wind) = {Weak, Strong}
- $S_{\text{weak}} = [6\oplus, 2\ominus]$
- $S_{\text{strong}} = [3\oplus, 3\ominus]$

Example: Play Tennis

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

Based on this decomposition we can calculate the following.

$$\begin{aligned}
\text{Entropy}\,(S) &= -\tfrac{9}{14}\lg\tfrac{9}{14} - \tfrac{5}{14}\lg\tfrac{5}{14} \approx 0.4098 + 0.5305 = 0.9403.\\
\text{Entropy}\,(S_{\text{Weak}}) &= -\tfrac{6}{8}\lg\tfrac{6}{8} - \tfrac{2}{8}\lg\tfrac{2}{8} \approx 0.3113 + 0.5 = 0.8113.\\
\text{Entropy}\left(S_{\text{Strong}}\right) &= -\tfrac{3}{6}\lg\tfrac{3}{6} - \tfrac{3}{6}\lg\tfrac{3}{6} = 0.5 + 0.5 = 1.
\end{aligned}$$

Example: Play Tennis

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

$$\text{Gain}(S, \text{Wind}) = \text{Entropy}(S) - \sum_{v \in \{\text{Weak, Strong}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$= 0.9403 - \frac{8}{14} \cdot 0.8113 - \frac{6}{14} \cdot 1 \approx 0.04813$$

Example: Play Tennis

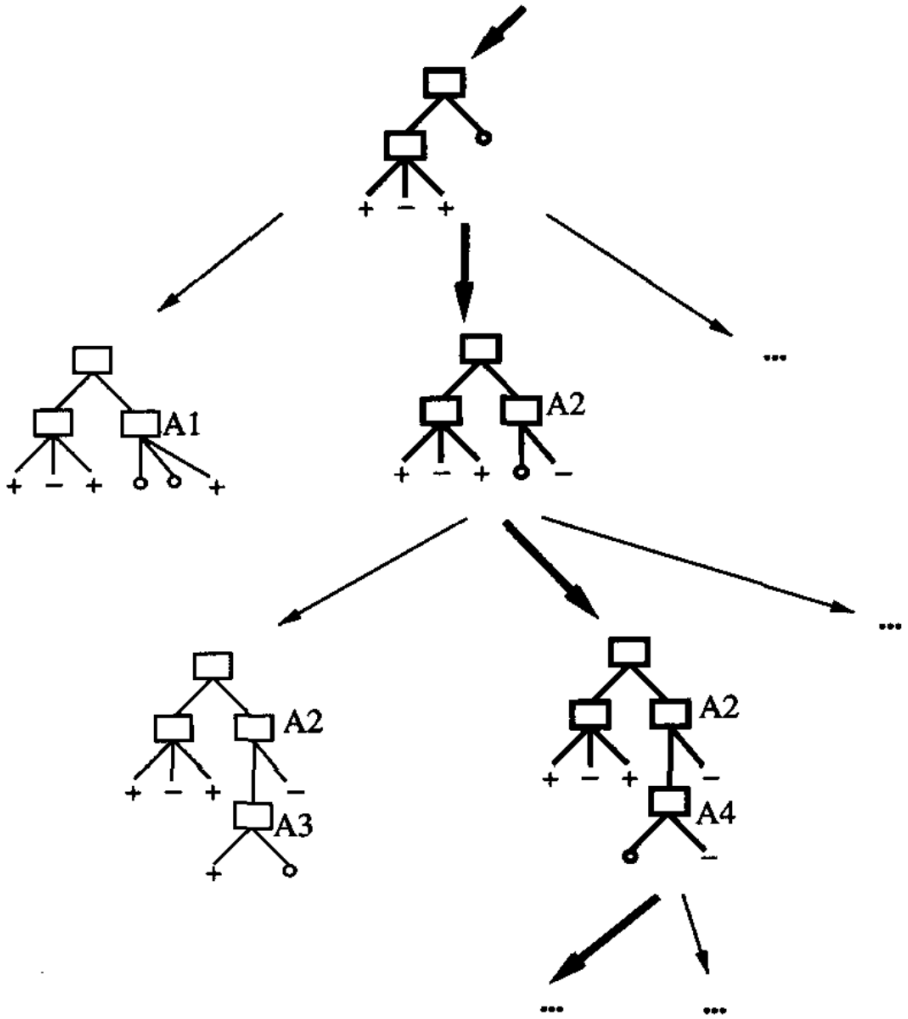
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool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

- Overall we have:

$$\begin{cases} \text{Gain}\,(S, \text{Wind}) & \approx & 0.048, \\ \text{Gain}\,(S, \text{Temperature}) & \approx & 0.029, \\ \text{Gain}\,(S, \text{Humidity}) & \approx & 0.151, \text{ and} \\ \text{Gain}\,(S, \text{Outlook}) & \approx & 0.247. \end{cases}$$

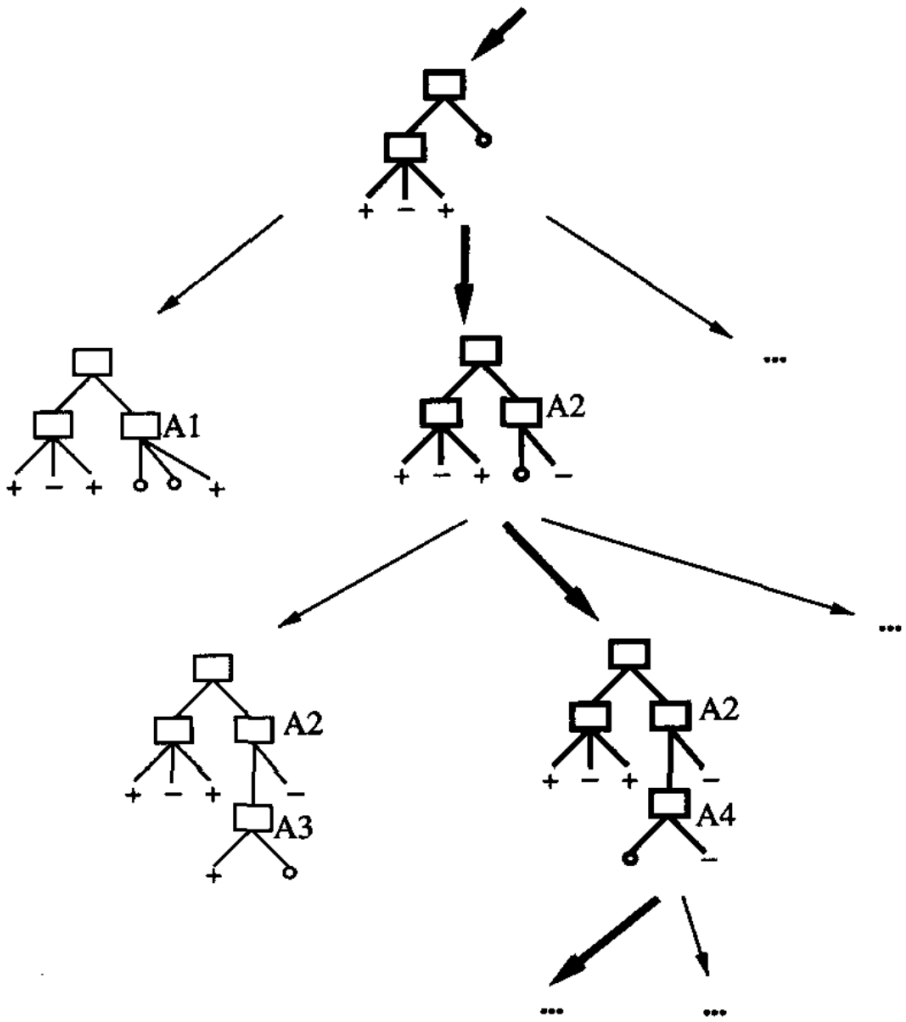
$\Longrightarrow$ Our choice is 'Outlook' as this yields the largest gain.

Hypothesis Space Search in Decision Tree Learning



- $\mathcal{H}$: the set of all possible decision trees.
- ID3: simple-to-complex, hill climbing search starting from the empty tree.
  - Guide: information gain.
- "Complete" search space.
  - The target is always there.
- Maintain only one hypothesis.
  Contrast with CANDIDATE-ELIMINATION algorithm. Disadvantages:
  - Don't know how many trees are consistent.
  - Cannot pose queries in order to resolve among competing hypotheses.

Hypothesis Space Search in Decision Tree Learning



- No backtracking (ID3).
  Disadvantage:
  - Local optima.
- Statistically-based decisions, using all the examples.
  Contrast with CANDIDATE-ELIMINATION algorithm. Advantages:
  - Much less sensitive to errors in individual training examples.
  - Can be modified to accept hypotheses that imperfectly fit the training data.

Bias in Decision Tree Learning

- Shorter trees are preferred over longer ones.
- Select trees that place attributes with highest-information gain closest to the root.

Note. Finding the shortest tree that fits the data is a computationally hard problem. In fact, it is computationally hard even to approximate the solution [2].

Search Bias vs Language Bias

Preference/Search Bias vs **Restriction/Language Bias**

ID3

- Complete hypothesis space
- Search incompletely
- Bias is consequence of the ordering of hypotheses during search. Not because of $\mathcal{H}$.

Candidate-Elimination

- Incomplete hypothesis space
- Search completely
- Bias due to the expressive power of $\mathcal{H}$. Not because of the search strategy.

- In general: we prefer search bias over language bias.
- Learning systems may combine both biases.
  - e.g., use a linear separator as evaluation function on a game; where the coefficients are tuned using a least-mean-squares approach.

Occam's Razor

"Nunquam ponenda est pluralitas sin necesitate"
William of Occam (1320 A.D.)

"Entities should not be multiplied beyond necessity "
Stephen Chak Tornay (1938 A.D.)

"Prefer the simplest hypothesis that fits the data" – Tom Mitchell (1997 A.D.)

- Fewer short hypotheses (*short descriptions*) compared to longer ones.
  $\Longrightarrow$ Less likely to find by a coincidence a hypothesis that fits the data.

Counterarguments...

1. Be very specific on something complex $\Longrightarrow$ Simple solutions are unlikely to fit the data. So, why not prefer complex ones?
2. The learner may arrive at a different hypothesis because it is using a different representation internally. Still, both learners justify their solutions through Occam's razor principle and may even predict differently on unseen data.

(True) Risk and Empirical Risk

Definition 1 (Risk)

Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$, the **risk** of h is defined by

$$R_{\mathcal{D}}(h, c) = \mathbf{Pr}_{x \sim \mathcal{D}}(h(x) \neq c(x)) = \mathop{\mathbf{E}}_{x \sim \mathcal{D}}[\mathbf{1}\{h(x) \neq c(x)\}] .$$

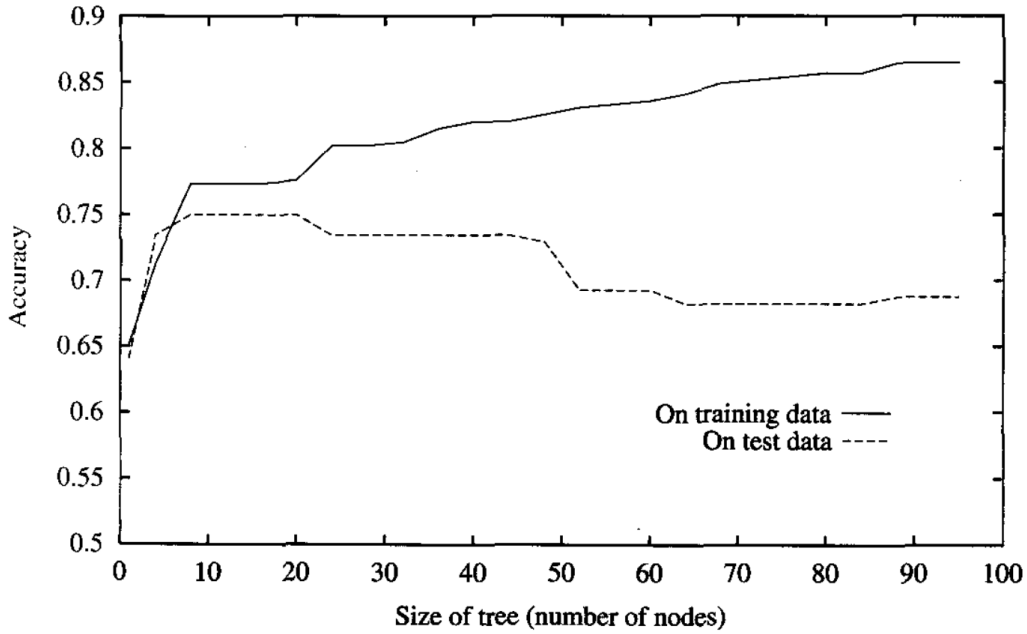
- $\mathbf{1}\{\mathcal{A}\}$ returns 1 if the event $\mathcal{A}$ holds, o.w. returns 0.

Definition 2 (Empirical Risk)

Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \ldots, x_m)$, the **empirical risk** of h is defined by

$$\widehat{R}_S(h, c) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbf{1}\{h(x_i) \neq c(x_i)\} .$$

Overfitting and Pruning



Definition 3

Given a hypothesis space $\mathcal{H}$, a hypothesis $h \in \mathcal{H}$ is said to **overfit** the training data S if there exists some alternative hypothesis $h' \in \mathcal{H}$, such that

$$\widehat{R}_S(h, c) \leq \widehat{R}_S(h', c) \qquad \text{but} \qquad R_D(h, c) > R_D(h', c) \ .$$

Overfitting (cont'd)

Reasons for overfitting.

- Noise. For example, the teacher makes a mistake and the learner receives an incorrect label.
  - As a consequence we create a more elaborate tree so that we can fit the training data better.
  - However, we will not generalize well on unseen data.
- Coincidental regularities, especially when we have few training data.

Two ideas to overcome overfitting.

1. Stop growing the tree earlier.
   - Not easy to say when to stop.
2. Allow overfitting and prune the tree later.

Which criterion should determine the 'correct' final tree?

Tuning our Model in Order to Reduce Overfitting

Three approaches.

1. [Most common approach] Use a training set and a validation set.
   - Validation set acts as a safety net against overfitting the spurious characteristics of the training set.
   - Typically the training set is 3-4 times larger than the validation set.
2. Use a statistical test to predict whether expanding a node of a decision tree will improve the performance over the entire distribution, or only on the training set; e.g., [5].
3. Apply the *Minimum Description Length* principle; i.e., try to minimize a complexity-theoretic notion on encoding a hypothesis that we generate; e.g., [3, 8].

Reduced-Error Pruning

Pruning. Pruning a node consists of:

1. Removing subtree rooted at that node.
2. Making the node a leaf.
3. Assign most common classification of examples affiliated with that node.

Algorithm (Quinlan; [7])

- Each node is candidate for pruning.
- A node is removed only if the pruned tree behaves no worse than the original (clearly in the validation set).
- 'Greedy': remove the node that increases the accuracy the most (again, over the validation set).
- Pruning stops when all candidate operations are harmful.

Rule Post-Pruning (C4.5)

Algorithm (Quinlan; [6, 7])

1. Infer the decision tree and allow overfitting to occur.
2. Convert the tree into a set of rules; one rule for each path root $\rightsquigarrow$ leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving the estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Say we have the following rule:

$$\begin{array}{ll} \text{IF} & (\text{Outlook} = \text{Sunny}) \wedge (\text{Humidity} = \text{High}) \\ \text{THEN} & \text{PlayTennis} = \text{No} \end{array}$$

Then, we may delete either one of (Outlook = Sunny) or (Outlook = Sunny).

Benefits of Rule Post-Pruning

1. Different contexts in which a decision node is used.
   - In some paths it is retained and it is removed in some other paths.
   - Contrast with removing the node completely.
2. Removes the distinction of pruning nodes near the root and those near the leaves.
3. Rules improve readability.

Handling Continuous-Valued Attributes

- For a continuous-valued attribute A, we consider a threshold t and then create the Boolean attribute A_t which is TRUE if $A < t$ and FALSE otherwise.
- How do we select the best threshold t?

Algorithm.

1. Sort the examples according to the attribute A
2. Identify adjacent examples that differ in their target classification.
   - Introduce thresholds at the average values of the pairs that are adjacent and the classification is different.
3. Test these points and pick the one that produces the greatest information gain.

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

Candidate thresholds: $t_1 = \frac{48+60}{2} = 54$ and $t_2 = \frac{80+90}{2} = 85$.

Alternative Measures for Selecting Attributes

Gain $(\cdot, \cdot)$ has a natural bias: favors attributes with many values over attributes with fewer values.

- Extreme example: attribute 'Day'.
  Creates a very broad tree of depth 1 and predicts correctly.
  However, this is going to be a poor predictor.
- Problem: many possible values $\Rightarrow$ small sets of examples along each path.

Gain Ratio as an Alternative to Information Gain

Alternative to Information Gain: Gain Ratio. Let A be an attribute that takes k different values. Then,

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

- $\text{SplitInformation}\left(\cdot, \cdot\right)$ is the entropy of S with respect to the values of A – not with respect to the labels!

The **GainRatio** is defined to be:

$$\text{GainRatio}\left(S, A\right) \equiv \frac{\text{Gain}\left(S, A\right)}{\text{SplitInformation}\left(S, A\right)}.$$

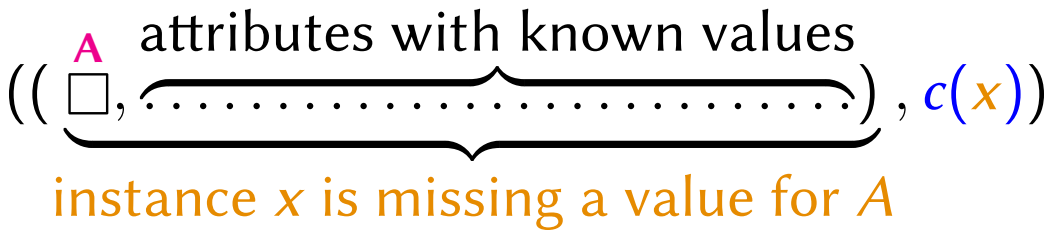
Issue. Denominator can be 0 or very small when $|S_i| \approx |S|$. Workarounds:

- First compute $\text{Gain}\left(S, A\right)$ for each attribute A. Then consider only the attributes that have above average gain and apply $\text{GainRatio}\left(\cdot, \cdot\right)$ there.
- Distance-based heuristic comparing the data partition and the perfect partition.

Other Alternatives to Information Gain also Exist.

Handling Training Examples with Missing Attribute Values

- We may be missing the result of a lab test in a medical domain; e.g., blood test.



- How can we consider a split around A if we are missing values as in this example?
  - How can we compute Gain (S, A)?

Solutions to assign a value to $A(x)$.

1. Assign the most common value among the examples associated with the particular node.
2. Assign the most common value among the examples associated with the particular node, for which the prediction is $c(x)$.
3. Fractional examples [6]; assign a probability to each possible value $A(x)$ based on the frequencies of the observed values in the examples associated with the particular node.

Some Last Remarks on Decision Trees

Attributes with Differing Costs.

- One can also deal with attributes that have different costs when we want to know their values.
  - Measuring patient's temperature is cheap (no monetary cost, no patient discomfort)
  - A blood test on the other hand costs money and it is not as comfortable.
- We would like to work with decision trees that work with low-cost attributes and only use high-cost attributes if necessary.

Some ideas:

$$\frac{(\text{Gain}\,(S, A))^2}{\text{Cost}(A)},$$

or,

$$\frac{2^{\text{Gain}(S,A)} - 1}{(Cost(A) + 1)^w},$$

where $w \in [0, 1]$.

References I

[1] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[2] Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower Bounds on Learning Decision Lists and Trees. *Information and Computation*, 126(2):114–122, 1996.

[3] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-Based Decision Tree Pruning. In *First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 216–221. AAAI Press, 1995.

[4] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[5] J. Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[6] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

References II

[7] J. Ross Quinlan. Simplifying decision trees. *International Journal of Human Computer Studies*, 51(2):497–510, 1999.

[8] J. Ross Quinlan and Ronald L. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80(3):227–248, 1989.