Decision Tree Learning

Based on "Machine Learning", T. Mitchell, McGRAW Hill, 1997, ch. 3

Acknowledgement:

The present slides are an adaptation of slides drawn by T. Mitchell

PLAN

- **DT Learning: Basic Issues**

  1. **Concept learning: an example**
  2. **Decision tree representation**
  3. **ID3 learning algorithm (Ross Quinlan, 1986)**

     Hypothesis space search by ID3

     Statistical measures in decision tree learning:
     Entropy, Information gain
  4. **Inductive bias in ID3**
  5. **Time complexity of the ID3 algorithm**
  6. **Other "impurity" measures (apart entropy): Gini, missclassification**

PLAN (cont'd)

- **Useful extensions to the ID3 algorithm**
  1. **Dealing with...**
     continuous-valued attributes
     attributes with many values
     attributes with different costs
     training examples with missing attributes values
  2. **Avoiding overfitting of data:**
     reduced-error prunning, and rule post-pruning
- **Advanced Issues**
  Ensemble Learning using DTs: boosting, bagging, Random Forests

When to Consider Decision Trees

- Instances are described by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

1. Basic Issues in DT Learning

1.1 Concept learning: An example

Given the data:

Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
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

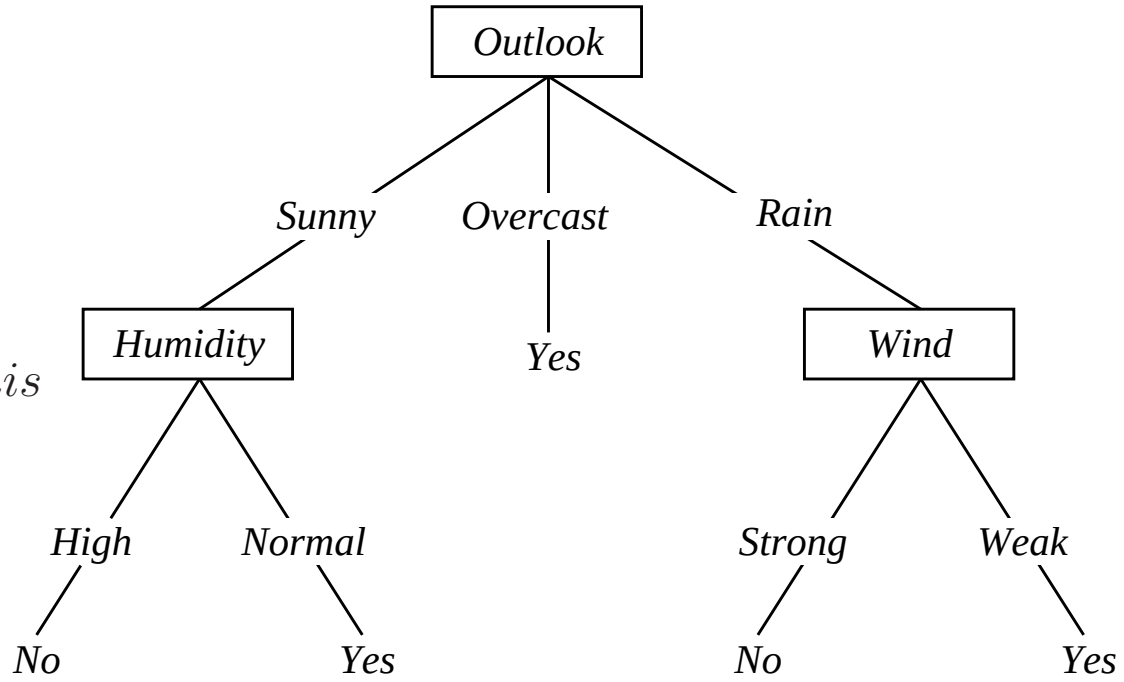
predict the value of EnjoyTennis for

$$\langle Outlook = sunny,\ Temp = cool,\ Humidity = high,\ Wind = strong \rangle$$

1.2. Decision tree representation

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

Example:
Decision Tree for $EnjoyTennis$



Another example:
A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

1.3. Top-Down Induction of Decision Trees:

ID3 algorithm outline

[Ross Quinlan, 1979, 1986]

START

create the root *node*;
assign all examples to root;

Main loop:

1. $A \leftarrow$ **the "best" decision attribute for next** *node*;
2. **for each value of** A**, create a new descendant of** *node*;
3. **sort training examples to leaf nodes;**
4. **if training examples perfectly classified, then STOP;**
   else iterate over new leaf nodes

ID3 Algorithm: basic version

ID3(*Examples, Target_attribute, Attributes*)

- create a *Root* node for the tree; assign all *Examples* to *Root*;
- 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 = the most common value of *Target_attribute* in *Examples*;
- otherwise // **Main loop:**

  $A \leftarrow$ 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 the value v_i for A;
  if $Examples_{v_i}$ is empty

  below this new branch add a leaf node with label = the most common value of *Target_attribute* in *Examples*;

  else

  below this new branch add the subtree
  ID3($Examples_{v_i}$, *Target_attribute, Attributes*$\backslash\{A\}$);

- return *Root*;

* The *best* attribute is the one with the highest *information gain.*

Hypothesis Space Search by ID3

- Hypothesis space is complete!

  The target function surely is in there...

- Outputs a single hypothesis

  Which one?

- Inductive bias:

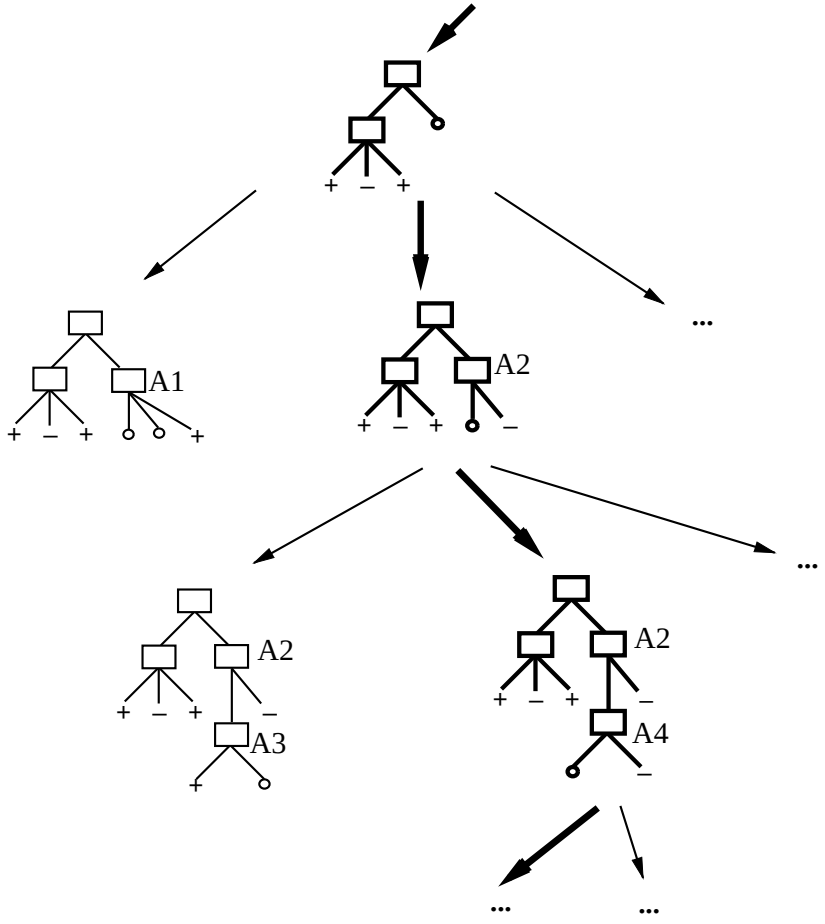
  approximate "prefer the shortest tree"

- Statisically-based search choices

  Robust to noisy data...

- No back-tracking

  Local minima...



Statistical measures in DT learning: Entropy, and Information Gain

Information gain:

the expected reduction of the entropy of the instance set S due to sorting on the attribute A

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

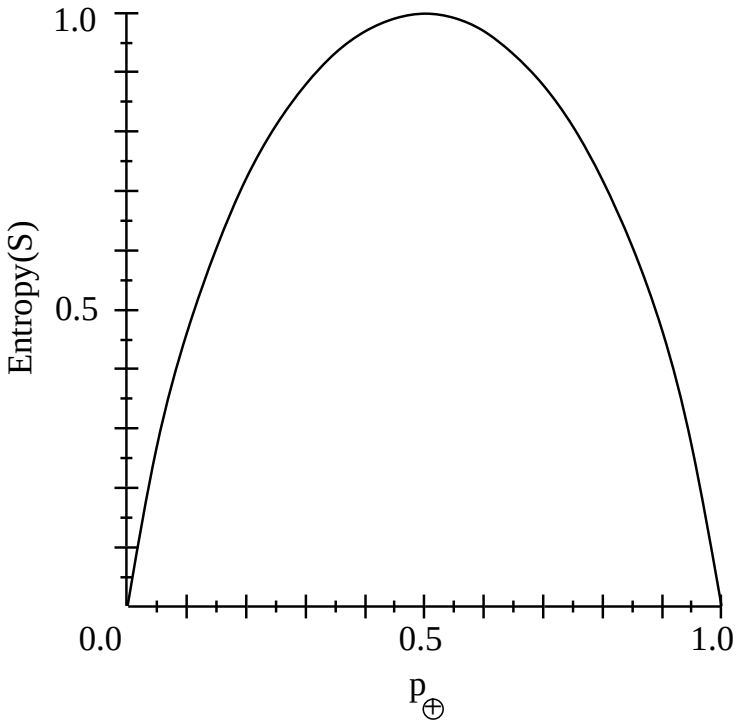
Entropy

- **Let S be a sample of training examples**
  $p_\oplus$ is the proportion of positive examples in S
  $p_\ominus$ is the proportion of negative examples in S
- **Entropy measures the impurity of S**
- **Information theory:**

  $Entropy(S)$ **= expected number of bits needed to encode $\oplus$ or $\ominus$ for a randomly drawn member of S (under the optimal, shortest-length code)**

  The optimal length code for a message having the probability p is $-\log_2 p$ bits. So:

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



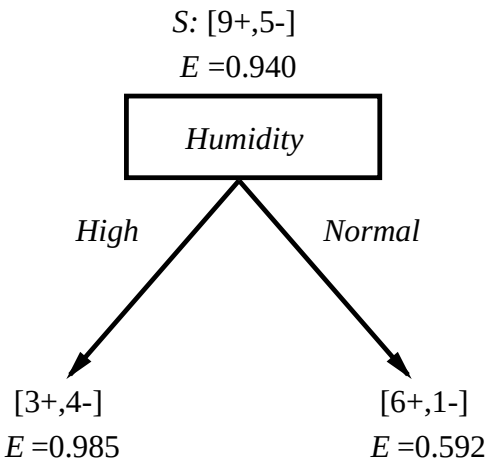
$$Entropy(S) = p_{\oplus} \log_2 \frac{1}{p_{\oplus}} + p_{\ominus} \log_2 \frac{1}{p_{\ominus}}$$

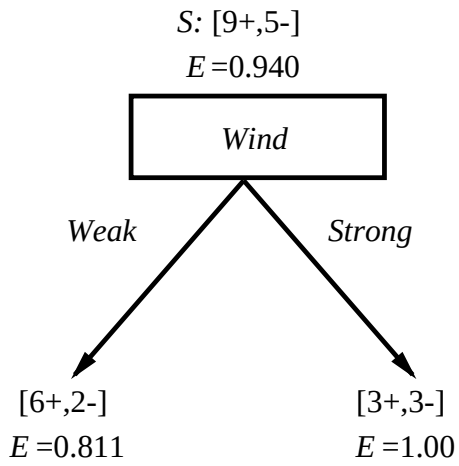
Note: By convention, $0 \cdot \log_2 0 = 0$**.**

Back to the *EnjoyTennis* example:

Selecting the root attribute

Which attribute is the best classifier?





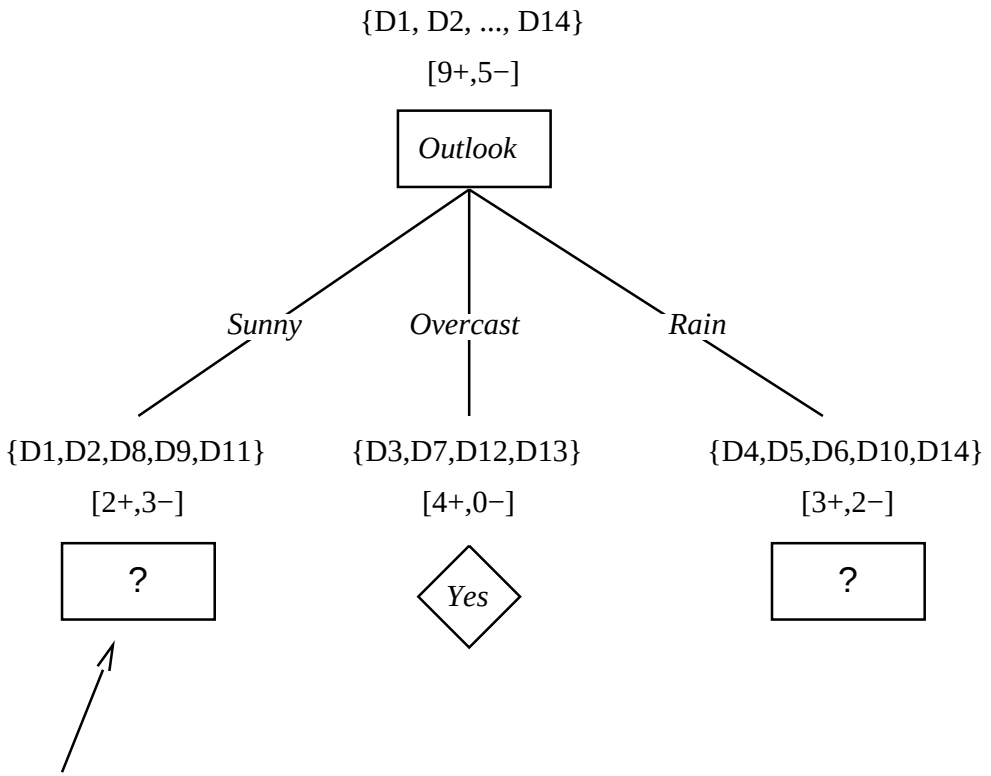
Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

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

Similarly,

$Gain(S,\ Outlook) = 0.246$
$Gain(S,\ Temperature) = 0.029$

A partially learned tree



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

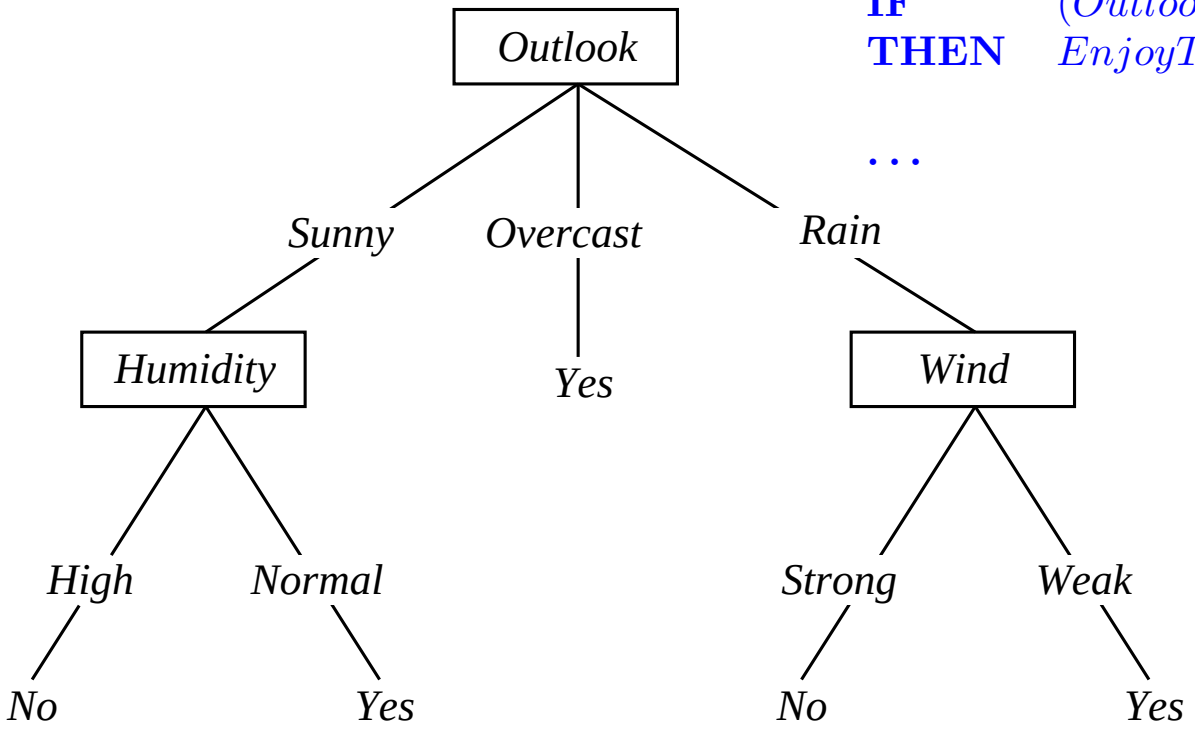
Gain (S_{sunny} , *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Converting A Tree to Rules

IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $EnjoyTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $EnjoyTennis = Yes$

...



1.4 Inductive Bias in ID3

Is ID3 unbiased?

Not really...

- **Preference for short trees, and for those with high information gain attributes near the root**
- **The ID3 bias is a *preference* for some hypotheses (i.e., a *search bias*); there are learning algorithms (e.g. CANDIDATE-ELIMINATION, ch. 2) whose bias is a *restriction* of hypothesis space H (i.e, a *language bias*).**
- **Occam's razor: prefer the shortest hypothesis that fits the data**

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hypotheses than long hypsotheses

  → a short hypothesis that fits data unlikely to be coincidence

  → a long hypothesis that fits data might be coincidence

Argument opposed:

- There are many ways to define small sets of hypotheses
  (e.g., all trees with a prime number of nodes that use attributes beginning with "Z")
- What's so special about small sets based on the *size* of hypotheses?

1.5 Complexity of decision tree induction

from "Data mining. Practical machine learning tools and techniques"
Witten et al, 3rd ed., 2011, pp. 199-200

- **Input:** d **attributes, and** m **training instances**
- **Simplifying assumptions:**

  (A1): the depth of the ID3 tree is $\mathcal{O}(\log m)$**,**

  (i.e. it remains "bushy" and doesn't degenerate into long, stringy branches);

  (A2): [most] instances differ from each other;

  (A2'): the d **attributes provide enough tests to allow the instnces to be differentiated.**

- **Time complexity:** $\mathcal{O}(d\, m \log m)$**.**

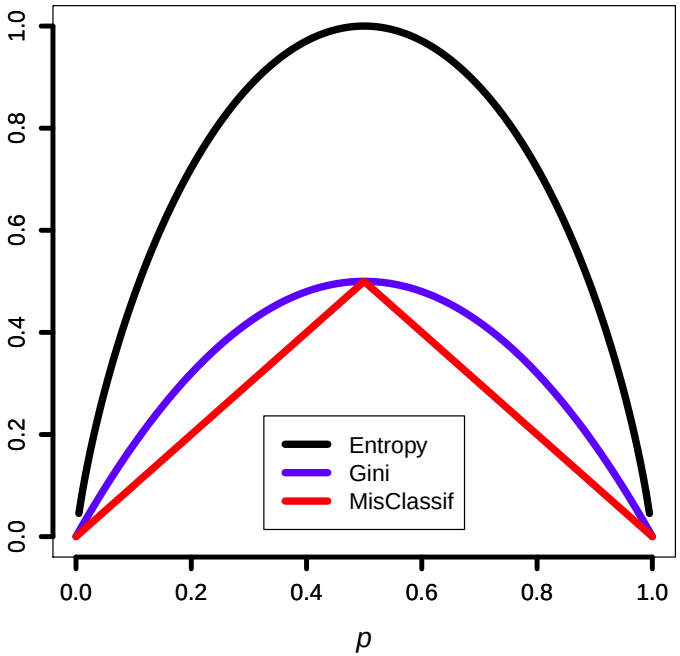
1.6 Other "impurity" measures (apart entropy)

- $i(n) \overset{not.}{=} \begin{cases} \textbf{\textit{Gini Impurity}}: \ 1 - \sum_{i=1}^{k} P^2(c_i) \\ \textbf{\textit{Misclassification Impurity}}: \ i(n) = 1 - \max_{i=1}^{k} P(c_i) \end{cases}$

- ***Drop-of-Impurity***: $\Delta i(n) \overset{not.}{=} i(n) - P(n_l) i(n_l) - P(n_r) i(n_r)$,
  where n_l and n_r are left and right child of node n after splitting.

For a Bernoulli variable of parameter p:

$$
\begin{aligned}
\textbf{\textit{Entropy}}(p) &= -p \log_2 p - (1-p) \log_2 (1-p) \\
\textbf{\textit{Gini}}(p) &= 1 - p^2 - (1-p)^2 = 2p(1-p) \\
\textbf{\textit{MisClassif}}(p) &= \begin{cases} 1 - (1-p), & \textbf{if } p \in [0; 1/2) \\ 1 - p, & \textbf{if } p \in [1/2; 1] \end{cases} \\
&= \begin{cases} p, & \textbf{if } p \in [0; 1/2) \\ 1 - p, & \textbf{if } p \in [1/2; 1] \end{cases}
\end{aligned}
$$



2. Extensions of the ID3 algorithm
2.1 Dealing with ...Continuous valued attributes

Create one or more discrete attribute to test the continuous.

For instance:

$$\textit{Temperature} = 82.5$$

$$(\textit{Temperature} > 72.3) = t, f$$

How to choose such (threshold) values:

Sort the examples according to the values of the continuous attribute, then identify examples that differ in their target classification.

For EnjoyTennis:

***Temperature*:**	**40**	**48**	**60**	**72**	**80**	**90**
***EnjoyTennis*:**	**No**	**No**	**Yes**	**Yes**	**Yes**	**No**

$$\textit{Temperature}_{>54}$$

$$\textit{Temperature}_{>85}$$

...Attributes with many values

Problem:

- **If an attribute has many values,** $Gain$ **will select it**
- **Imagine using** $Date = Jun\_3\_1996$ **as attribute**

One approach: use $GainRatio$ **instead**

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

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

where S_i **is the subset of** S **for which** A **has the value** v_i

...Attributes with different costs

Consider

- **medical diagnosis,** $BloodTest$ **has cost \$150**
- **robotics,** $Width\_from\_1ft$ **has cost 23 sec.**

Question:

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- $\frac{Gain^2(S,A)}{Cost(A)}$ **(Tan and Schlimmer, 1990)**
- $\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$ **(Nunez, 1988)**

where $w \in [0,1]$ **determines the importance of cost**

...Training examples with unknown attribute values

Question:

What if an example is missing the value of an attribute A?

Answer:

Use the training example anyway, sort through the tree, and if node n tests A,

- assign the most common value of A among the other examples sorted to node n, or
- assign the most common value of A among the other examples with same target value, or
- assign probability p_i to each possible value v_i of A; assign the fraction p_i of the example to each descendant in the tree.

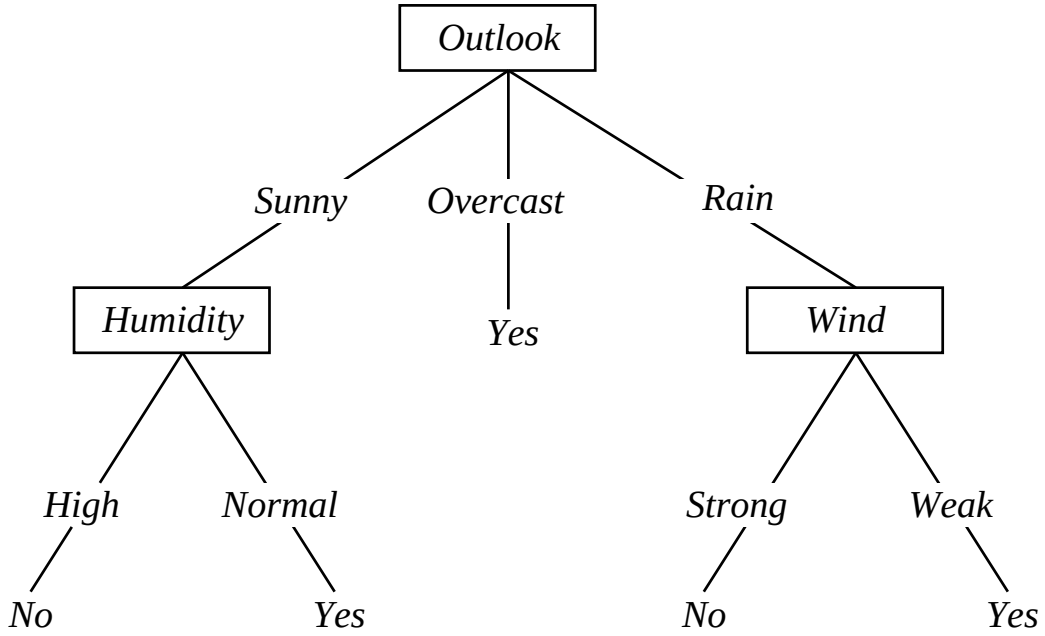
Classify the test instances in the same fashion.

2.2 Overfitting in Decision Trees

Consider adding noisy training example #15:

$$(Sunny,\ Hot,\ Normal,\ Strong,\ EnjoyTennis = No)$$

What effect does it produce on the earlier tree?



Overfitting: Definition

Consider error of hypothesis h over

- **training data:** $error_{train}(h)$
- **entire distribution $\mathcal{D}$ of data:** $error_{\mathcal{D}}(h)$

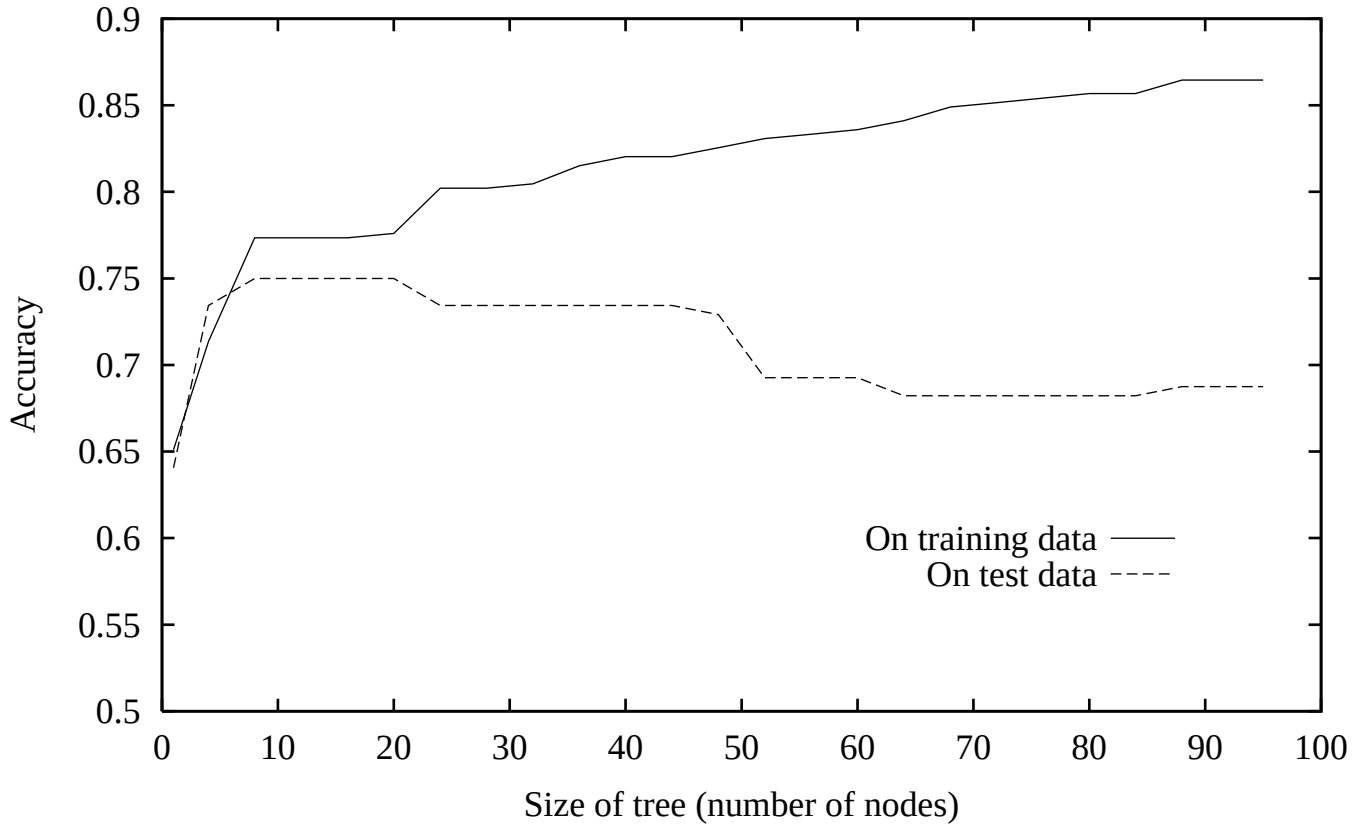
Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

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

and

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

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when the data split is not anymore statistically significant
- grow full tree, then post-prune

How to select the "best" tree:

- Measure performance over a separate validation data set
- Minimum Description Length (MDL) principle:
  minimize *size(tree) + size(misclassifications(tree))*

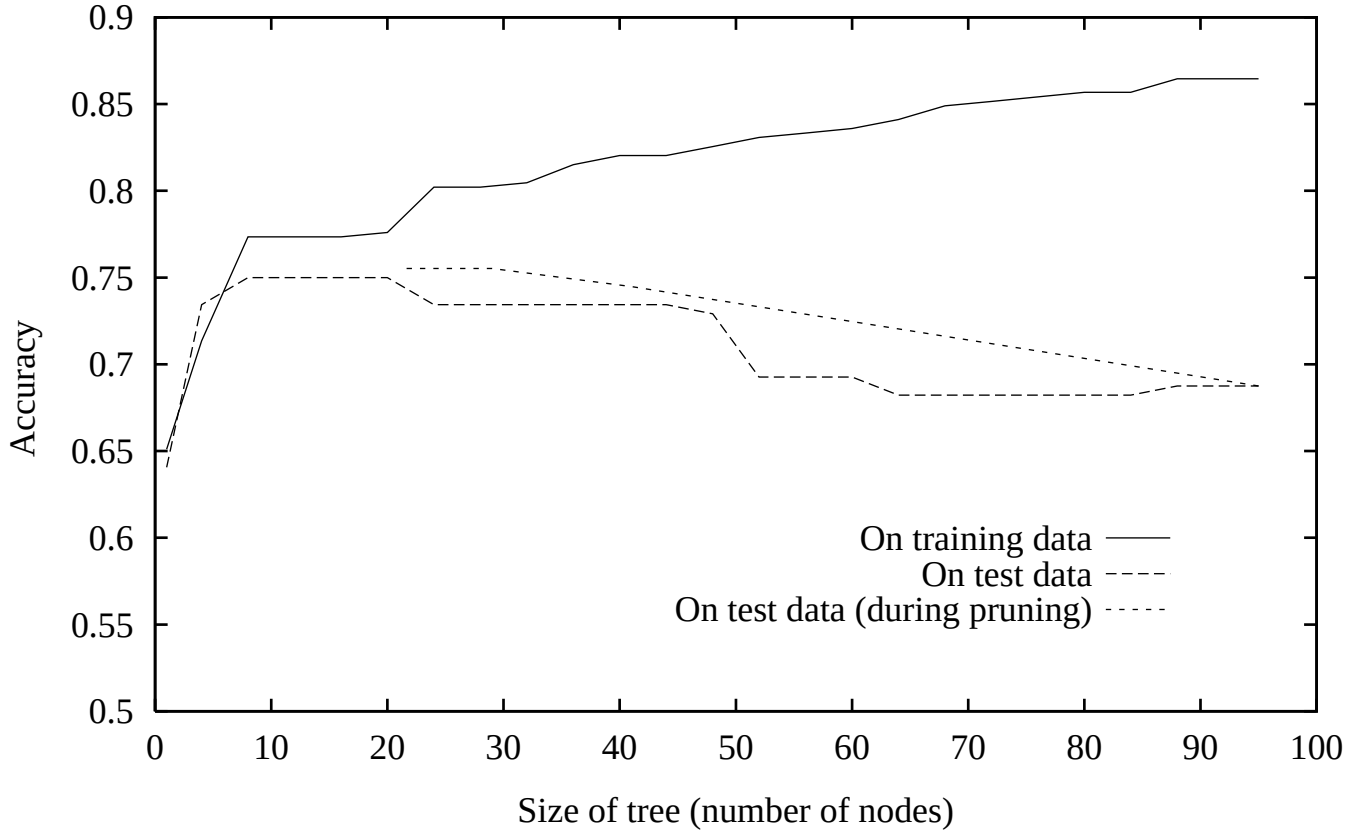
2.2.1 Reduced-Error Pruning

Split data into training set and validation set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy

Effect of Reduced-Error Pruning



Note:
A validation set (distinct from both the training and test sets) was used for pruning. Accuracy over this validation set is not shown here.

2.2.2 Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort the final rules into the desired sequence (e.g. according to the estimated accuracy) for use

It is perhaps most frequently used method (e.g., C4.5 by Ross quinlan, 1993)

3. Ensemble Learning: a very brief introduction

There exist several well-known meta-learning techniques that aggregate *decision trees*:

Boosting [Freund et al., 1995; Shapire et al., 1996]:

When constructing a new tree, the data points that have been incorrectly predicted by earlier trees are given some extra wight, thus forcing the learner to concentrate successively on more and more difficult cases.
In the end, a weighted vote is taken for prediction.

Bagging [Breiman, 1996]:

New trees do not depend on earlier trees; each tree is independently constructed using a boostrap sample (i.e. sampling with replacing) of the data set.
The final classification is done via simple majority voting.

The AdaBoost Algorithm

[pseudo-code from *Statistical Pattern Recognition*, Andrew Webb, Keith Copsey, 2011]

Input:

$\{(x_i, y_i) \mid i = 1, \ldots, n\}$ — **a set of labeled instances;**
$T \in \mathbb{N}^*$ — **the number of boosting rounds;**

Training:

initialization: $w_i = 1/n$, **for** $i = 1, \ldots, n$
for $t = 1, \ldots, T$

a. **construct a classifier** η_t **(e.g., a decision tree) using the given training data, with weights** $w_i, i = 1, \ldots, n$**;**

b. $e_t = \sum_i w_i$, **where** i **indexes all instances misclassified by** η_t**;**

c. **if** $e_t = 0$ **or** $e_t > 1/2$ **then terminate the procedure;**
else $w_i \leftarrow w_i \left(\frac{1}{e_t} - 1 \right)$ **for all instances which were misclassified by** η_t**, and then renormalize the weights** $w_i, i = 1, \ldots, n$ **so that they sum to** 1**;**

Prediction:

given a test instance x**, and**
assuming that the classifiers η_t **have two clases,** -1 **and** $+1$**, compute**
$\hat{\eta} = \sum_{t=1}^{T} \left(\log \left(\frac{1}{e_t} - 1 \right) \right) \eta_t(x)$**;**
assign x **the label** ***sign***$(\hat{\eta})$**;**

The Bagging Algorithm
(Bootstrap aggregating)

[pseudo-code from *Statistical Pattern Recognition*, Andrew Webb, Keith Copsey, 2011]

Input:

$\{(x_i, y_i) \mid i = 1, \ldots, n\}$ — a set of labeled instances;
$B \in \mathbb{N}^*$ — the number of samples/(sub)classifiers to be produced;

Training:

for $b = 1, \ldots, B$

a. generate a ***boostrap sample*** of size n by extracting with replacement from the training set;
(Note: some instances will be replicated, others will be omitted.)

b. construct a classifier η_b (e.g., a decision tree), using the boostrap sample as training data;

Prediction:

given a test instance x, assign it the most common label in the set $\{\eta_b(x) \mid b = 1, \ldots, B\}$;

Random Forests (RF)

[Breiman, 2001]

RF extends bagging with and additional layer of randomness:

random feature selection:

While in standard classification trees each node is split using the best split among all variables, in RF each node is split using the best among a subset of features randomly chosen at that node.

RF uses only two parameters:

- the number of variables in the random subset at each node;
- the number of trees in the forest.

This somehow counter-intuitive strategy is robust against overfitting, and it compares well to other machine learning techniques (SVMs, neural networks, discriminat analysis etc).

The Random Forests (RF) Algorithm

[pseudo-code from *Statistical Pattern Recognition*, Andrew Webb, Keith Copsey, 2011]

Input:

$\{(x_i, y_i) \mid i = 1, \ldots, n\}$ — **a set of labeled instances;**
$B \in \mathbb{N}^*$ — **the number of samples to be produced / trees in the forest;**
m — **the number of features to be selected**

Training:

for $b = 1, \ldots, B$

a. generate a *boostrap sample* of size n by extracting with replacement from the training set;
(Note: some instances will be replicated, others will be omitted.)

b. construct a a decision tree η_b by using the boostrap sample as training data, and choosing at each node the "best" among m randomly selected attributes;

Computation of the *out-the-bag error*:

a training instance x_i, is misclassified by RF if its label y_i differs from z_i, the most common label in the set $\{\eta_{b'}(x_i) \mid b' \in \{1, \ldots, B\}$, such that $x_i \notin$ the boostrap sample for the classifier $\eta_{b'}\}$;

Prediction:

given a test instance x, assign it the most common label in the set $\{\eta_b(x) \mid b = 1, \ldots, B\}$;