Lecture 2: Decision Trees

Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

- Learning through search (AI)

- Blind search
  - Impossible since the search space grows exponentially fast
- Greedy Search
  - Doesn't guarantee to find the best result

- Relation to Gradient Descent, local minima problem….

When to consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required

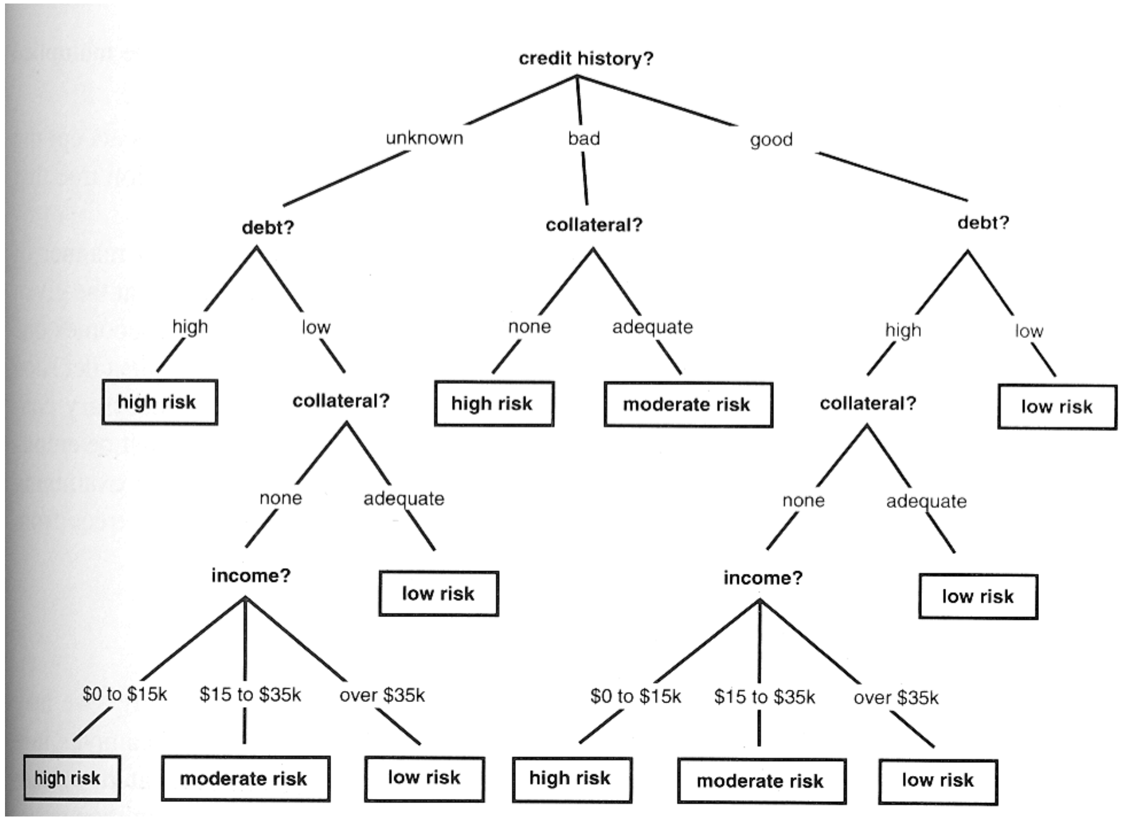
- Possibly noisy training data
- Missing attribute values

- Examples:
  - Medical diagnosis
  - Credit risk analysis
  - Object classification for robot manipulator (Tan 1993)

TABLE 12.1 DATA FROM CREDIT HISTORY OF LOAN APPLICATIONS

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	\$0 to \$15k
2.	high	unknown	high	none	\$15 to \$35k
3.	moderate	unknown	low	none	\$15 to \$35k
4.	high	unknown	low	none	\$0 to \$15k
5.	low	unknown	low	none	over \$35k
6.	low	unknown	low	adequate	over \$35k
7.	high	bad	low	none	\$0 to \$15k
8.	moderate	bad	low	adequate	over \$35k
9.	low	good	low	none	over \$35k
10.	low	good	high	adequate	over \$35k
11.	high	good	high	none	\$0 to \$15k
12.	moderate	good	high	none	\$15 to \$35k
13.	low	good	high	none	over \$35k
14.	high	bad	high	none	\$15 to \$35k

Decision tree for credit risk assessment



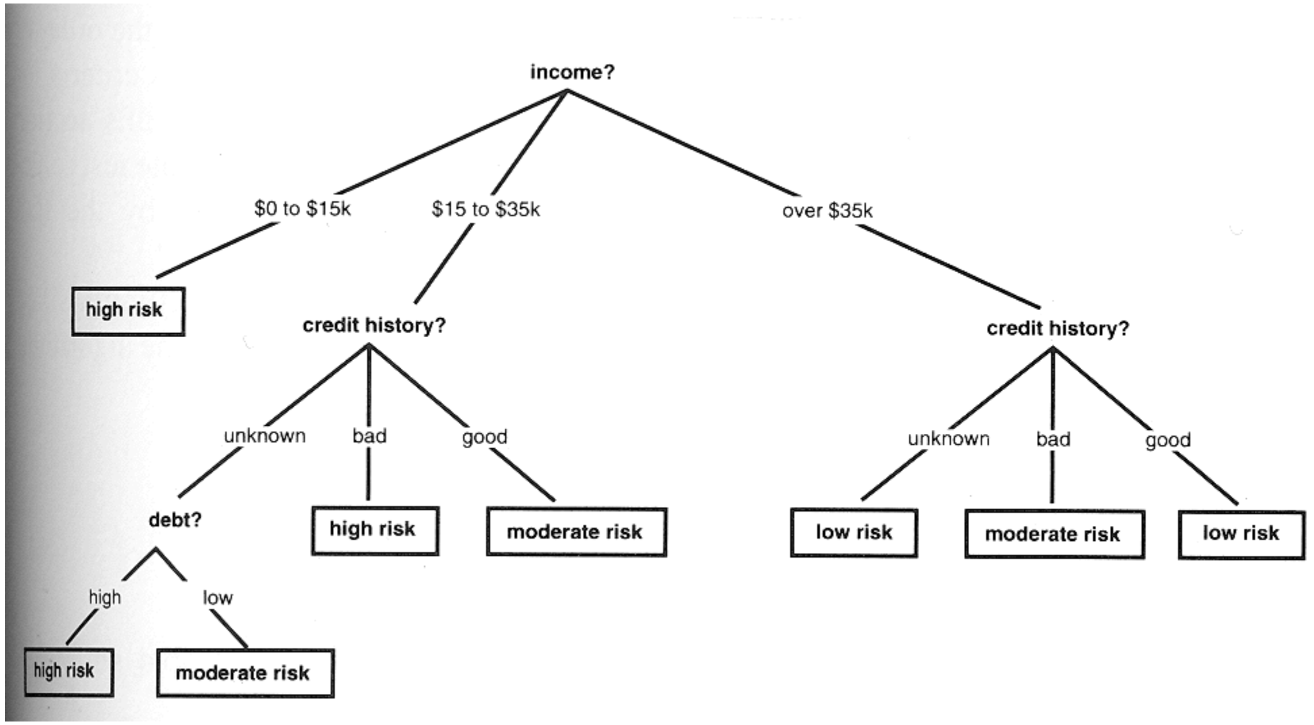
- The decision tree represents the classification of the table
- It can classify all the objects in the table
- Each internal node represents a test on some property
- Each possible value of that property corresponds to a branch of the tree
- An individual of unknown type may be classified be traversing this tree

- In classifying any given instance, the tree does not use all the properties in the table
- Decision tree for credit risk assessment

- If a person has a good credit history and low debit, we ignore her collateral income and classify her as low risk
- In spite of omitting certain tests, the tree classifies all examples in the table

- In general, the size of a tree necessary to classify a given set of examples varies according to the order with which properties (=attributes) are tested

- Given a set of training instances and a number of different decision trees that correctly classify the instances, we may ask which tree has the greatest likelihood of correctly classifying using instances of the population?



- This is a simplified decision tree for credit risk assessment
  - It classifies all examples of the table correctly

- ID3 algorithm assumes that a good decision tree is the simplest decision tree

- Heuristic:
  - Preferring simplicity and avoiding unnecessary assumptions
  - Known as Occam‘s Razor

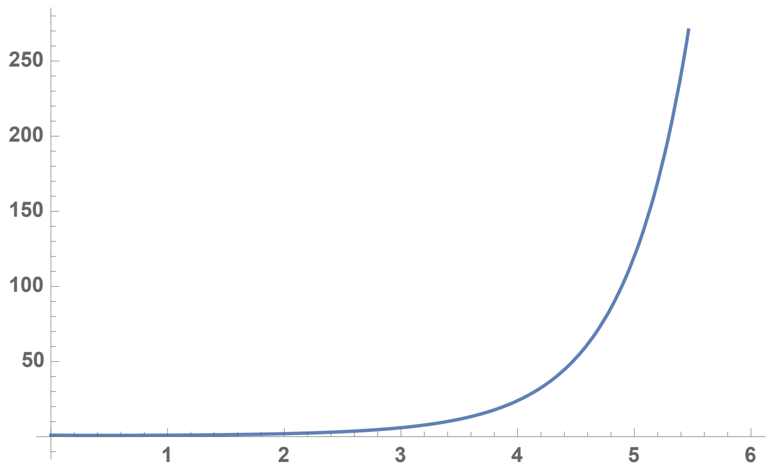
- The simplest decision tree that covers all examples should be the least likely to include unnecessary constraints

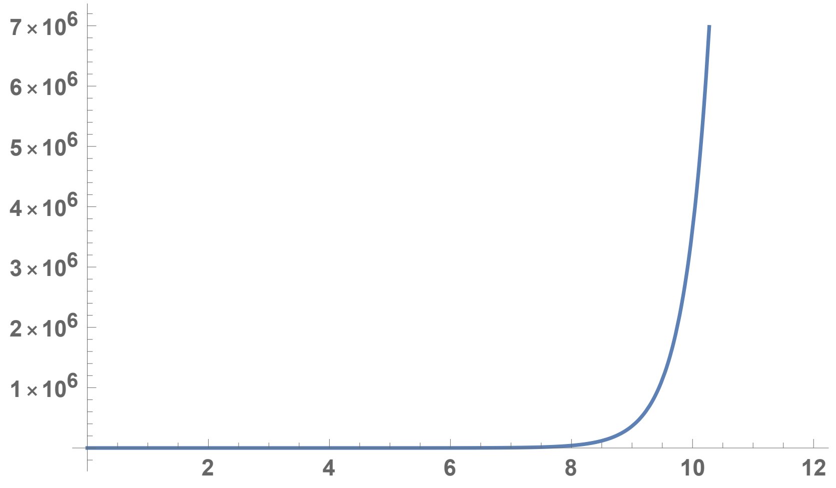
- Occam Razor was first articulated by the medieval logician William of Occam in 1324
    - born in the village of Ockham in Surrey (England) about 1285, believed that he died in a convent in Munich in 1349, a victim of the Black Death
    - It is vain do with more what can be done with less..
  - We should always accept the simplest answer that correctly fits our data
  - The smallest decision tree that correctly classifies all given examples

- Because the order of tests is critical to constructing a simple tree, ID3 relies heavily on its criteria for selecting the test at the root of each sub tree
- How many different decision tree exist?
- *N*=Number Of Attributes
- There exist *N!* different ordering of attributes, different decision trees
- Algorithm:
  - Blind Search finds the global minima, the smallest decision tree (optimal)
  - Compute all *N!* decision trees and chose the smallest one

N! grows extremely fast

- N=4, 4!=24   manageable, no problem





- Plot of *N!*

We have to use a heuristic function

- ID3 selects a property to test at the current node of the tree and uses this test to partition the set of examples
- The algorithm then recursively constructs a sub tree for each partition
- This continuous until all members of the partition are in the same class

- That class becomes a leaf node of the tree

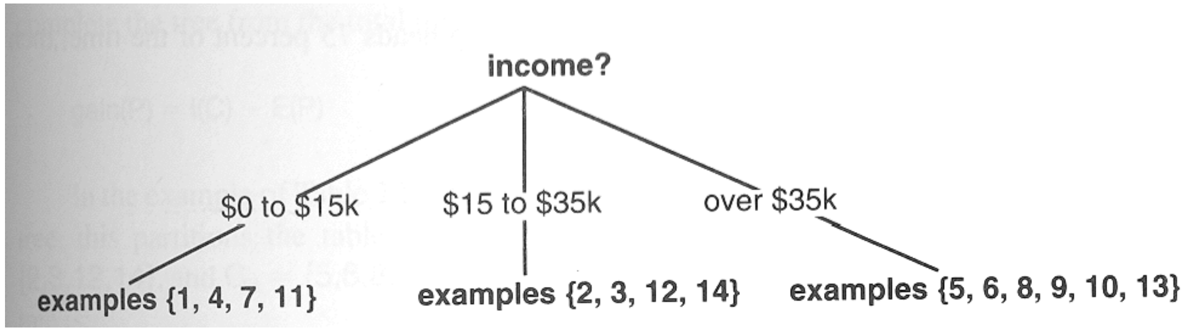
Top-Down Induction of Decision Trees ID3

1. A $\leftarrow$ the “best” decision attribute for next *node*
2. Assign A as decision attribute (=property) for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes

TABLE 12.1 DATA FROM CREDIT HISTORY OF LOAN APPLICATIONS

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

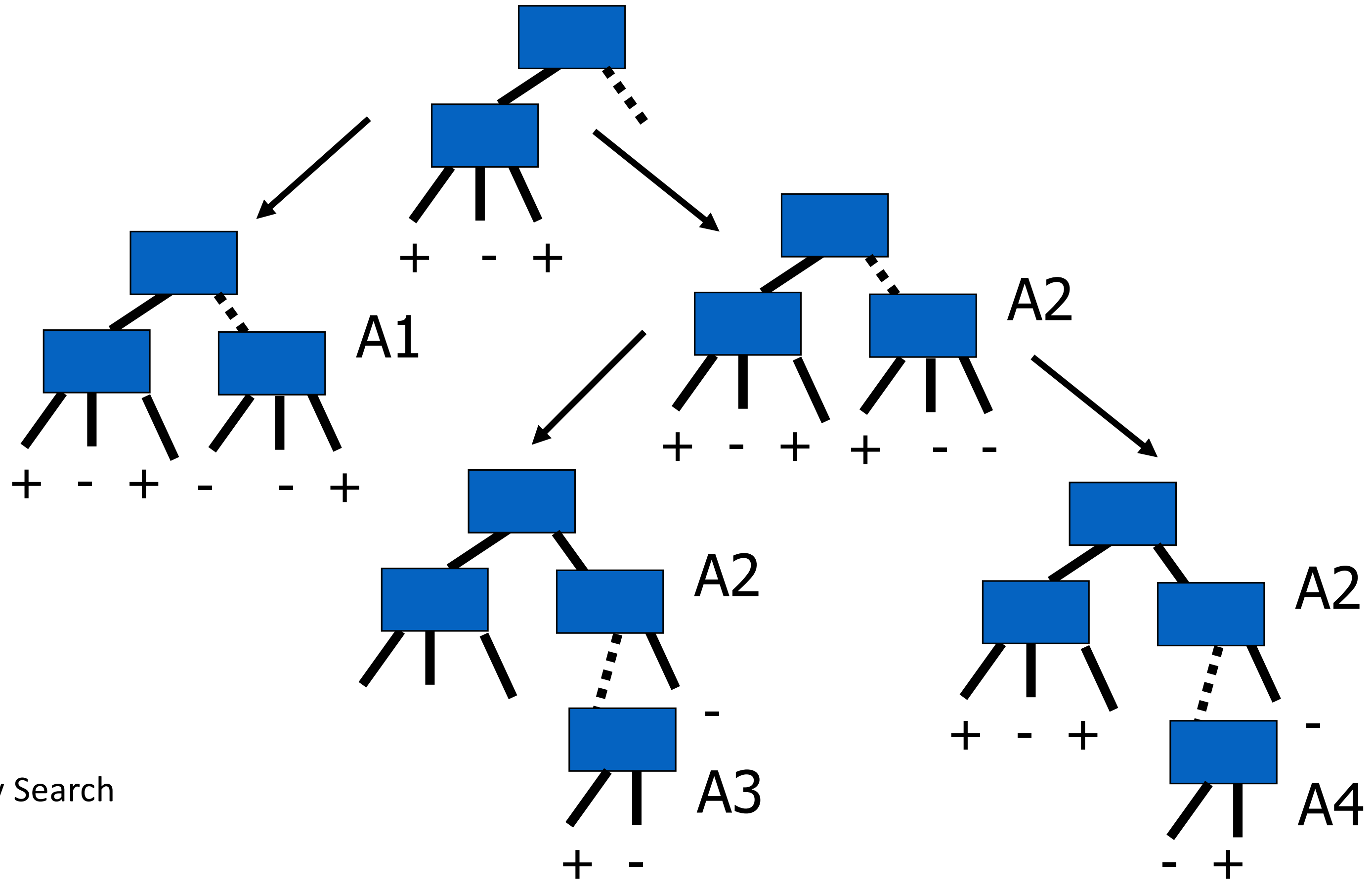
- ID3 constructs the tree for credit risk assessment
  - Beginning with the full table of examples, ID3 selects **income** as the root property using function selecting "best" property (attribute)
  - The examples are divided, listed by their number in the list



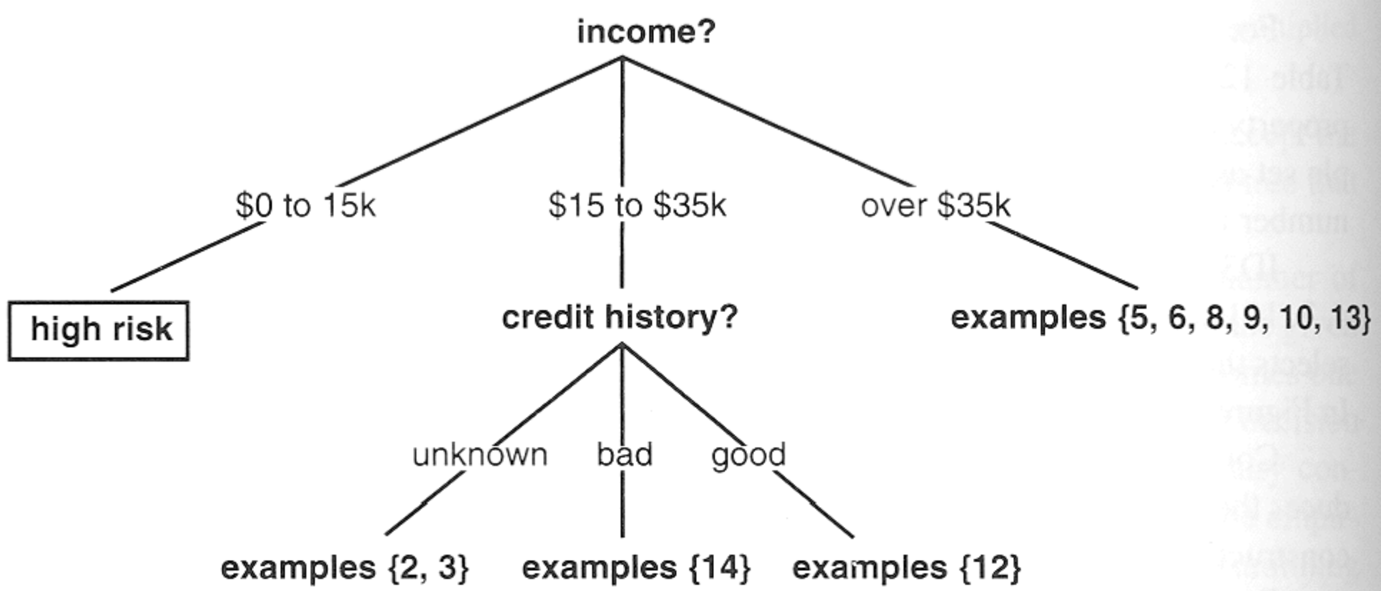
- ID3 applies the method recursively for each partition
  - The partition {1,4,7,11} consists entirely of high-risk individuals, a node is created
  - ID3 selects credit history property as the root of the subtree for the partition {2,3,12,14}
  - Credit history further divides this partition into {2,3},{14} and {12}
- ID3 implements a form of hill climbing in the space of all possible trees using a heuristic function
- Doesn’t guarantee to find the smallest decision tree, can find a local maxima.

Hypothesis Space Search ID3

two classes: +,-



Greedy Search


income?
$0 to 15k
$15 to $35k
over $35k
high risk
credit history?
examples {5, 6, 8, 9, 10, 13}
unknown
bad
good
examples {2, 3}
examples {14}
examples {12}

Heuristic function: Shannon Entropy

- Shannon formalized these intuitions
- Given a universe of messages $M=\{m_1,m_2,\ldots,m_n\}$ and a probability $p(m_i)$ for the occurrence of each message, the **information** content (also called entropy)of a message *M* is given

$$I(M)=\sum_{i=1}^{n}-p(m_i)\log_2(p(m_i))$$

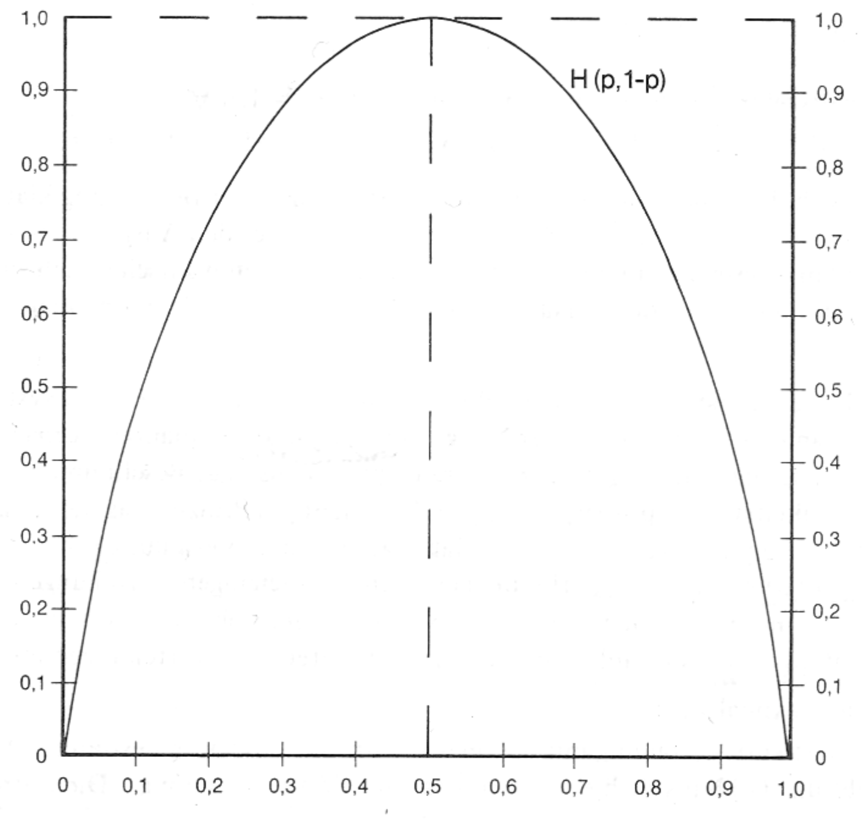
- Information content of a message telling the outcome of the flip of an honest coin

$$I(Coin\_toss) = -p(heads)\log_2(p(heads)) - p(tails)\log_2(p(tails))$$
$$I(Coin\_toss) = -p(0.5)\log_2(p(0.5)) - p(0.5)\log_2(p(0.5))$$
$$I(Coin\_toss) = 1 \qquad bit$$

- However if the coin has been rigged to come up heads 75 percent

$$
\begin{aligned}
I(Coin\_toss) &= -p(heads)\log_2(p(heads)) - p(tails)\log_2(p(tails)) \\
I(Coin\_toss) &= -p(0.75)\log_2(p(0.75)) - p(0.25)\log_2(p(0.25)) \\
I(Coin\_toss) &= 0.811 \qquad bits
\end{aligned}
$$

Only two probabilities



- We may think of a decision tree as conveying information about the classification of examples in the decision table

- The information content of the tree is computed from the probabilities of different classifications

- The credit history loan table has following information
  - *p(risk is high)=6/14*
  - *p(risk is moderate)=3/14*
  - *p(risk is low)=5/14*

$$I(credit\_table) = -\frac{6}{14}\log_2\left(\frac{6}{14}\right) - \frac{3}{14}\log_2\left(\frac{3}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$I(credit\_table) = 1.531 \quad bits$$

- For a given test, the information gain provided by making that test at the root of the current tree is equal to

- Total information of the table - the amount of information needed to complete the classification after performing the test
- The amount of information needed to complete the tree is defined as weighted average of the information content of each sub tree

- The amount of information needed to complete the tree is defined as weighted average of the information content of each sub tree by the percentage of the examples present
- *C* a set of training instances. If property (for example *income*) with *n* values, *C* will be divided into the **subsets** $\{C_1, C_2, .., C_n\}$
- Expected information needed to complete the tree after making *P* root

$$E(P) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} I(C_i)$$

$$E(P) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} I(C_i)$$

- The gain from the property P is computed by subtracting the expected information to complete E(P) fro the total information

$$gain(P) = I(C) - E(P)$$

- In the credit history loan table we make income the property tested at the root
- This makes the division into
  - *C_1={1,4,7,11},C_2={2,3,12,14},C_3={5,6,8,9,10,13}*

$$E(income) = \frac{4}{14} I(C_1) + \frac{4}{14} I(C_2) + \frac{6}{14} I(C_3)$$

$$E(income) = \frac{4}{14} 0 + \frac{4}{14} 1.0 + \frac{6}{14} 0.65$$

$$E(income) = 0.564 \qquad bits$$

gain(income)=I(credit_table)-E(income)

gain(income)=1.531-0.564

gain(income)=0.967 bits

gain(credit history)=0.266

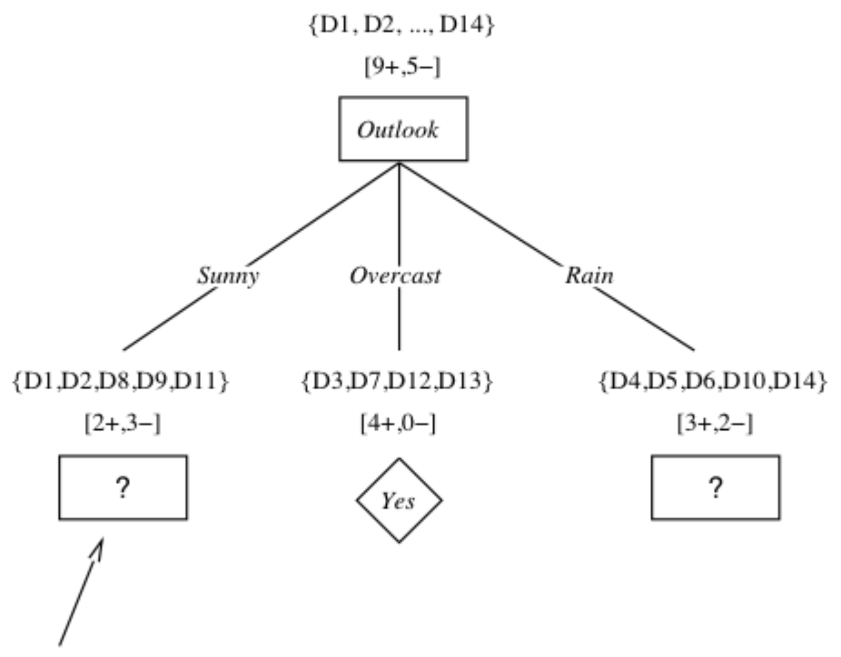
gain(debt)=0.581

gain(collateral)=0.756

- Because income provides the greatest information gain, ID will select it as the root of the tree

- The algorithm continues to apply this analysis recursively to each **subtree**, until it has completed the tree.

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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 be tested here?

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

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

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

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

Overfiting

- The ID3 algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples
- Difficulties may be present:
  - When there is noise in the data
  - When the number of training examples is too small to produce a representative sample of the true target function
- The ID3 algorithm can produce trees that **overfit** the training examples

- We will say that a hypothesis overfits the training examples - if some other hypothesis that fits the training examples *less well* actually performs better over the *entire* distribution of instances (included instances beyond training set)

Overfitting

Consider error of hypothesis h over

- Training data: $\text{error}_{\text{train}}(h)$
- Entire distribution D of data: $\text{error}_D(h)$

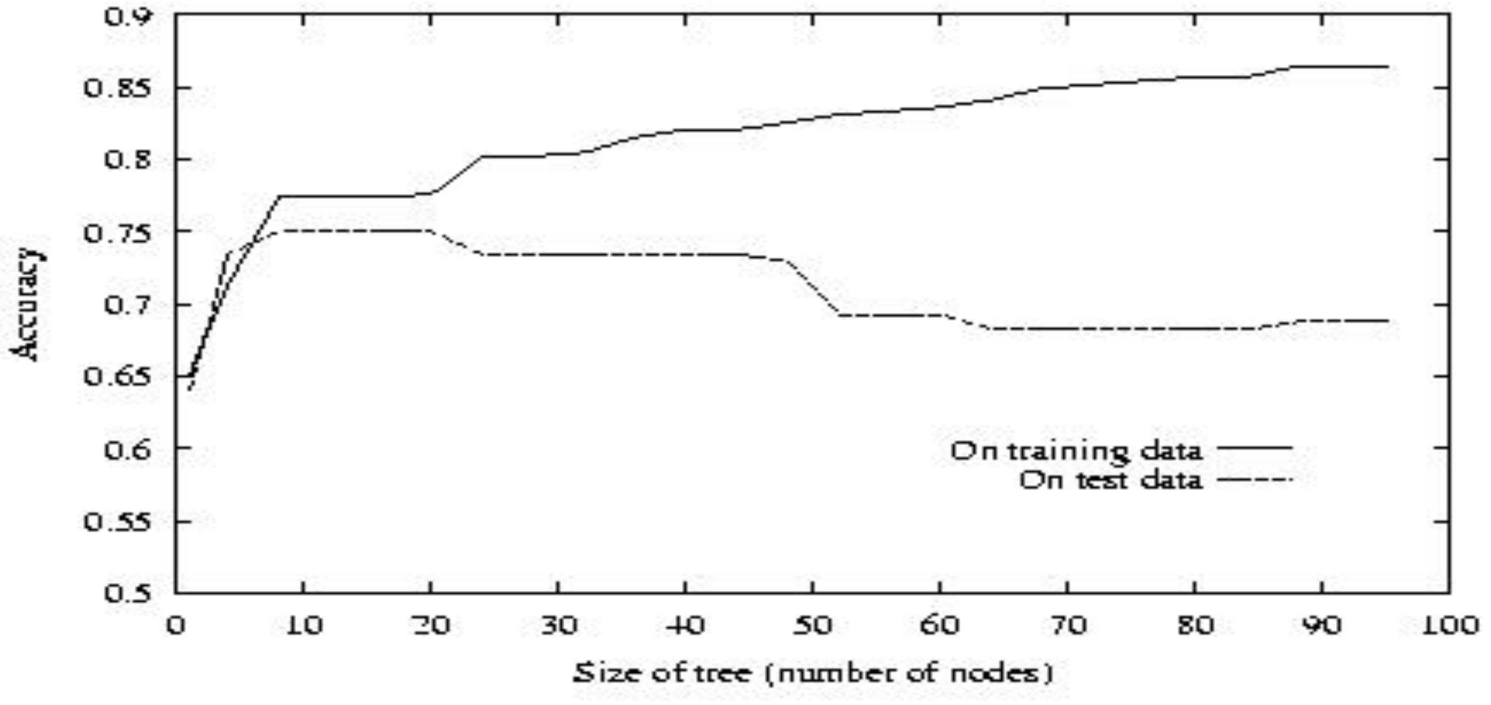
Hypothesis $h \in H$ *overfits* training data if there is an alternative hypothesis $h' \in H$ such that

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$

and

$$\text{error}_D(h) > \text{error}_D(h')$$

Overfitting

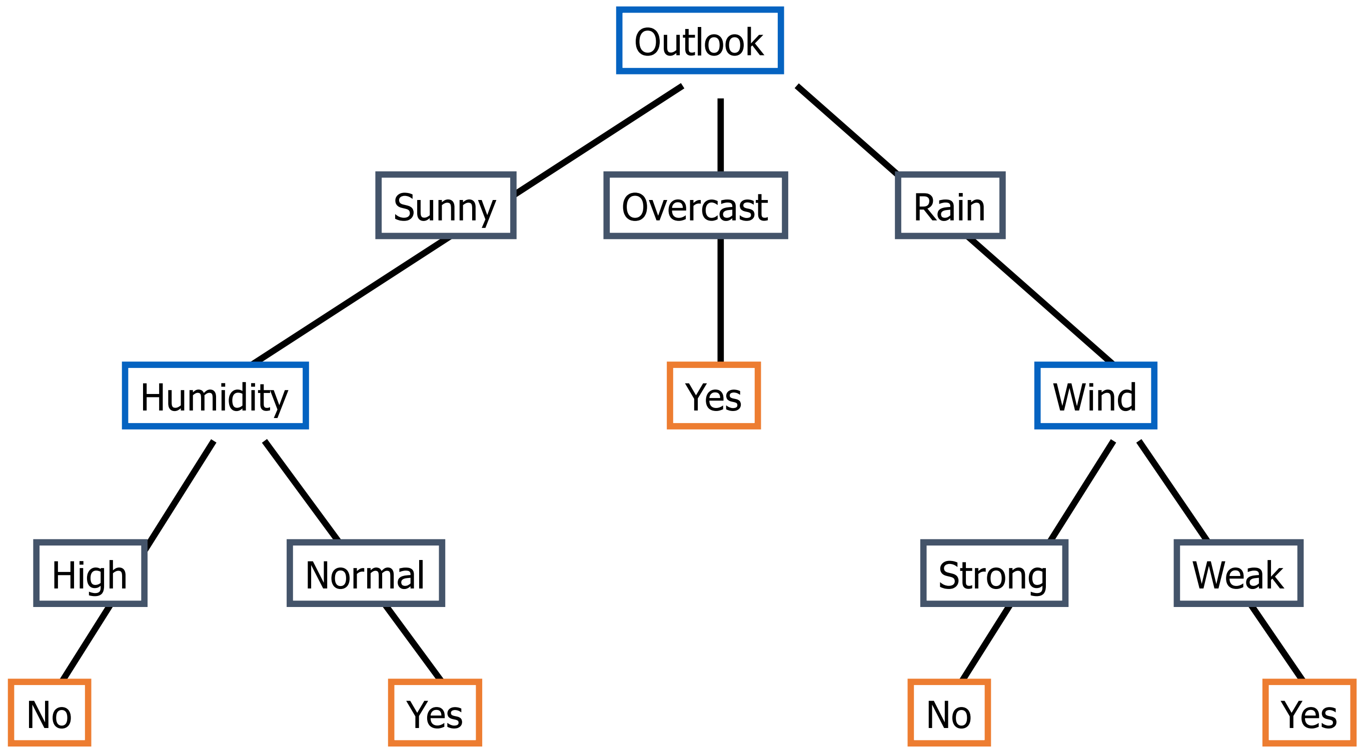


- How can it be possible for a tree h to fit the training examples better than *h′*, but to perform more poorly over subsequent examples
- One way this can occur when the training examples contain random errors or noise

Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

Decision Tree for PlayTennis



- Consider of adding the following positive training example, incorrectly labaled as negative
- *Outlook=Sunny, Temperature=Hot, Humidty=Normal, Wind=Strong, PlayTenis=No*
- The addition of this incorrect example will now cause ID3 to construct a more complex tree
- Because the new example is labeled as a negative example, ID3 will search for further refinements to the tree

- As long as the new erroneous example differs in some attributes, ID3 will succeed in finding a tree
- ID3 will output a decision tree *(h)* that is more complex then the original tree *(h‘)*

- Given the new decision tree a simple consequence of fitting noisy training examples, h‘ will outperform h on the test set

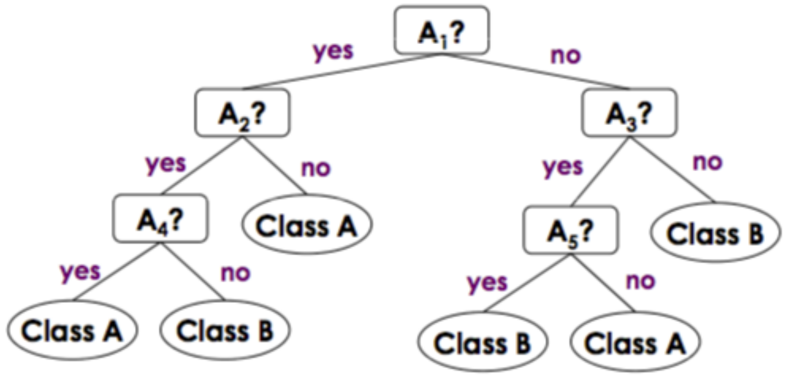
Avoid Overfitting

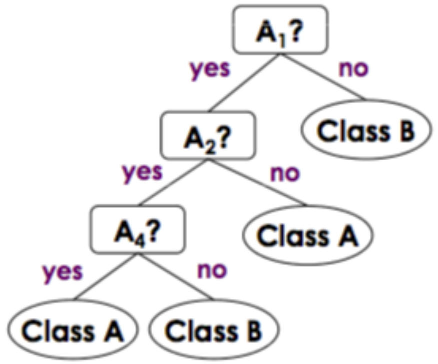
- How can we avoid overfitting?
  - Stop growing when data split not statistically significant
  - Grow full tree then post-prune

- How to select ``best'' tree:
  - Measure performance over training data
  - Measure performance over separate validation data set

Pruning

- Remove the least reliable branches





Quinlan strategies of C4.5

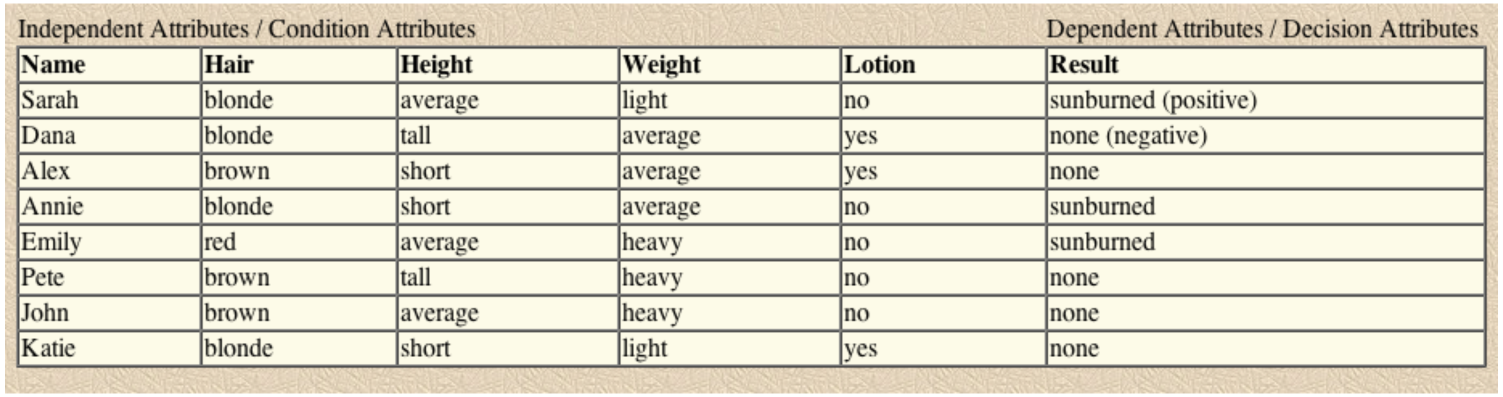
- Derive an initial rule set by enumerating paths from the root to the leaves
- Generalize the rules by possible deleting conditions deemed to be unnecessary
- Group the rules into subsets according to the target classes they cover
  - Delete any rules that do not appear to contribute to overall performance on that class
- Order the set of rules for the target classes, and chose a default class to which cases will be assigned

- The resultant set of rules will probably not have the same coverage as the decision tree
- Its accuracy should be equivalent
- Rules are much easier to understand
- Rules can be tuned by hand by an expert

From Trees to Rules

Once an identification tree is constructed, it is a simple matter to concert it into a set of equivalent rules

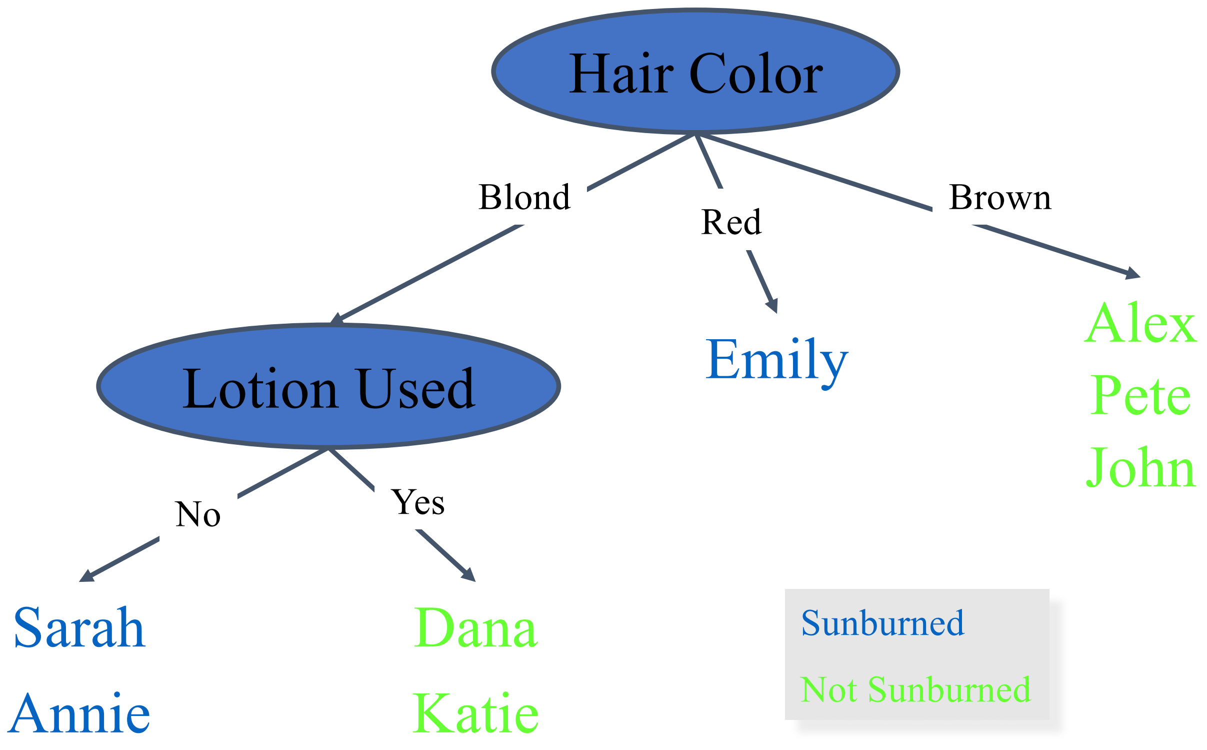
- Example from Artificial Intelligence, P.H. Winston 1992



Independent Attributes / Condition Attributes — Dependent Attributes / Decision Attributes

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned (positive)
Dana	blonde	tall	average	yes	none (negative)
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

An ID3 tree consistent with the data



Corresponding rules

If the person‘s hair is blonde
and the person uses lotion
then nothing happens

If the person‘s hair color is blonde
and the person uses no lotion
then the person turns red

If the person‘s hair color is red
then the person turns red

If the person‘s hair color is brown
then nothing happens

Unnecessary Rule Antecedents should be eliminated

If the person‘s hair is blonde
and the person uses lotion
then nothing happens

Are both antecedents are really necessary?
Dropping the first antecedents produce a rule with the same results

If the the person uses lotion
then nothing happens

- To make such reasoning easier, it is often helpful to construct a **contingency table**
- it shows the degree to which a result is contingent on a property

- In the following contingency table one sees the number of *lotion users* who are *blonde* and *not blonde* and are *sunburned* or *not*
  - Knowledge about whether a person is blonde has no bearing whether it gets sunburned

	Not sunburned	Sunburned
Person is blonde *(uses lotion)*	2	0
Person is not blonde *(uses lotion)*	1	0

- Check for lotion for the same rule

	Not sunburned	Sunburned
Person uses lotion	2	0
Person uses no lotion	0	2

- Has a bearing on the result

Unnecessary Rules should be Eliminated

If the person uses lotion
then nothing happens

If the person‘s hair color is blonde
and the person uses no lotion
then the person turns red

If the person‘s hair color is red
then the person turns red

If the person‘s hair color is brown
then nothing happens

- Note that two rules have a consequent that indicate that a person will turn red, and two that indicate that nothing happens

- One can replace either the two of them by a **default rule**

Default rule

If the person uses lotion
then nothing happens

If the person‘s hair color is brown
then nothing happens

If no other rule applies
then the person turns red

What is CART?

- Classification And Regression Trees
- Developed by Breiman, Friedman, Olshen, Stone in early 80’s.
    - Introduced tree-based modeling into the statistical mainstream
    - Rigorous approach involving cross-validation to select the optimal tree
- One of many tree-based modeling techniques.
    - CART -- the classic
    - CHAID
    - C5.0
    - Software package variants (SAS, S-Plus, R…)
    - Note: the “rpart” package in “R” is freely available

Idea: Recursive Partitioning

- Take all of your data.
- Consider *all* possible values of *all* variables.
- Select the variable/value **(X=t_1)** that produces the greatest "separation" in the target.
    - **(X=t_1)** is called a "split".
- If $X< t_1$ then send the data to the "left"; otherwise, send data point to the "right".
- Now repeat same process on these two "nodes"
    - You get a "tree"
    - Note: CART only uses *binary* splits.

Gini Index

- The Gini Index (used in CART) measures the impurity of a data partition ***D***

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- ***m***: the number of classes
- $\boldsymbol{p_i}$: the probability that a tuple in *D* belongs to class C_i

- Gini index of the flip of an honest coin
  - *$1 - (0.5^2 + 0.5^2) = 0.5$*
- Coin has been rigged to come up heads 75 percent
  - *$1 - (0.75^2 + 0.25^2) = 0.375$*

- The Gini Index considers a **binary split** for each attribute **A**, say D1 and D2. The **Gini index** of D given that partitioning is:

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

- A weighted sum of the impurity of each partition
  - Weighted mean, the same as before…..

- The reduction in impurity is given by

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity is chosen as the splitting attribute

Binary Split: Continuous-Valued Attributes

- **D**: a data partition
- Consider attribute **A** with continuous values
- To determine the best binary split on **A**

- What to examine?
- Examine each possible split point
  The midpoint between each pair of (sorted) adjacent values is taken as a possible split-point

How to examine?

- For each split-point, compute the weighted sum of the impurity of each of the two resulting partitions
    - (*D1: A<= split-point, D2: A > split-point)*

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

- The split-point that gives the minimum Gini index for attribute *A* is selected as its splitting subset

Binary Split: Discrete-Valued Attributes

- **D**: a data partition
- Consider attribute **A** with **v** outcomes ***$\{a_1 \ldots, a_v\}$***
- To determine the best binary split on **A**
  - Examine the partitions resulting from all possible subsets of ***$\{a_1 \ldots, a_v\}$***
  - Each subset ***S_A*** is a binary test of attribute ***A*** of the form ***"$A \in S_A$?"***
  - ***2^v*** possible subsets. We exclude the power set and the empty set, then we have ***2^v-2*** subsets

How to examine?

- For each subset, compute the weighted sum of the impurity of each of the two resulting partitions

$$Gini_A(D) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

- The subset that gives the minimum Gini index for attribute A is selected as its splitting subset

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Compute the Gini index of the training set D: 9 tuples in class yes and 5 in class no

$$Gini(D) = 1 - \left(\left(\frac{9}{14} \right)^2 + \left(\frac{5}{14} \right)^2 \right) = 0.459$$

Using attribute income: there are three values: **low, medium** and **high**

Choosing the subset {low, medium} results in two partions:

- **D1 (income $\in$ {low, medium}):** 10 tuples
- **D2 (income $\in$ {high}):** 4 tuples

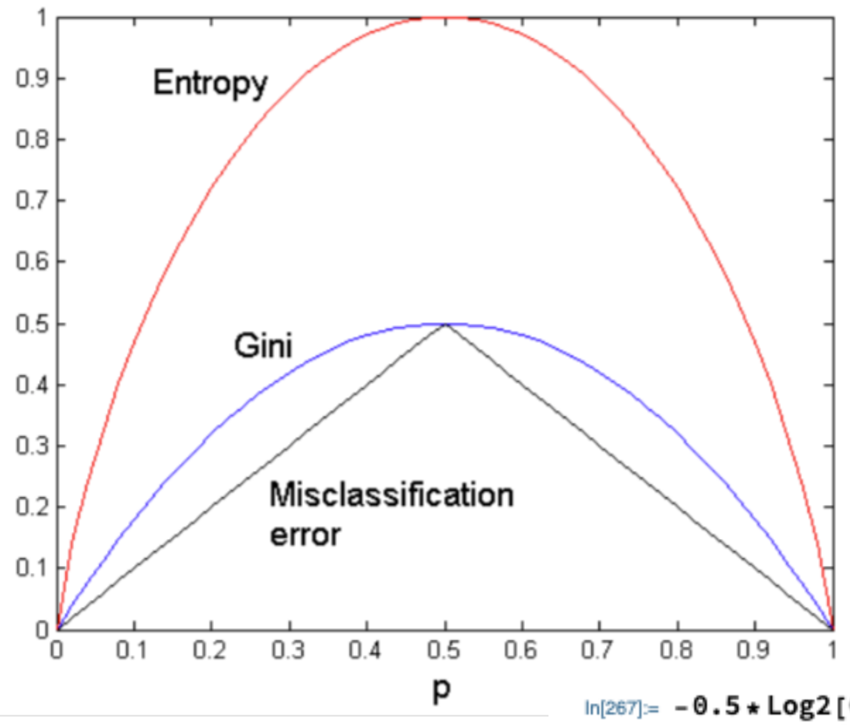
$$
\begin{aligned}
Gini_{income \in \{low, median\}}(D) &= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
&= 0.450 \\
&= Gini_{income \in \{high\}}(D)
\end{aligned}
$$

The Gini Index measures of the remaining partitions are:

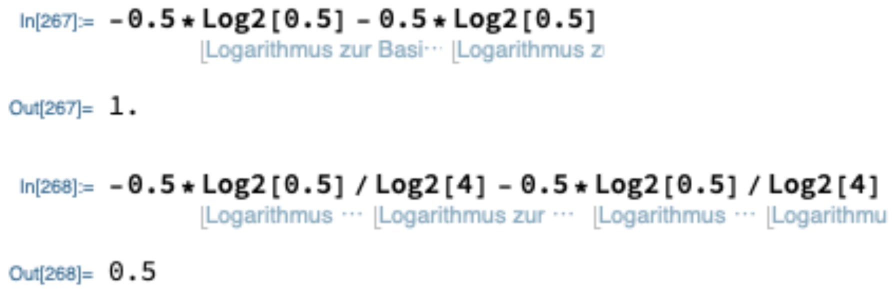
$$
\begin{aligned}
Gini_{\{low, high\} \, and \, \{medium\}}(D) &= 0.315 \\
Gini_{\{medium, high\} \, and \, \{low\}}(D) &= 0.300
\end{aligned}
$$

Therefore, the best binary split for attribute income is on {medium, high} and {low}

For a 2-class problem:



- *This argument does not halt ☺ :*



```
In[267]:= -0.5*Log2[0.5] - 0.5*Log2[0.5]

Out[267]= 1.

In[268]:= -0.5*Log2[0.5] / Log2[4] - 0.5*Log2[0.5] / Log2[4]

Out[268]= 0.5
```

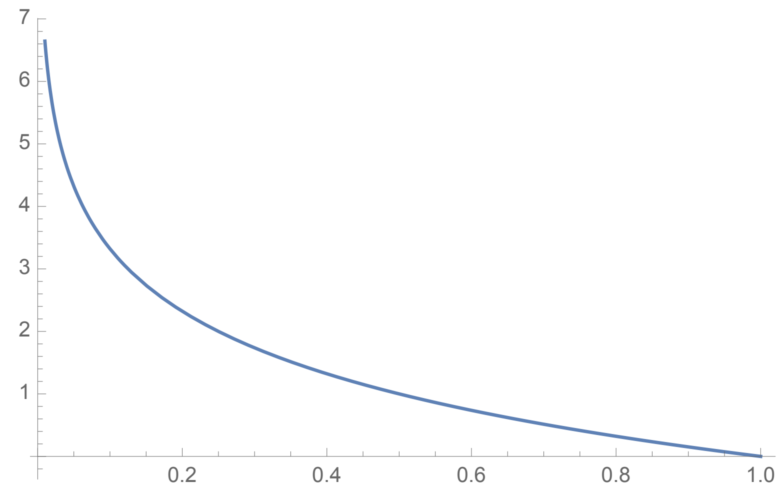
Shannon Information vs. Gini

- $-0.5*Log_2[0.5] - 0.5*Log_2[0.5]=-Log_2[0.5]=1$
- $-Log_2[0.25]=2$
- *…*
- $-Log_2[1/256]=8$

- $1 - (0.5^2 + 0.5^2)=0.5$
- $1 - (0.25^2 + 0.25^2 + 0.25^2 + 0.25^2)=0.75$
- $1 - (256*(1/256)^2=0.996094$

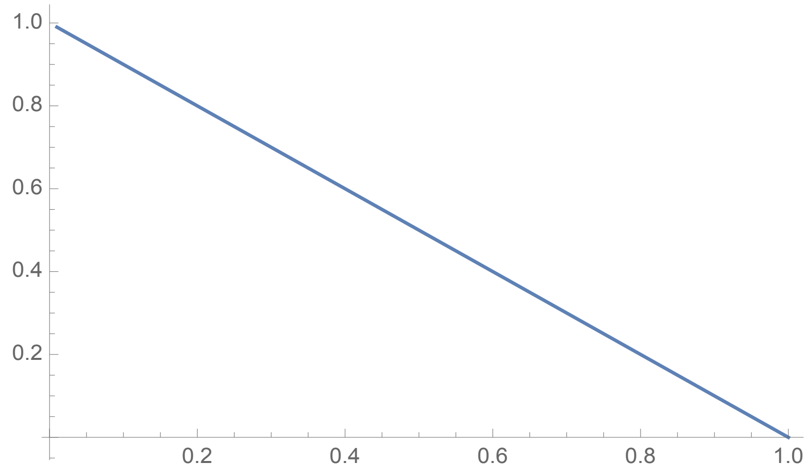
Shannon’s entropy for equal distribution vs. Gini…

```
Plot[-Log2[x], {x, 0.01, 1}]
```



Shannon’s entropy grows nonlinear! Does not converge

```
Plot[1 - ((1/x)*x^2), {x, 0.01, 1}]
```



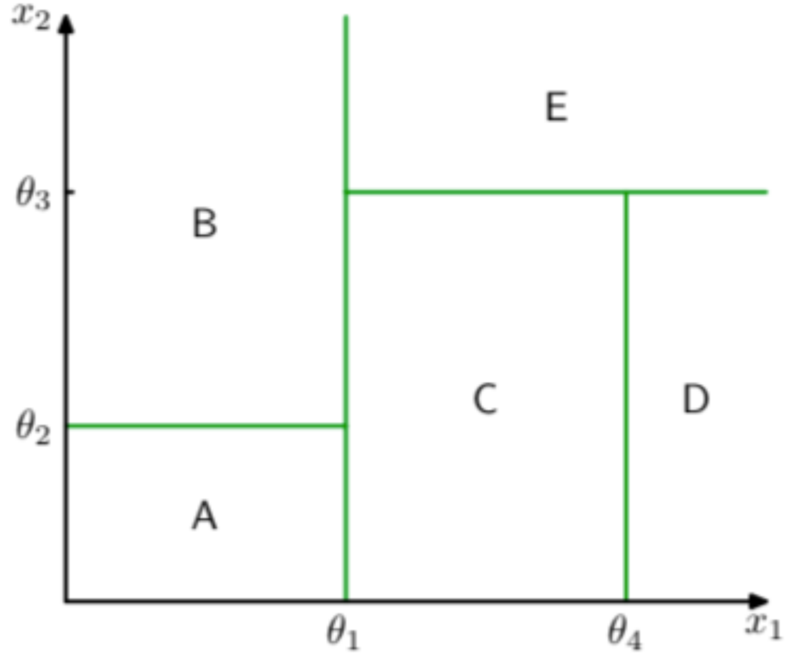
Gini grows linear, converges to one

Comparing Attribute Selection Measures

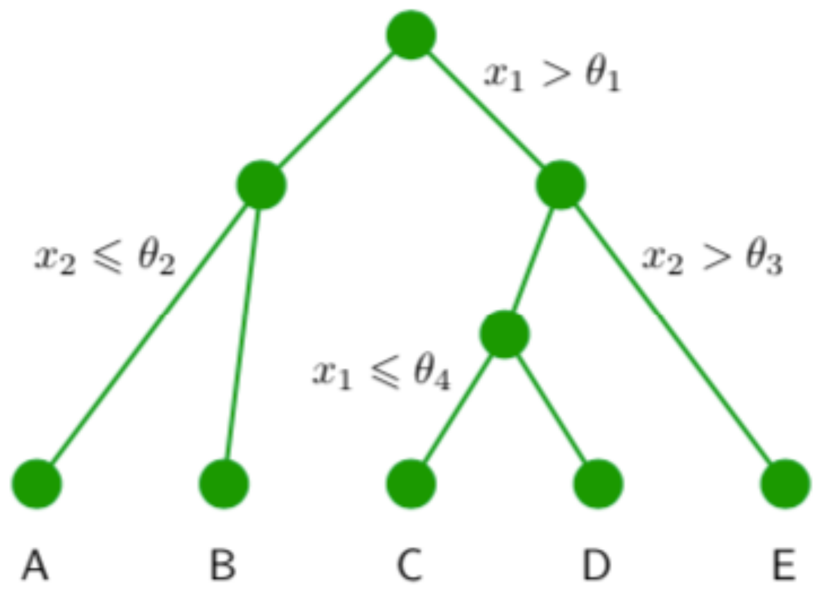
- **Information Gain**
    - Biased towards **multivalued** attributes
    - ***Multivalued*** Splits

- **Gini Index**
    - Biased towards *multivalued* attributes
    - Has difficulties when the number of classes is large
    - Tends to favor tests that result in equal-sized partitions and purity in both partitions
    - Only uses ***binary*** splits

Binary Tree



x_2
θ_3
θ_2
E
B
C
D
A
θ_1
θ_4
x_1



$x_1 > \theta_1$
$x_2 \leqslant \theta_2$
$x_2 > \theta_3$
$x_1 \leqslant \theta_4$
A
B
C
D
E

Attributes with Cost

Consider:

- Medical diagnosis : *blood test* costs 1000 secs
- Robotics: *width_from_one_feet* has cost 23 secs.

How to learn a consistent tree with low expected cost?

Replace *Gain* by :

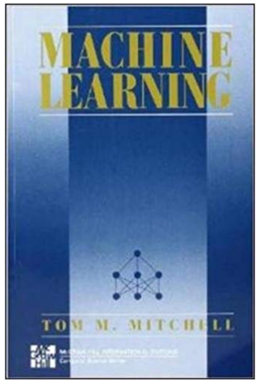
$Gain^2(A)/\mathbf{Cost(A)}$ [Tan, Schimmer 1990]

$2^{Gain(A)}-1/(Cost(A)+1)^w \; w \in [0,1]$ [Nunez 1988]

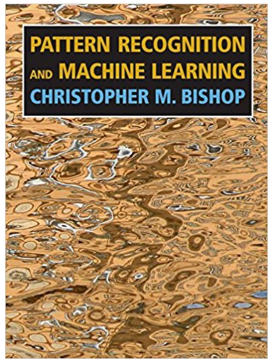
Next:

Probability and Information

Literature

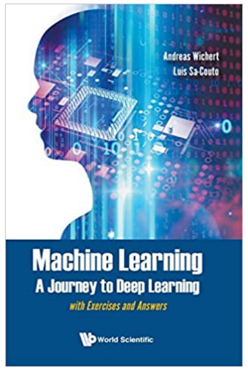


- Tom M. Mitchell, Machine Learning, McGraw-Hill; 1st edition (October 1, 1997)
  - Chapter 3



- Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer 2006
  - Section 14.4

Literature



- Machine Learning - A Journey to Deep Learning, A. Wichert, Luis Sa-Couto, World Scientific, 2021
  - Chapter 2, Section 2.5
    - C Decision Trees