Machine Learning - Lecture # 1

Bruce M. McLaren

Senior Researcher, DFKI

Slides courtesy of

Erica Melis, Russell/Norvig, Tom Mitchell

Overview

Introduction to Machine Learning

- Including a Brief Sales Pitch for a DFKI project …

Supervised Inductive Learning

Decision Trees

Overfitting

Ensemble Learning

What is Machine Learning?

- Computer algorithms used to make predictions about new data given old data
- Well-established and thriving sub-field of Artificial Intelligence since the 1970s

Why Machine Learning?

- Growing flood of online data - Data Mining!
  - The process of discovering patterns in data
  - Explosion of access to digital data has made data mining an important practical field since the 1990s
- Computational power is available
- Progress in algorithms and theory
- Software applications we can't program by hand
  - e.g., autonomous driving, analysis of large amounts of data
- Self-customizing programs
  - Newsreader that learns user interests

How are Data Mining and Machine Learning related?

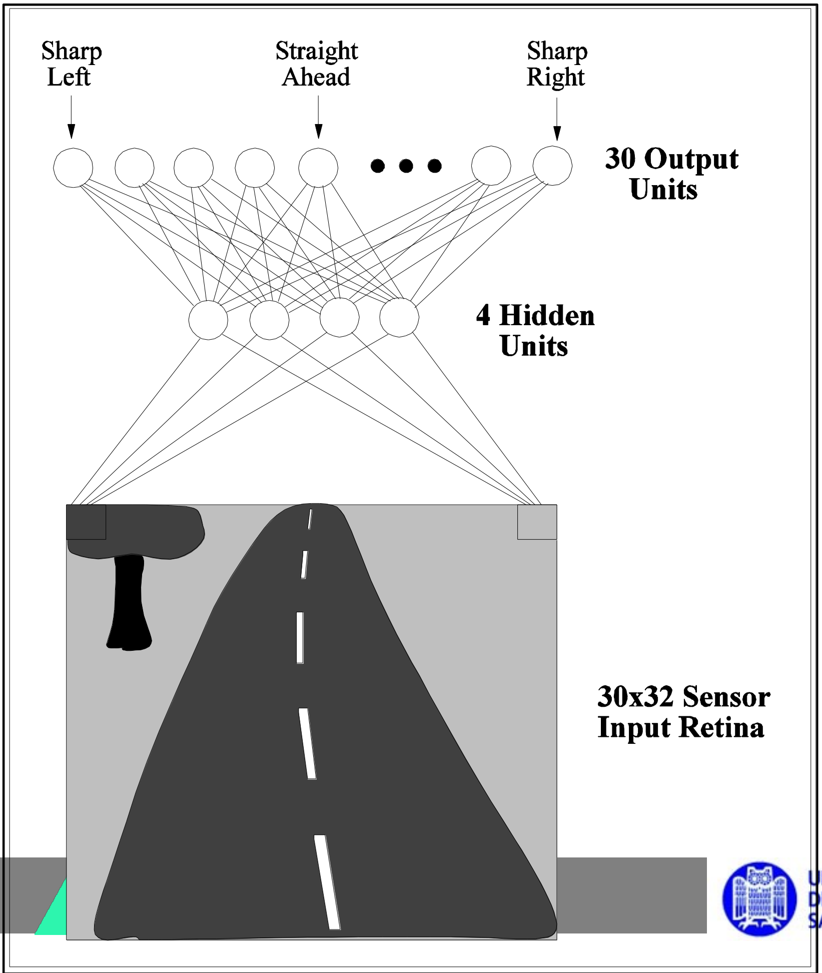
- Data Mining unifies ideas from:
  - AI & Machine Learning
    - Problems cast as search through a space of possible hypotheses
    - Symbolic representation of concepts
    - Rule-based representations
  - Statistics
    - Probability
    - Bayesian statistics
    - Regression analysis
- Interesting historical note …
  - One of the most famous and commonly used data mining methods, *decision trees*, was independently developed by statisticians (Breiman et al) and an AI researcher (Quinlan) in the 1970s and 80s

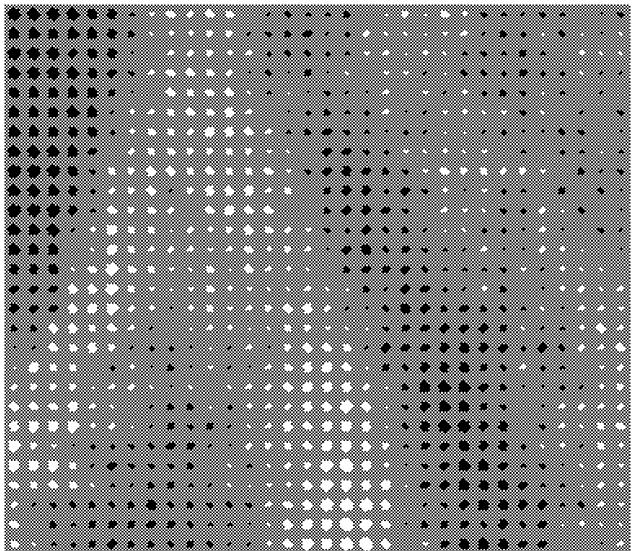
Some Practical Success Stories

- Driving Autonomous Vehicles - ALVINN
- Human Speech Recognition
- Handwriting recognition
- Fraudulent Use of Credit Cards
- Predict Stock Rates
- Intelligent Elevator Control
- World champion Backgammon
- Robot Soccer
- DNA Classification

Problems Too Difficult to Program by Hand

ALVINN is a perception system which learns to control the NAVLAB vehicles by "watching" a person drive. ALVINN's architecture consists of a single hidden layer back-propagation network.
ALVINN drives 70 mph on highways





UNIVERSITÄT DES SAARLANDES

Credit Risk Analysis

Customer103: (time=t0)	*Customer103:* (time=t1)	...	*Customer103:* (time=tn)
Years of credit: 9	Years of credit: 9		Years of credit: 9
Loan balance: $2,400	Loan balance: $3,250		Loan balance: $4,500
Income: $52k	Income: ?		Income: ?
Own House: Yes	Own House: Yes		Own House: Yes
Other delinquent accts: 2	Other delinquent accts: 2		Other delinquent accts: 3
Max billing cycles late: 3	Max billing cycles late: 4		Max billing cycles late: 6
Profitable customer?: ?	Profitable customer?: ?		**Profitable customer?: No**
...	...		...

```
If    Other-Delinquent-Accounts > 2, and
      Number-Delinquent-Billing-Cycles > 1
Then  Profitable-Customer? = No
      [Deny Credit Card application]

If    Other-Delinquent-Accounts = 0, and
      (Income > $30k)  OR  (Years-of-Credit > 3)
Then  Profitable-Customer? = Yes
      [Accept Credit Card application]
```

Typical Data Mining Task

Customer103: (time=t0)	*Customer103:* (time=t1)	...	*Customer103:* (time=tn)
Sex: M	Sex: M		Sex: M
Age: 53	Age: 53		Age: 53
Income: $50k	Income: $50k		Income: $50k
Own House: Yes	Own House: Yes		Own House: Yes
MS Products: Word	MS Products: Word		MS Products: Word
Computer: 386 PC	Computer: Pentium		Computer: Pentium
Purchase Excel?: ?	Purchase Excel?: ?		**Purchase Excel?: Yes**
...	...		...

Given:

- 9714 patient records, each describing a pregnancy and birth
- Each patient record contains 215 features

Learn to predict:

Classes of future patients at high risk for Emergency Cesarean Section

Data Mining Result

Patient103 time=1 →	*Patient103* time=2 ... →	*Patient103* time=n
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C-Section: ?	Elective C-Section: no	Elective C-Section: no
Emergency C-Section: ?	Emergency C-Section: ?	**Emergency C-Section: Yes**
...	...	...

```
IF   No previous vaginal delivery, and
     Abnormal 2nd Trimester Ultrasound, and
     Malpresentation at admission
THEN Probability of Emergency C-Section is 0.6

 Over training data: 26/41 = .63,
 Over test data: 12/20 = .60
```

Interesting DFKI Project: ARGUNAUT

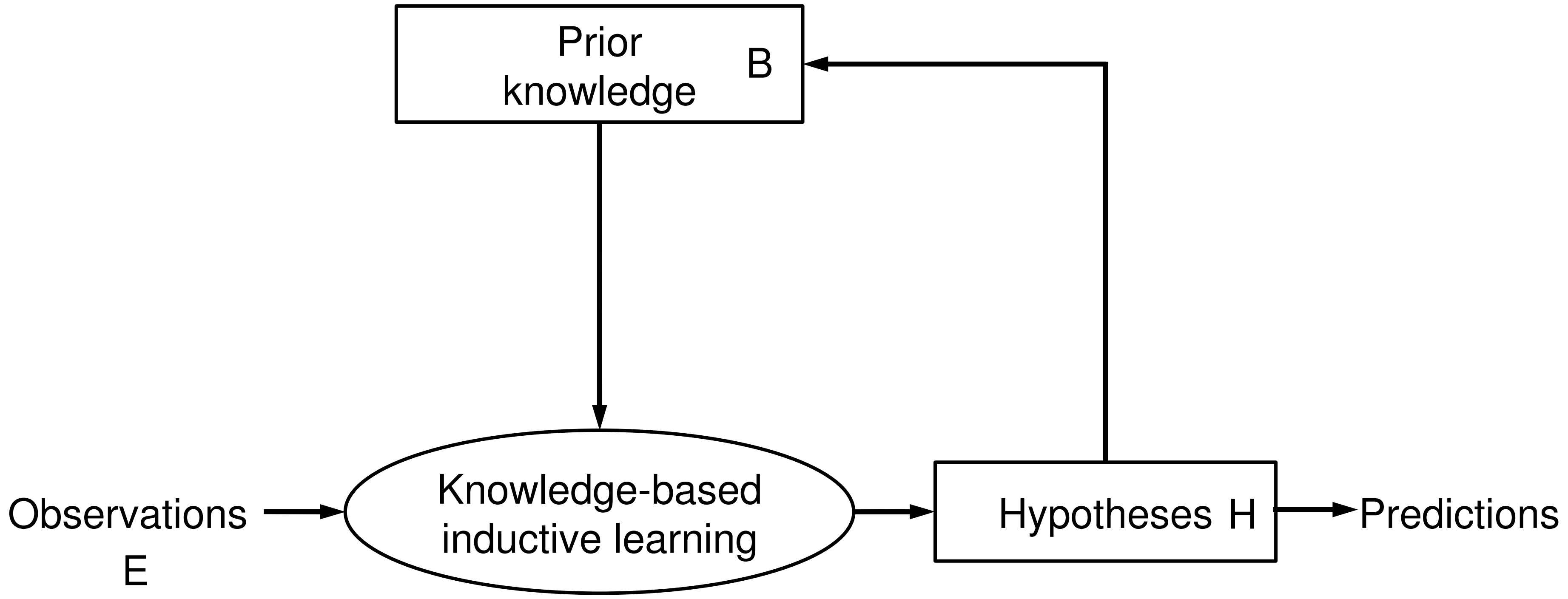
- *Educational Data Mining* - the ARGUNAUT Project
  - Use of Machine Learning Techniques to learn collaborative / argumentative behaviors of students
  - Use the resulting classifiers to support the moderator of an online collaborative discussion
- Looking for an enthusiastic student / HIWI to work on this project
  - Learn to use modern Machine Learning tools
  - Integrating data and text mining
  - Work with a team of learning scientists and PhD students
  - Academic publications!
- Talk to me after the lecture or send email to **bmclaren@dfki.de**

Machine Learning Techniques

Lots and lots of approaches!

- Decision tree learning
- Artificial neural networks
- Naive Bayes
- Bayesian Nets
- Instance-based learning
- Reinforcement learning
- Genetic algorithms
- Support vector machines
- Explanation-Based Learning
- Inductive logic programming

How does an Agent learn?



What is the Learning Problem?

Learning = Improving with experience at some task

Let’s work on representing and formalizing this…

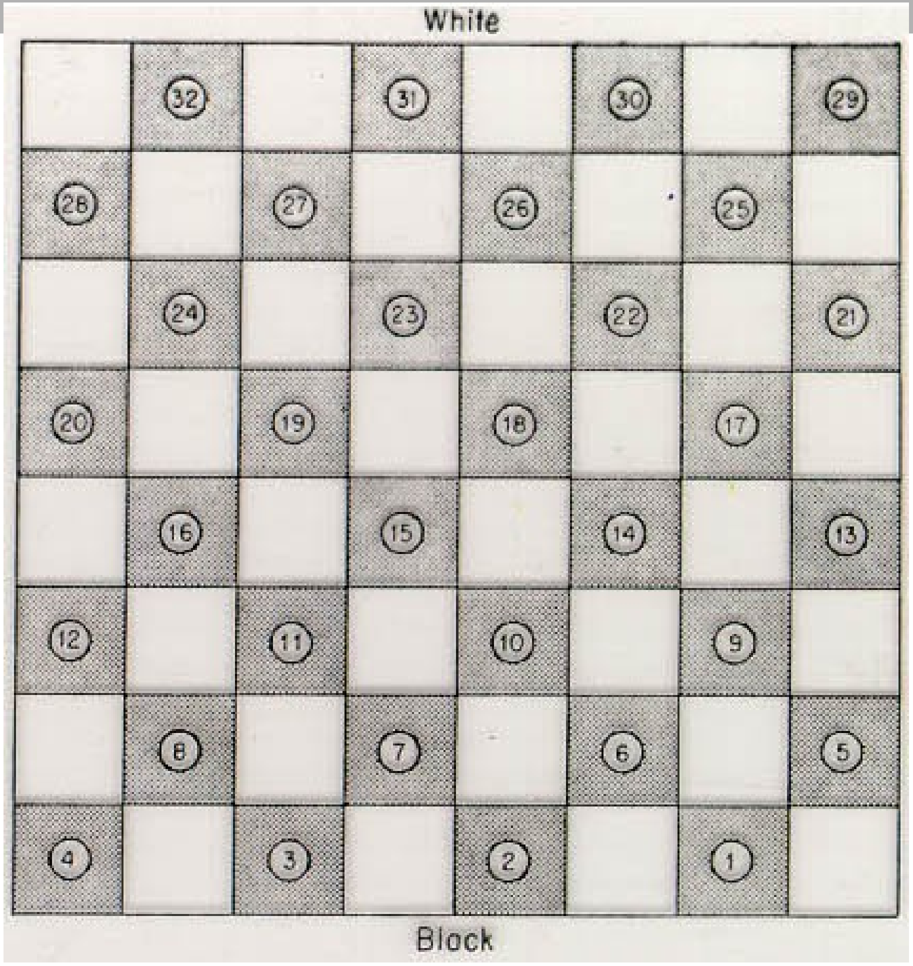
- Improve over Task T
- With respect to performance measure P
- Based on experience E
- Using the function V to decide what to do

How about an example to illustrate?

The Game of Checkers

Samuel's Checkers program (1959, 1967) - First significant learning program of any kind…

The program was an example of what today would be called "temporal-difference learning," a type of reinforcement learning.



Learning to Play Checkers

- T (Task): Play checkers
- P (Performance Measure): Percent of games won in world tournament.
- E (Experience): Games played against self.
- V (Learned Function): Evaluation of possible next moves

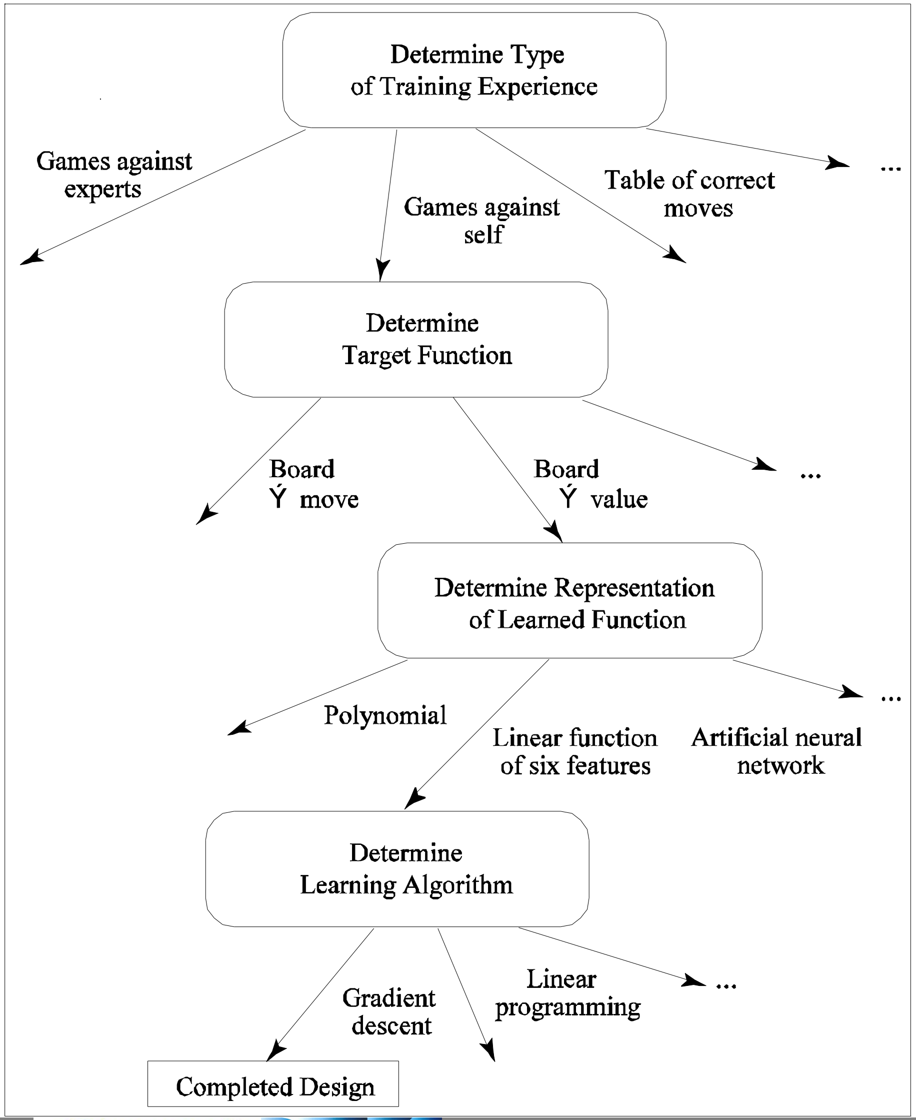
What exactly should be learned?

How shall it be represented?

What specific algorithm to learn it?

Design Choices for Checker Learning

These design choices need to be made for any learning problem…


Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board Ý move
Board Ý value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

A Learned Function *V'(b)* - *Weighted Linear Function*

Target function: V: Board $\longrightarrow$ IR

Target function representation:

$$V'(b) = w_0 + (w_1 * x1) + (w_2 * x_2) + (w_3 * x_3) + (w_4 * x_4) + (w_5 * x_5) + (w_6 * x_6)$$

- x_1: number of black pieces on board *b*
- x_2: number of red pieces on board *b*
- x_3: number of black kings on board *b*
- X_4: number of red kings on board *b*
- x_5: number of red pieces threatened by black (i.e., those that can be taken on black's next turn)
- x_6: number of black pieces threatened by red

Function Approximation Algorithm

- $V(b)$: the true target function
- $V'(b)$: the learned function
- $V_{train}(b)$: the training value
- $(b, V_{train}(b))$ training example

One rule for estimating training values:

$V_{train}(b) \leftarrow V'(\mathit{Successor}(b))$ for intermediate b

Choose Weight Tuning Rule

LMS Weight update rule:

Do repeatedly:

1. Select a training example *b* at random

2. Compute *error(b)* with current weights

$$error(b) = V_{train}(b) - V'(b)$$

3. For each board feature x_i, update weight w_i:

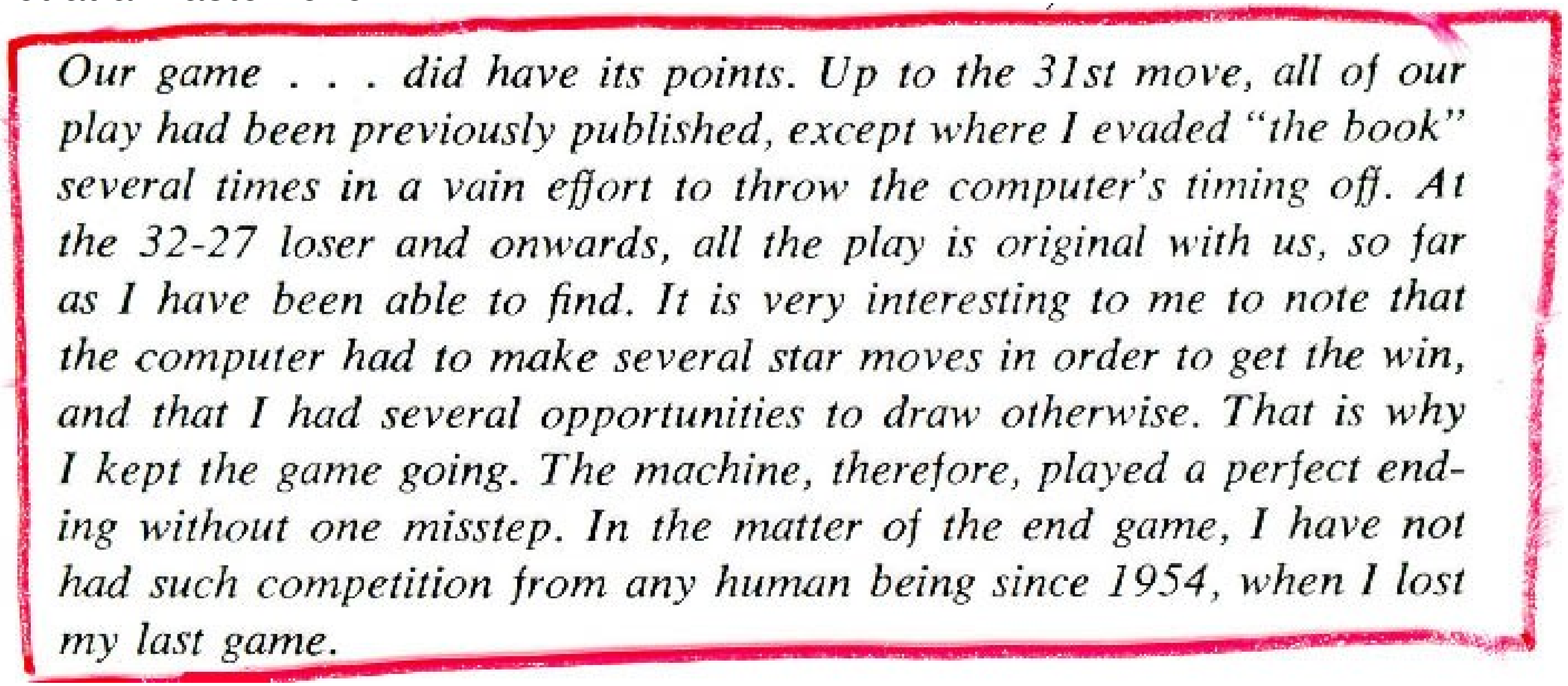
$$w_i \leftarrow w_i + c * x_i * error(b)$$

c is small constant to moderate the rate of learning

Samuel's checkers player using the generalization learning method approached “better-than-average” play.

Fairly good amateur opponents characterized it as “tricky but beatable” (Samuel, 1959).

A later version (Samuel, 1967) included refinements in its search procedure and hierarchical lookup tables. This version learned to play much better than the 1959 program, though still not at a master level.



Our game . . . did have its points. Up to the 31st move, all of our play had been previously published, except where I evaded “the book” several times in a vain effort to throw the computer's timing off. At the 32-27 loser and onwards, all the play is original with us, so far as I have been able to find. It is very interesting to me to note that the computer had to make several star moves in order to get the win, and that I had several opportunities to draw otherwise. That is why I kept the game going. The machine, therefore, played a perfect ending without one misstep. In the matter of the end game, I have not had such competition from any human being since 1954, when I lost my last game.

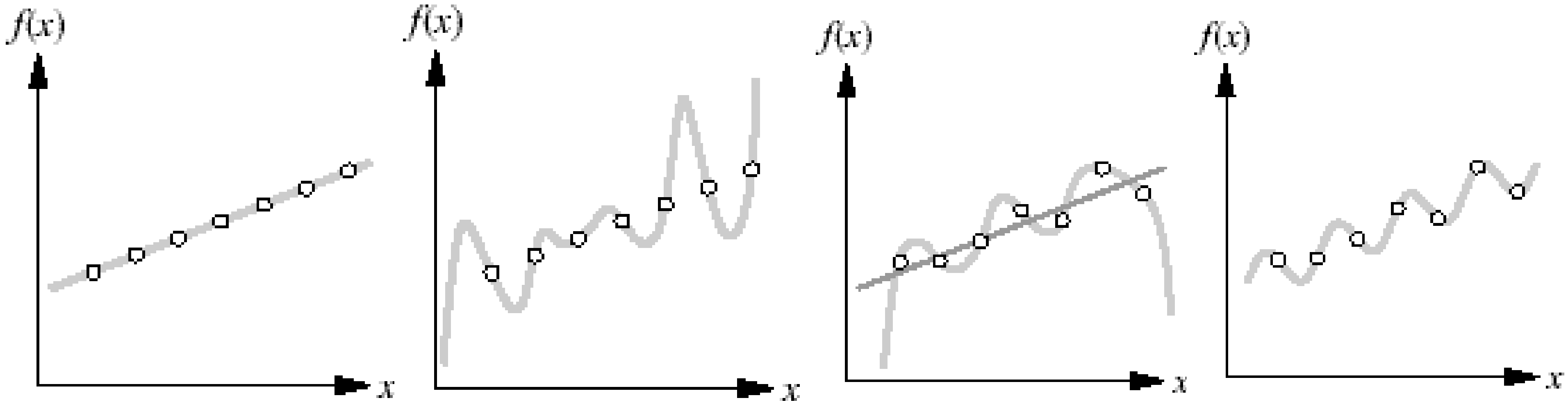
...A.L. Samuel

Overview

Supervised Inductive Learning (1)

Why is learning difficult?

- inductive learning generalizes from specific examples
  - cannot be proven true; it can only be proven false
- not easy to tell whether hypothesis *h* is a good approximation of a target function *f*
- complexity of hypothesis – fitting data



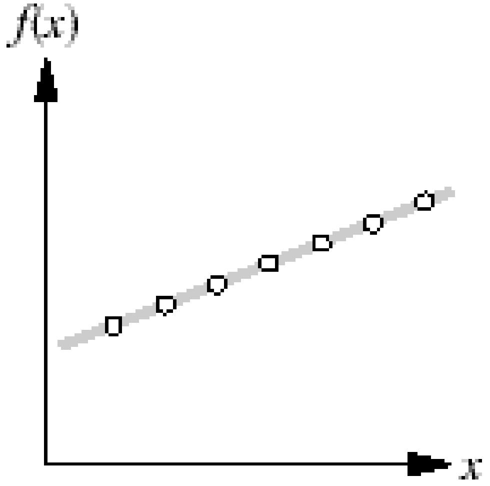
Supervised Inductive Learning (2)

To generalize beyond the specific examples, one needs constraints or biases on what *h* is best.

For that purpose, one has to specify

- the overall class of candidate hypotheses → restricted hypothesis space bias
- a metric for comparing candidate hypotheses to determine whether one is better than another → preference bias (Ockham‘s Razor - Simplest!)

So, under this bias, choose the linear function…



Supervised Inductive Learning (3)

Having fixed the bias, learning can be viewed as search in the hypothesis space guided by the used preference bias.

Overview

Introduction to Machine Learning

Supervised Inductive Learning

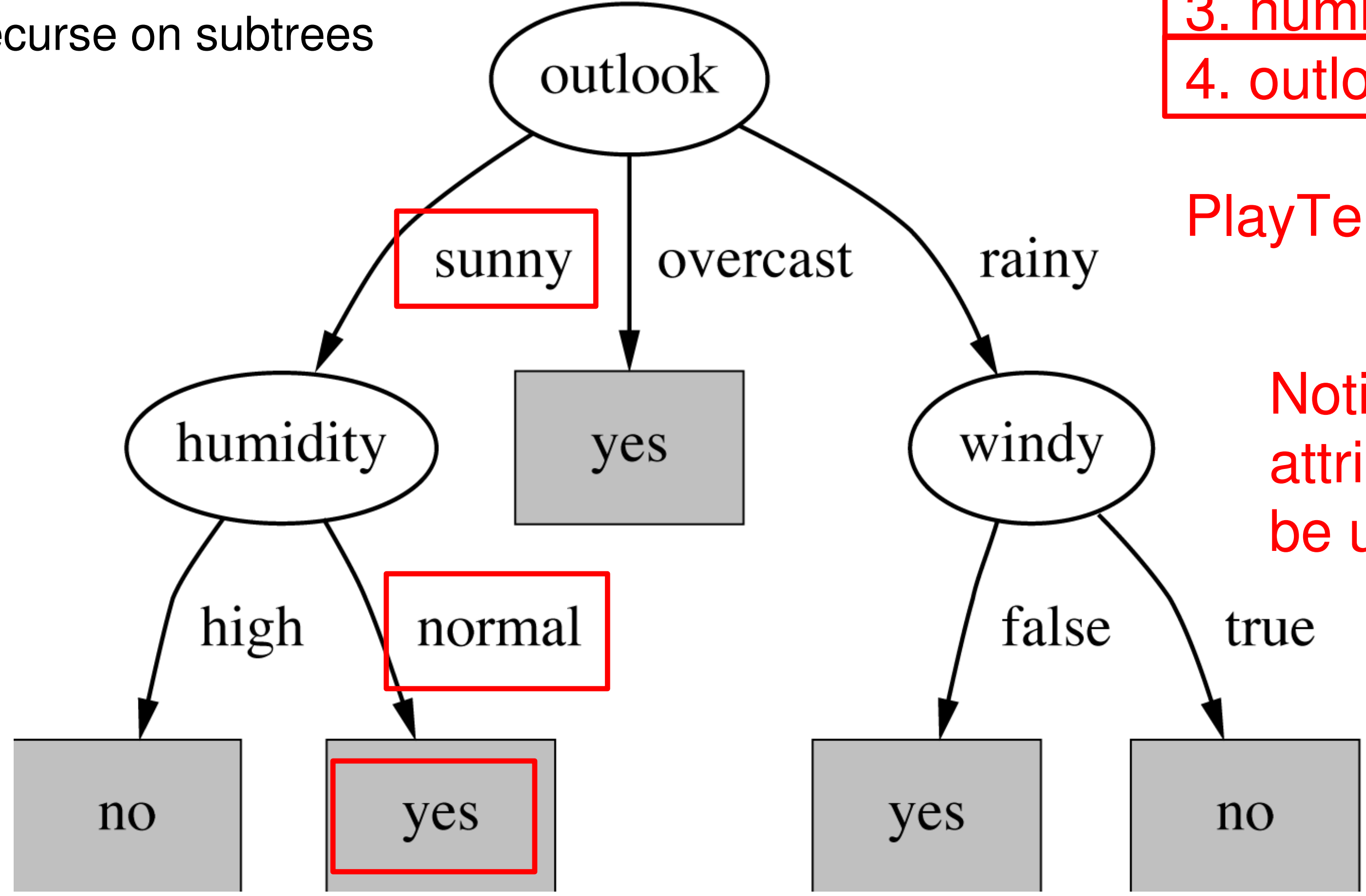
Decision Trees

Overfitting

Ensemble Learning

Decision Tree Learning (Quinlan, 86; Feigenbaum, 61)

Goal predicate: PlayTennis
- Nodes are attribute tests
- Branches are attribute values
- Recurse on subtrees



1. temperature = hot
2. windy = true
3. humidity = normal
4. outlook = sunny

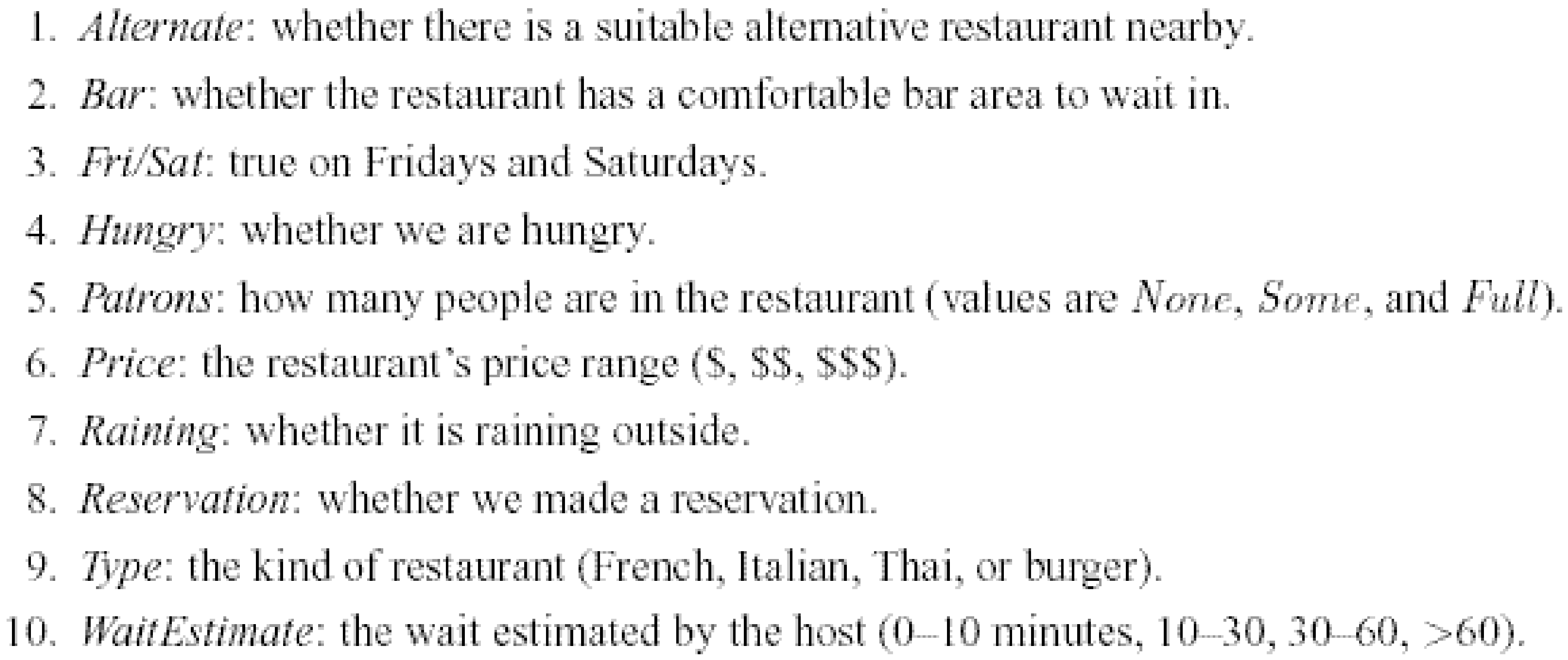
PlayTennis = ?

Notice not all attributes have to be used!

Illustrating Example (Russell/Norvig)

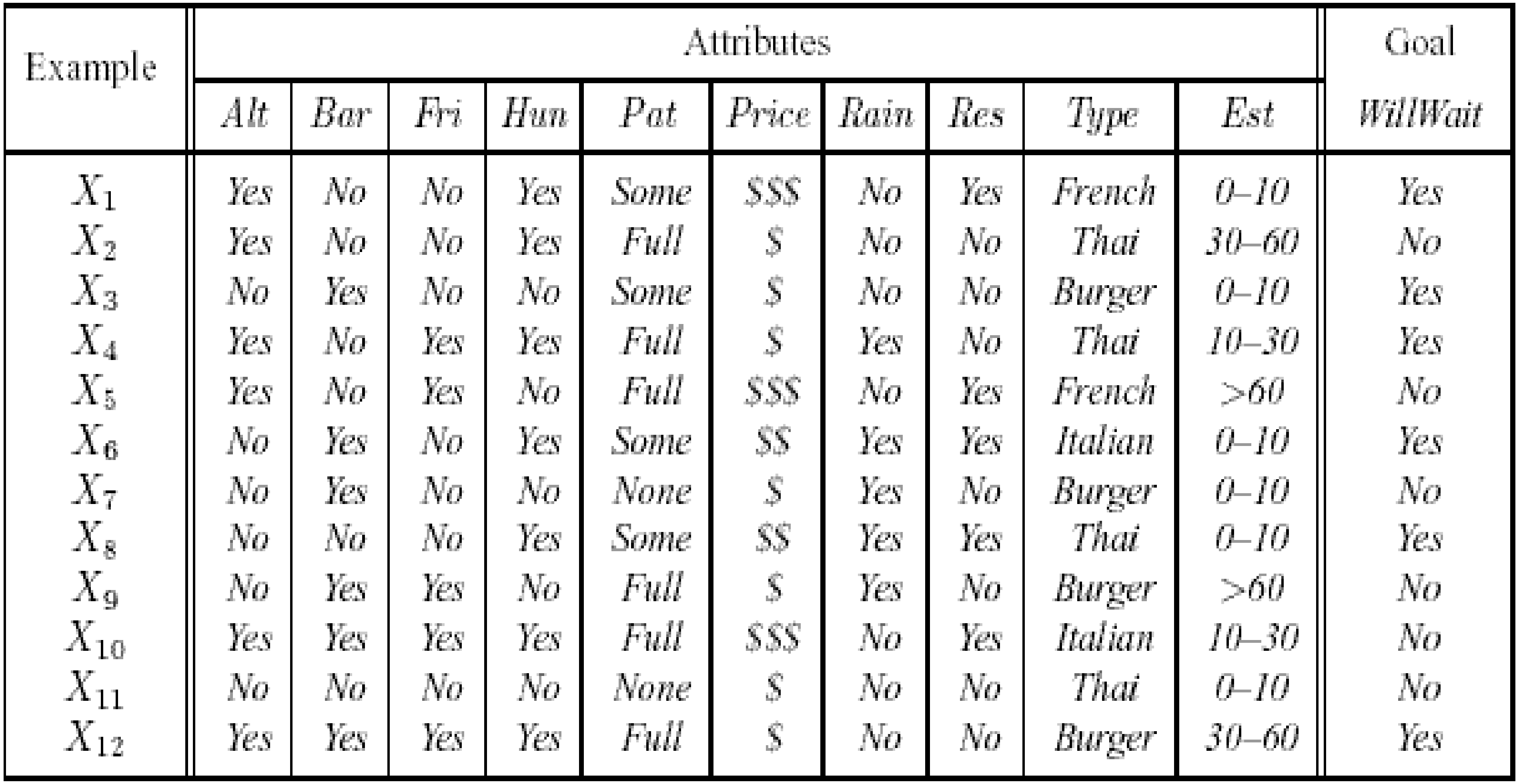
<u>The problem:</u> wait for a table in a restaurant?

Here are the attributes …



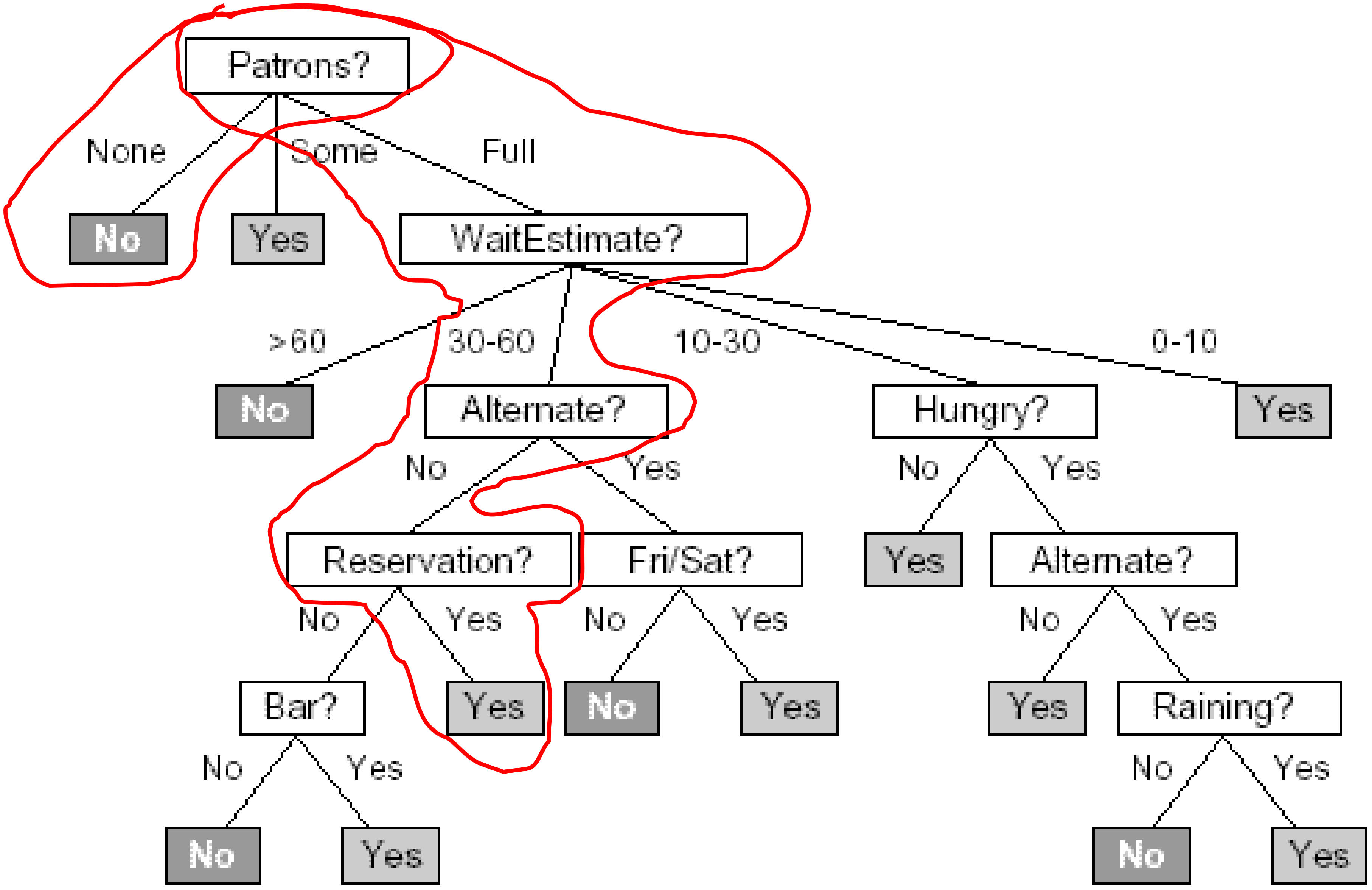
1. *Alternate*: whether there is a suitable alternative restaurant nearby.
2. *Bar*: whether the restaurant has a comfortable bar area to wait in.
3. *Fri/Sat*: true on Fridays and Saturdays.
4. *Hungry*: whether we are hungry.
5. *Patrons*: how many people are in the restaurant (values are *None*, *Some*, and *Full*).
6. *Price*: the restaurant's price range ($, $$, $$$).
7. *Raining*: whether it is raining outside.
8. *Reservation*: whether we made a reservation.
9. *Type*: the kind of restaurant (French, Italian, Thai, or burger).
10. *WaitEstimate*: the wait estimated by the host (0–10 minutes, 10–30, 30–60, >60).

Illustrating Example: Training Data



Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	*Yes*

A Decision Tree for *WillWait* (SR)



General Approach of Decision Trees

Let A_1, A_2, ..., and A_n be discrete attributes, i.e. each attribute has finitely many values

Let B be another discrete attribute, the goal attribute

Learning goal:

Learn a function $f: A_1 \times A_2 \times ... \times A_n \rightarrow B$

Examples:

Instances of $A_1 \times A_2 \times ... \times A_n \times B$

General Approach

Restricted hypothesis space bias:

The collection of all decision trees over the attributes A_1, A_2, ..., A_n, and B forms the set of possible candidate hypotheses

Preference bias:

Prefer small trees consistent with the training examples

Decision Trees: definition for record

A decision tree over the attributes A_1, A_2,.., A_n, and B is a tree in which

- each non-leaf node is labelled with one of the attributes A_1, A_2, ..., and A_n (e.g., "**Alternate?**")
- each leaf node is labelled with one of the possible values for the goal attribute B (e.g., WillWait = "**Yes**")
- a non-leaf node with the label A_i has as many outgoing arcs as there are possible values for the attribute A_i; each arc is labelled with one of the possible values for A_i (e.g., Alternate? = "**Yes**" or "**No**")

Decision Trees: application of tree for record

Let x be an instance from A_1 x A_2 x ... x A_n and let T be a decision tree.

- The instance x is processed by the tree T starting at the root and following the appropriate arc until a leaf node is reached.

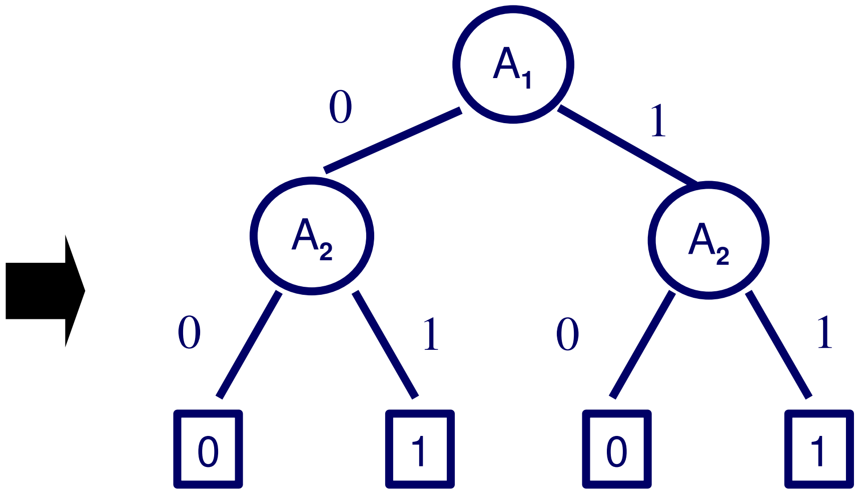
- x receives the value that is assigned to the leaf node.

Expressiveness of Decision Trees

Fully expressive within the class of propositional languages

Any boolean function can be written as a decision tree.

A_1	A_2	B
0	0	0
0	1	1
1	0	0
1	1	1



Appropriateness of Decision Trees

In some cases, decision trees are not appropriate …

Sometimes exponentially large decision trees
(e.g. parity function; returns 1 iff an even number of inputs are 1)

Replicated subtree problem
(e.g. when coding the following two rules in a tree:

„if A_1 and A_2 then B“
„if A_3 and A_4 then B“

Since trees imply the sharing of attributes for any solution path, a solution decision tree for the above requires subtree replication to represent.)

Decision Trees

Finding the smallest decision tree that is consistent with a set of examples presented is an NP-hard problem.

2^n bits to define a boolean function

2^{2^n} different functions on n attributes - what about 6 attributes?

$2^{2^6} = 18{,}446{,}744{,}073{,}709{,}551{,}616$

Instead of constructing a smallest decision tree the focus is on the construction of a “pretty small“ one

Inducing Decision Trees Algorithm

```
function DECISION-TREE-LEARNING(examples, attribs, default)
    returns a decision tree
    inputs:   examples, set of examples
              attribs, set of attributes
              default, default value for the goal predicate

    if examples is empty then return default
    else if all examples have the same classification
            then return the classification
    else if attribs is empty then return
                                       MAJORITY-VALUE(examples)
    else
            best   ← CHOOSE-ATTRIBUTE(attribs, examples)
            tree   ← a new decision tree with root test best
            m      ← MAJORITY-VALUE(examples_i)

    for each value v_i of best do
            examples_i ← {instances of examples with best = vi}
            subtree ← DECISION-TREE-LEARNING(examples_i,
                                       attribs - best, m)
            add a branch to tree with label v_i and subtree subtree
    return tree
```

Constructing decision trees

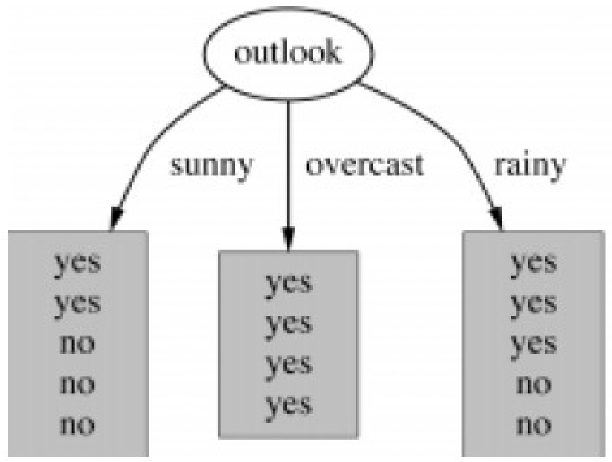
- Normal procedure: top down in recursive *divide-and-conquer* fashion
  - First: $\boxed{\text{attribute is selected}}$ for root node and branch is created for each possible attribute value
  - Then: the instances are split into subsets (one for each branch extending from the node)
  - Finally: procedure is repeated recursively for each branch, using only instances that reach the branch
- Process stops if all instances have the same class

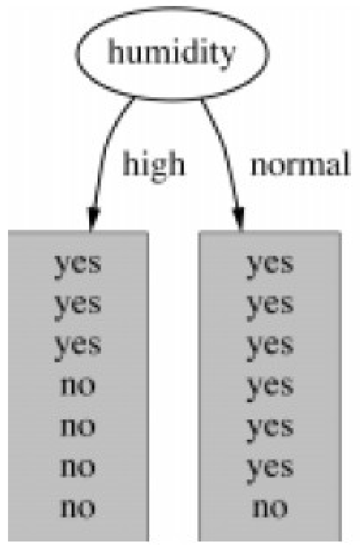
Training Examples

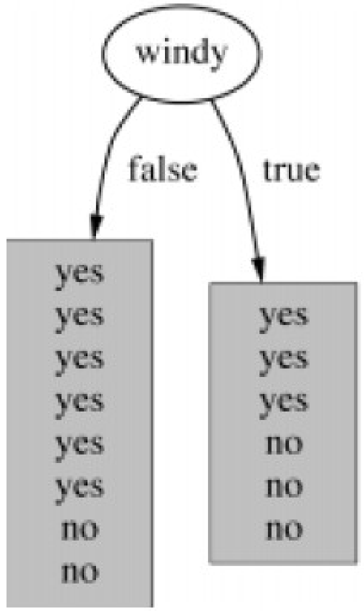
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

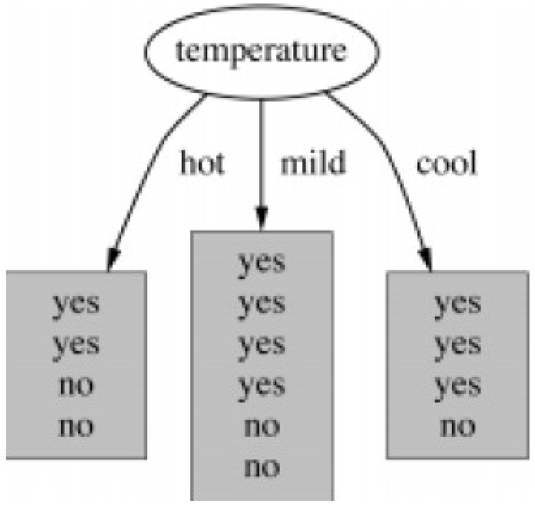
T. Mitchell, 1997

Which attribute to select?









Computing information

- Information is measured in *bits*
  - Given a probability distribution, the info required to predict an event is the distribution's *entropy*
  - Entropy gives the information required in bits (this can involve fractions of bits!)
- Formula for computing the entropy:

$$\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

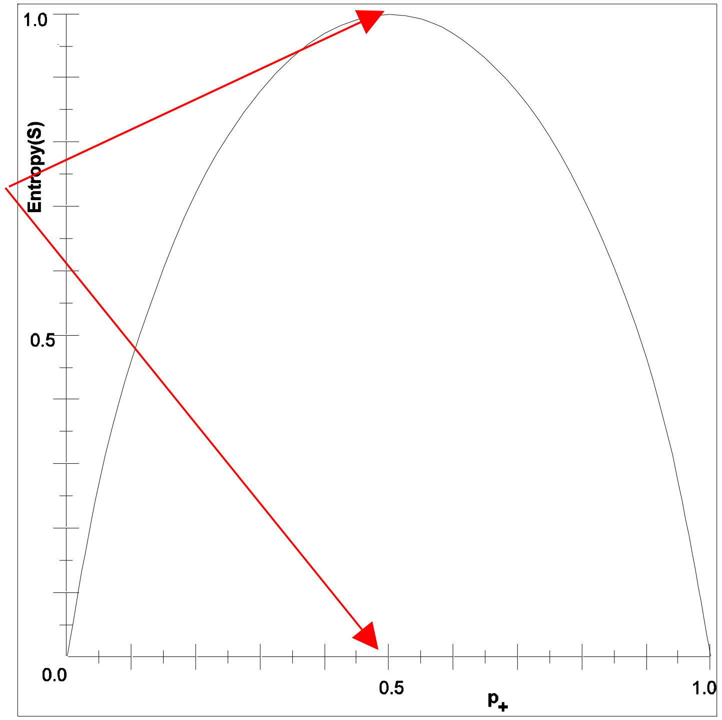
Entropy n = 2

Intuition 1:

Highest Entropy value is for the Attribute/Value pair with the most balanced distribution of positive and negative examples.

Intuition 2:

However, lower Entropy values make an attribute *more valuable*, as they typically lead to smaller tree.



- S is a sample of training examples
- p_+ is the proportion of positive examples in S
- p_- is the proportion of negative examples in S
- Entropy measures the impurity of S

$$\mathit{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Example: attribute "Outlook"

- "Outlook" = "Sunny":

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971 \text{ bits}$$

- "Outlook" = "Overcast":

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0 \text{ bits}$$

Note: this is normally not defined.

- "Outlook" = "Rainy":

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971 \text{ bits}$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$
$$= 0.693 \text{ bits}$$

Computing the information gain

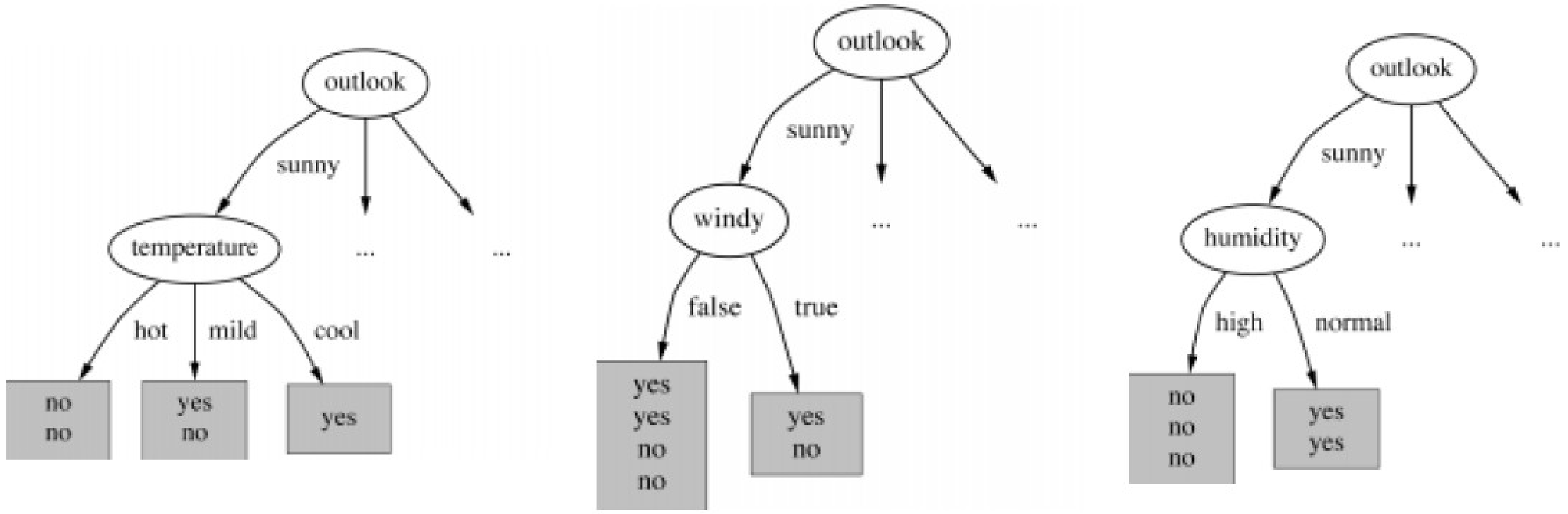
- Information gain: information before splitting – information after splitting

$$\text{gain}(\text{"Outlook"}) = \text{info}([9,5]) - \text{info}([2,3],[4,0,[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

- Information gain for attributes from weather data:

$$\text{gain}(\text{"Outlook"}) = 0.247 \text{ bits}$$
$$\text{gain}(\text{"Temperature"}) = 0.029 \text{ bits}$$
$$\text{gain}(\text{"Humidity"}) = 0.152 \text{ bits}$$
$$\text{gain}(\text{"Windy"}) = 0.048 \text{ bits}$$

Continuing to split

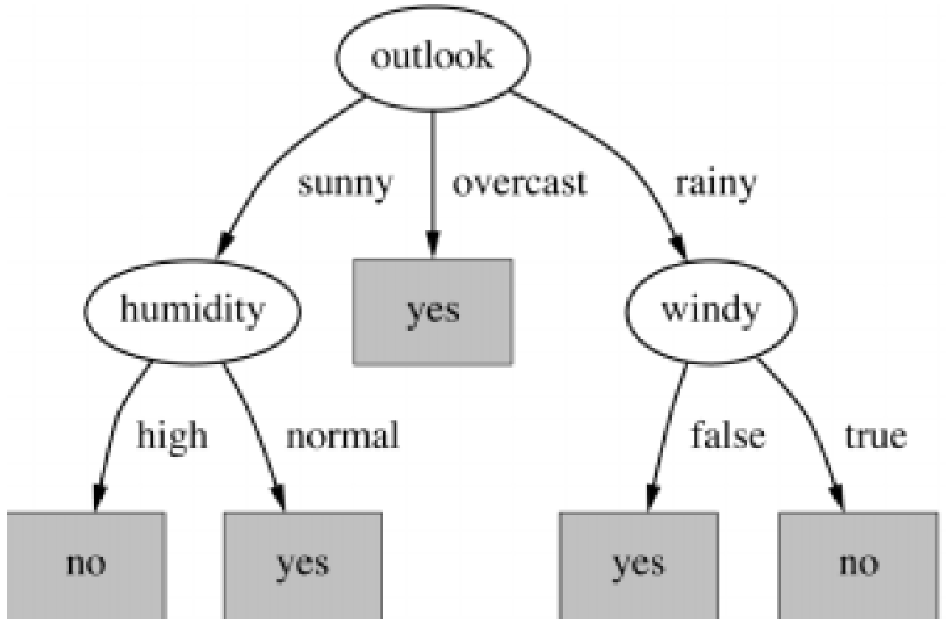


$$\text{gain}(\text{"Temperature"}) = 0.571\,\text{bits}$$

$$\text{gain}(\text{"Humidity"}) = 0.971\,\text{bits}$$

$$\text{gain}(\text{"Windy"}) = 0.020\,\text{bits}$$

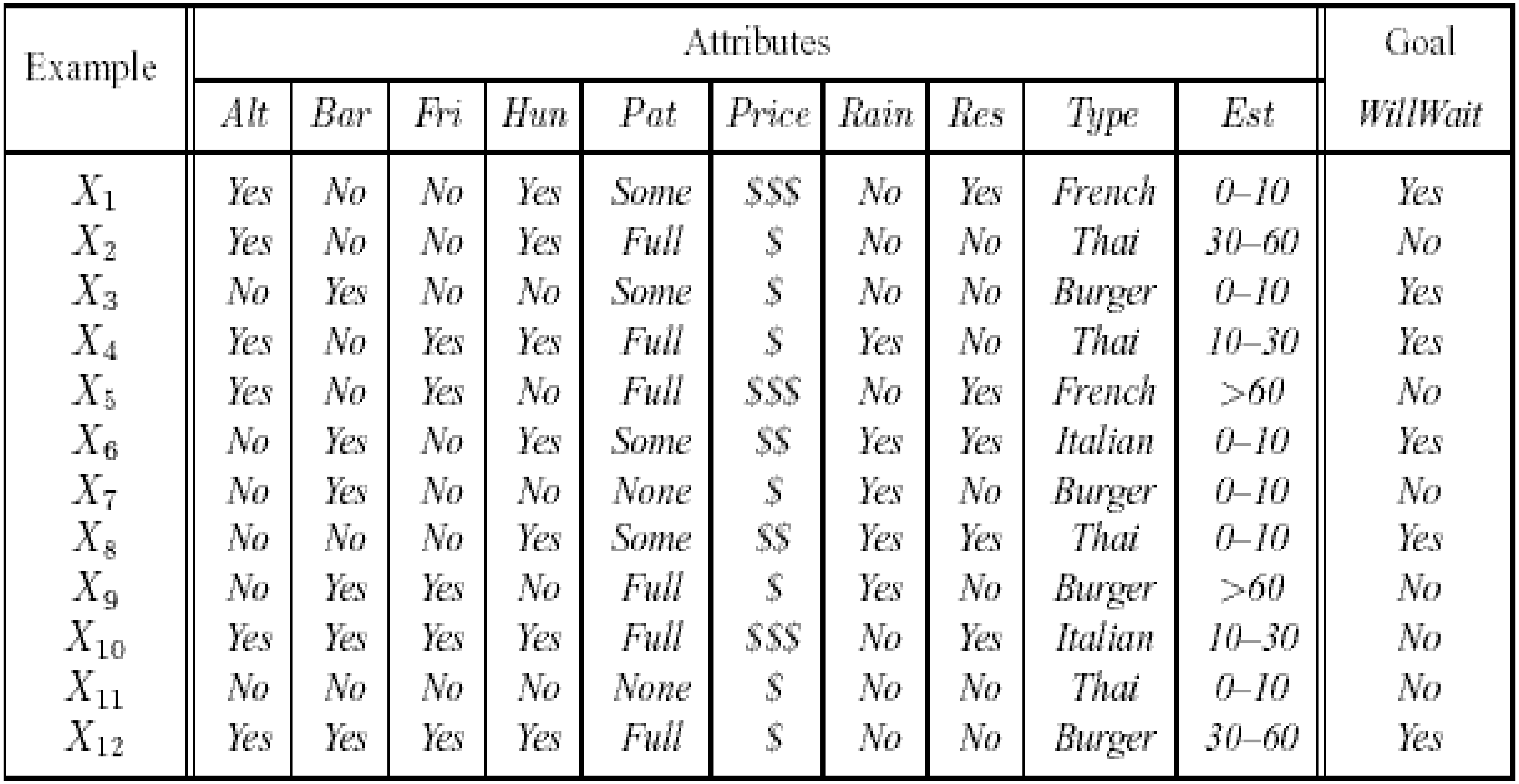
The final decision tree



- Note: not all leaves need to be pure; sometimes identical instances have different classes

  ⇒ Splitting stops when data can't be split any further

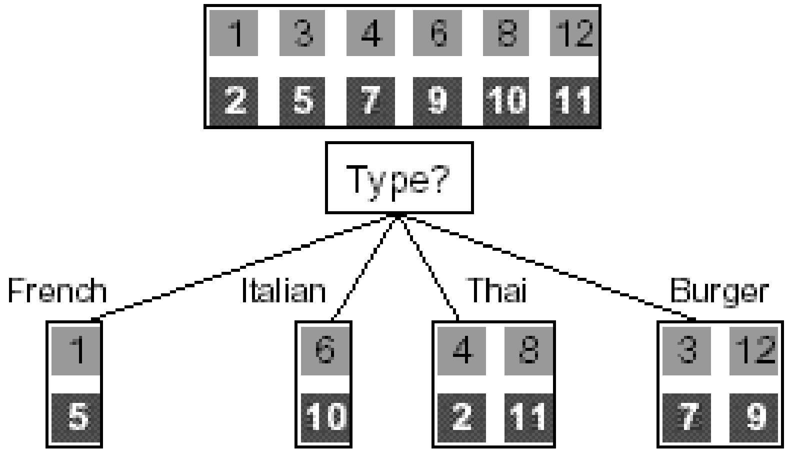
Return to Example *WillWait*

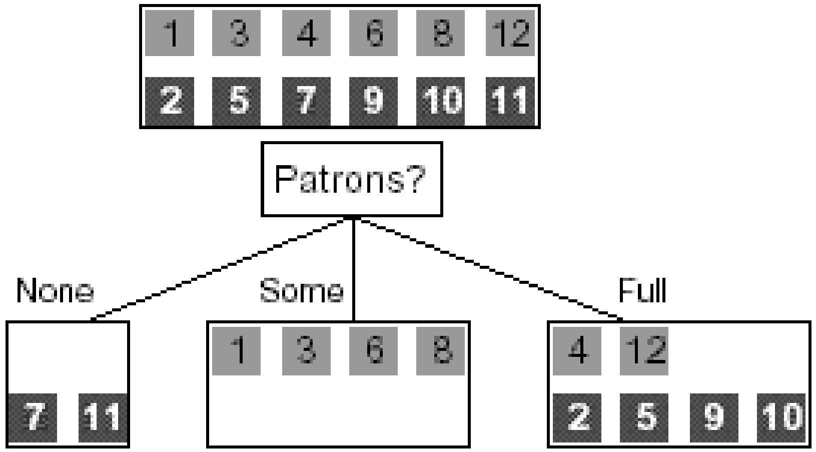


Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

the problem of whether to wait for a table in a restaurant

WillWait - Which Attribute to Choose?





Assessing Decision Trees

Assessing the performance of a learning algorithm:

a learning algorithm has done a good job if its final hypothesis predicts the value of the goal attribute of unseen examples correctly

General strategy (cross-validation)

1. collect a large set of examples
2. divide it into two disjoint sets: the training set and the test set
3. apply the learning algorithm to the training set, generating a hypothesis ***h***
4. measure the quality of ***h*** applied to the test set
5. repeat steps 1 to 4 for different sizes of training sets and different randomly selected training sets of each size

When is Decision Tree Learning Appropriate?

- Instances represented by attribute-value pairs
- Target function has discrete values (especially if boolean, e.g., WillWait? = {Yes, No}, PlayTennis = {Yes, No})
- Training data may contain missing or noisy data

Extensions and Problems

- Dealing with continuous attributes
  - select thresholds defining intervals; as a result each interval becomes a discrete value
  - dynamic programming methods to find appropriate split points still expensive
- Missing attributes
  - introduce a new value (e.g., „unknown“)
  - use default values (e.g. the majority value)
- Highly-branching attributes
  - e.g. *Date* has a different value for every example

Extensions and Problems

- **Noise**
  - e.g. two or more examples with the same description but different classifications -> Leaf nodes report the majority classification for its set or report estimated probability (relative frequency)
- **Overfitting**
  - the learning algorithm uses irrelevant attributes to find a hypothesis consistent with all examples;
  - pruning techniques; e.g. new non-leaf nodes will only be introduced if the information gain is larger than a particular threshold

Overview

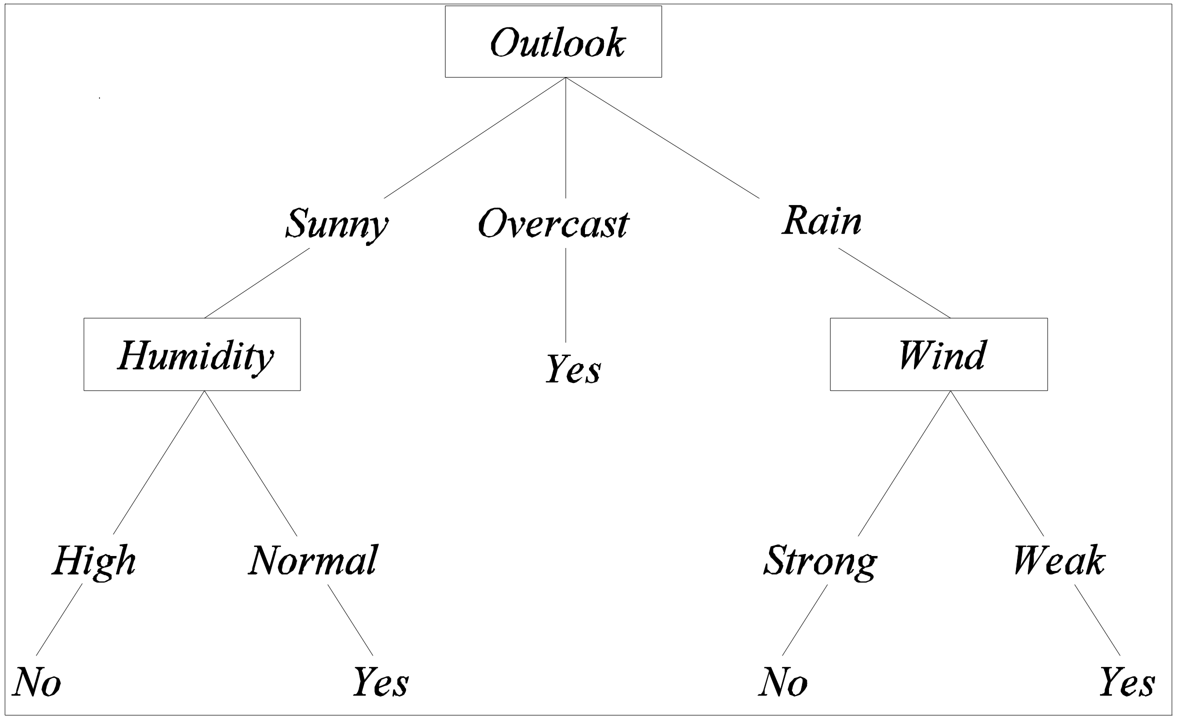
- Introduction to Machine Learning
- Supervised Inductive Learning
- Decision Trees
- Overfitting
- Ensemble Learning

Overfitting in Decision Trees

Consider adding training example #15:

Sunny, Hot, Normal, Strong, PlayTennis = No

What effect on earlier tree?



Overfitting

Consider error of hypothesis h over

training data: $error_{train}(h)$
entire data set: $error_D(h)$

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

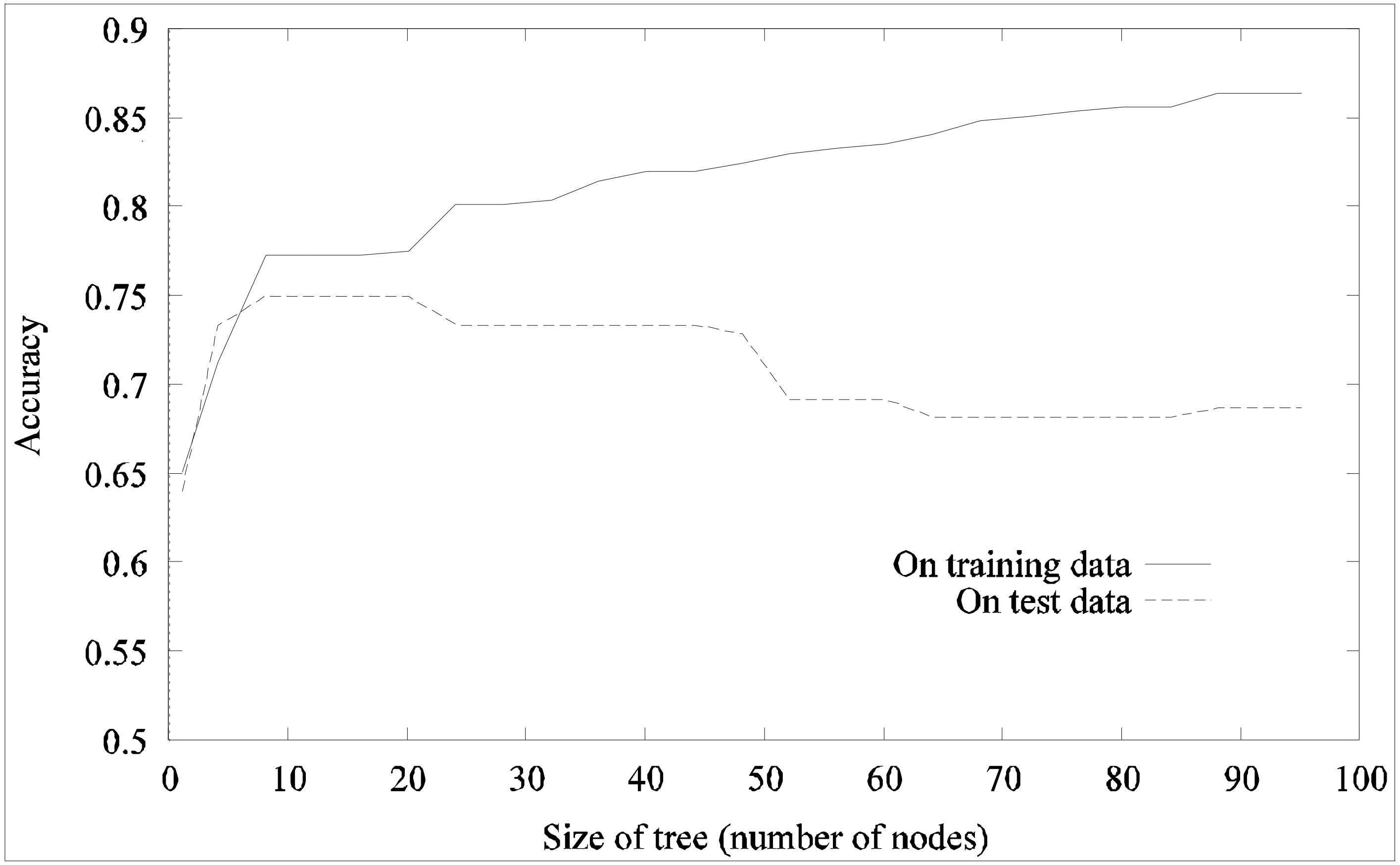
$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

Intuition: If induced hypothesis leads to lower error for given training examples, but not for data as a whole, we have overfitting.

Overfitting in Decision Tree Learning



T. Mitchell, 1997

Avoiding Overfitting

1. Stop growing when data split not statistically significant

2. Grow full tree, then post-prune

   How to select "best" tree:

   1. Measure performance over training data (threshold)
   2. Statistical significance test whether expanding or pruning at node will improve beyond training set χ^2
   3. Measure performance over separate validation data set (utility of post-pruning) general cross-validation

Reduced-Error Pruning

Split data into *training* 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

Produces smallest version of most accurate subtree

What if data is limited??

Overview

Introduction to Machine Learning

Supervised Inductive Learning

Decision Trees

Overfitting

Ensemble Learning

Ensemble Learning

- Run decision tree learning in parallel, perhaps with different parameter settings
- Collect resulting, individual hypotheses in an ***ensemble*** and combine their predictions appropriately

Motivation (1)

Let $h_1, h_2, \ldots, h_n$ be the set of individual hypotheses and let e be an example.

Typical voting schemes for ensembles:

- Unanimous vote:

  $h(e) = 1$ iff $\Sigma\, h_k(e) = n$

- Majority vote:

  $h(e) = 1$ iff $\Sigma\, h_k(e) > n/2$, i.e if the majority is positive

- Weighted majority vote:

  $h(e) = 1$ iff $\Sigma\, w_k h_k(e) > \Sigma\, w_k(1- h_k(e))$, where each hypothesis h_k has a weighting factor w_k

Motivation (2)

Advantage

To improve the quality of the overall hypothesis, for example:

- Collect the hypotheses of 5 learners in an ensemble
- Combine their hypotheses using a simple majority vote; to misclassify a new example, at least 3 out of 5 hypotheses have to misclassify it

Improvement under the assumptions

- each hypothesis h_k in the ensemble has an error of p; i.e. the probability that a randomly chosen example is misclassified by h_k is p
- the errors made by the individual hypotheses are independent

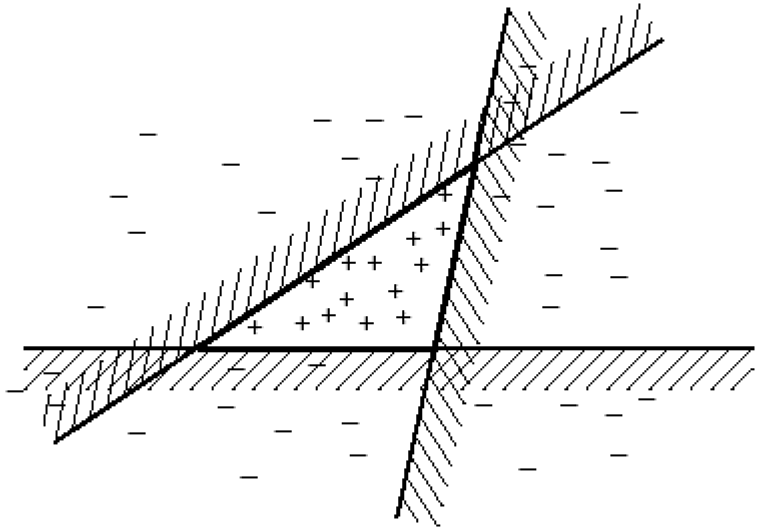
$$p_M = (\quad)\, p^3 (1-p)^2 + (\quad)\, p^4 (1-p) + (\quad)\, p^5$$

i.e. 1 in 100 instead of 1 in 10 !!

Motivation (3)

Advantage: enlarging the hypothesis space, expressive power

- an ensemble itself constitutes a hypothesis
- the new hypothesis space is the set of all possible ensembles constructible from hypotheses in the original hypothesis space of the individual learning algorithms
  - Example:



Three linear threshold hypotheses. Together, they create a hypothesis not expresible in the original space.

Boosting (1)

Improves the quality of the ensemble method

Boosts accuracy!

Basics:

- A weighted training set; i.e. each training example has an associated weight
- the method respects the weights of the training examples, i.e. the higher the weight of an example the higher its importance during the learning phase

Boosting (2)

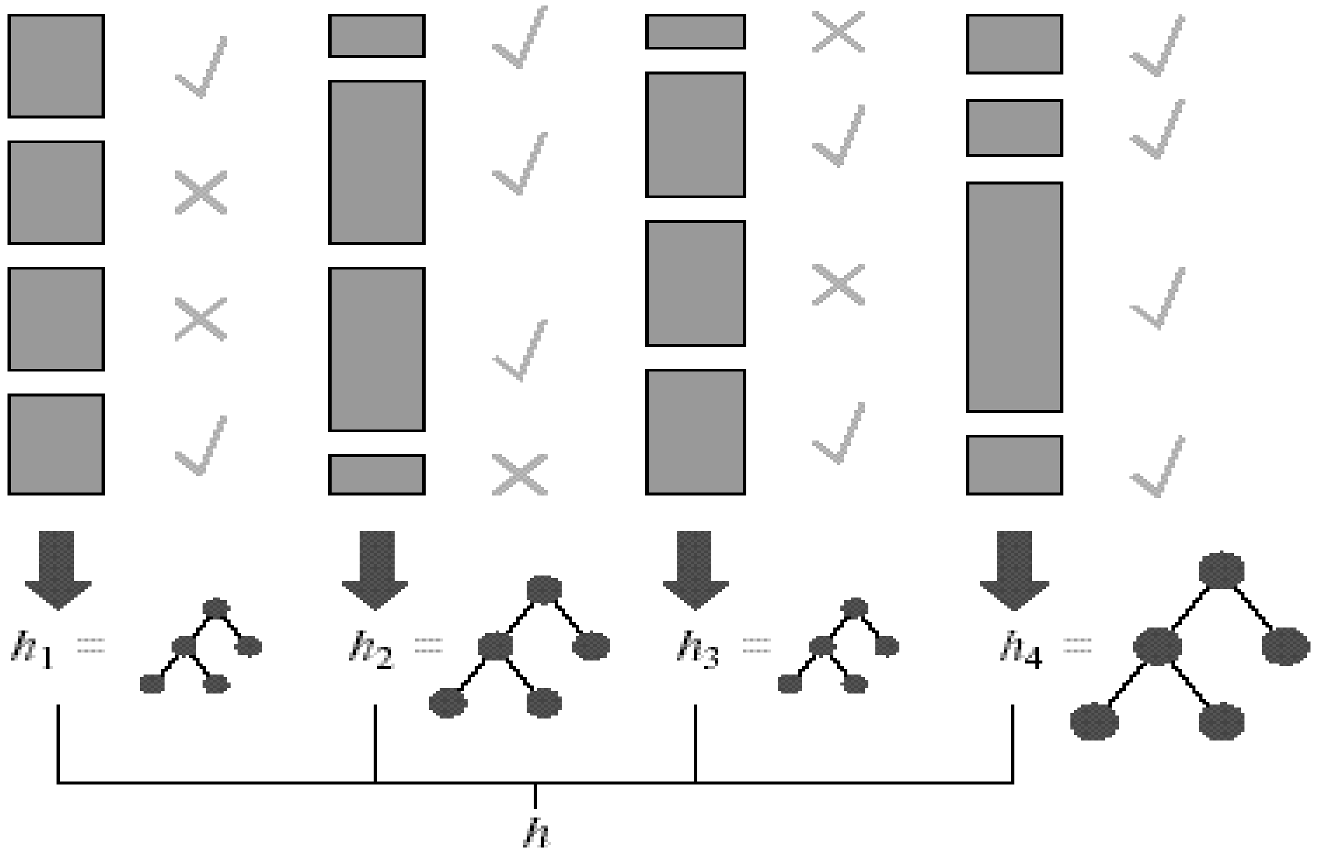
Initially, each training example has the fixed weight 1

The first round of learning – using learning method L – starts

n-th Round ($n \leq M$)

- run L on the given weighted training examples; let h_n be the hypothesis generated by L
- adopt the weight of the training examples as follows
  - decrease the weight, if the example is correctly classified by h_n
  - increase it, otherwise
- start the next round of learning

Boosting (3)



Intuition: As examples are given more weight, so are the hypotheses generated with these examples…

Boosting: Summary

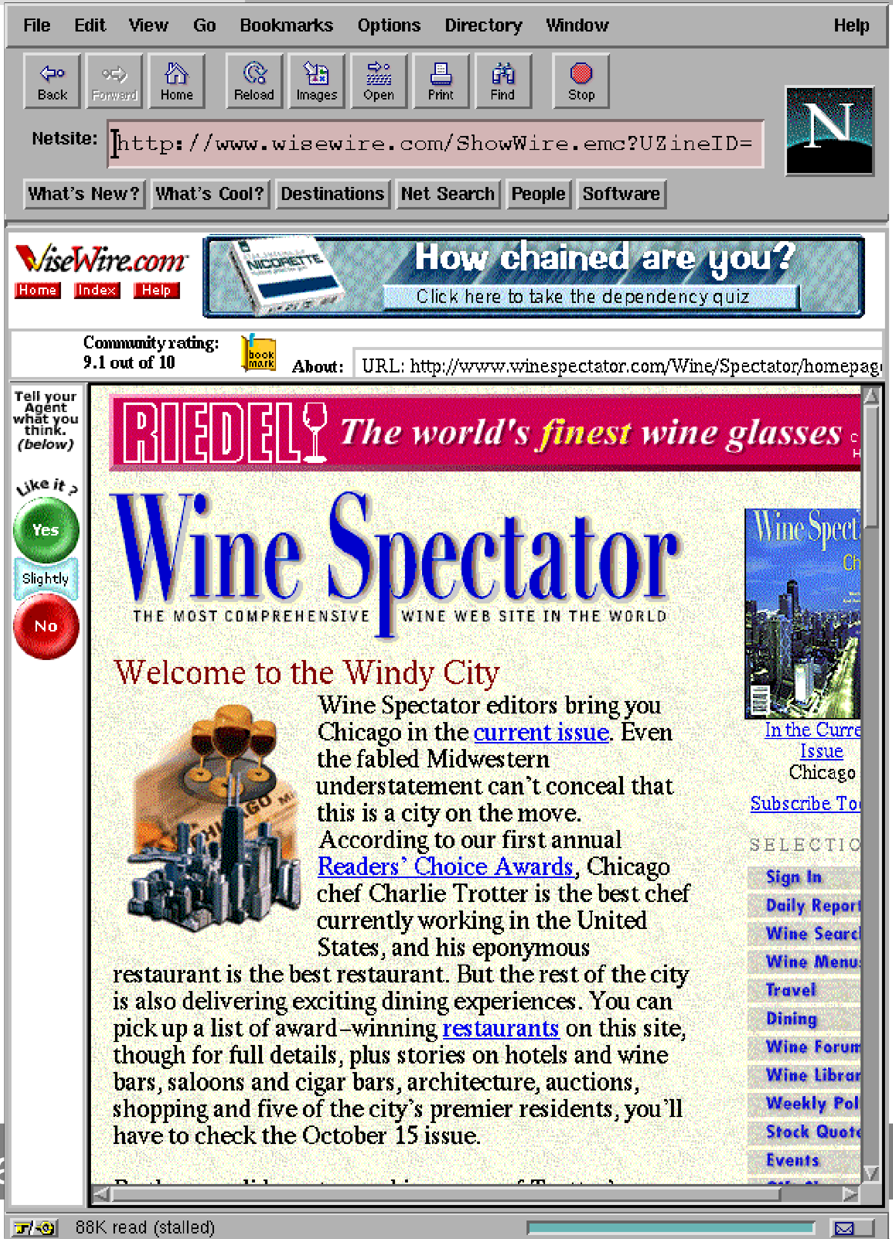
- There are many variants of boosting with different ways of adjusting the weights and combining the hypotheses

- Some have very interesting properties
  e.g. AdaBoost; even if the learning method L is *weak*
  AdaBoost will return a hypothesis that classifies the training data perfectly, provided M is large enough

weak?

L always returns a hypothesis with a weighted error on the training set that is slightly better than random guessing

Software that Customizes to User



File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Netsite:
What's New? What's Cool? Destinations Net Search People Software
WiseWire.com
Home Index Help
How chained are you?
Click here to take the dependency quiz
Tell your Agent what you think. (below)
Like it ?
Yes
Slightly
No
RIEDEL The world's finest wine glasses
Wine Spectator
THE MOST COMPREHENSIVE WINE WEB SITE IN THE WORLD
In the Current Issue
Chicago
Subscribe To
SELECTIO
Sign In
Daily Report
Wine Search
Wine Menus
Travel
Dining
Wine Forum
Wine Library
Weekly Poll
Stock Quote
Events
88K read (stalled)

http://www.wisewire.com/ShowWire.emc?UZineID=

Community rating: 9.1 out of 10

About: URL: http://www.winespectator.com/Wine/Spectator/homepag

Welcome to the Windy City

Wine Spectator editors bring you Chicago in the current issue. Even the fabled Midwestern understatement can't conceal that this is a city on the move. According to our first annual Readers' Choice Awards, Chicago chef Charlie Trotter is the best chef currently working in the United States, and his eponymous restaurant is the best restaurant. But the rest of the city is also delivering exciting dining experiences. You can pick up a list of award-winning restaurants on this site, though for full details, plus stories on hotels and wine bars, saloons and cigar bars, architecture, auctions, shopping and five of the city's premier residents, you'll have to check the October 15 issue.

Recommender systems (Amazon..)

End of Lecture: ARGUNAUT again …

- *Educational Data Mining* - the ARGUNAUT Project
  - Use of Machine Learning Techniques to learn collaborative / argumentative behaviors of students
  - Use the resulting classifiers to support the moderator of an online collaborative discussion
- Looking for an enthusiastic student / HIWI to work on this project
  - Learn to use modern Machine Learning tools
  - Integrating data and text mining
  - Work with a team of learning scientists and PhD students
  - Academic publications!
- Talk to me after the lecture or send email to **bmclaren@dfki.de**