Decision Tree Learning

Decision Trees

- Examples of systems that learn decision trees: c4.5, CLS, IDR, ASSISTANT, ID5, CART, ID3.
- Suitable problems:
  - Instances are described by attribute-value couples
  - The target function has discrete values
  - Disjunctive descriptions of concepts may be required
  - The training set may contain errors (noise)
  - The training set may contain incomplete data

c4.5

- c4.5 [Qui93b,Qui96]: evolution of ID, also by J. R. Quinlan
- Inspired to one of the first decision tree learning system, CLS (Concept Learning Systems) by E.B. Hunt
- Benchmark for many learning systems

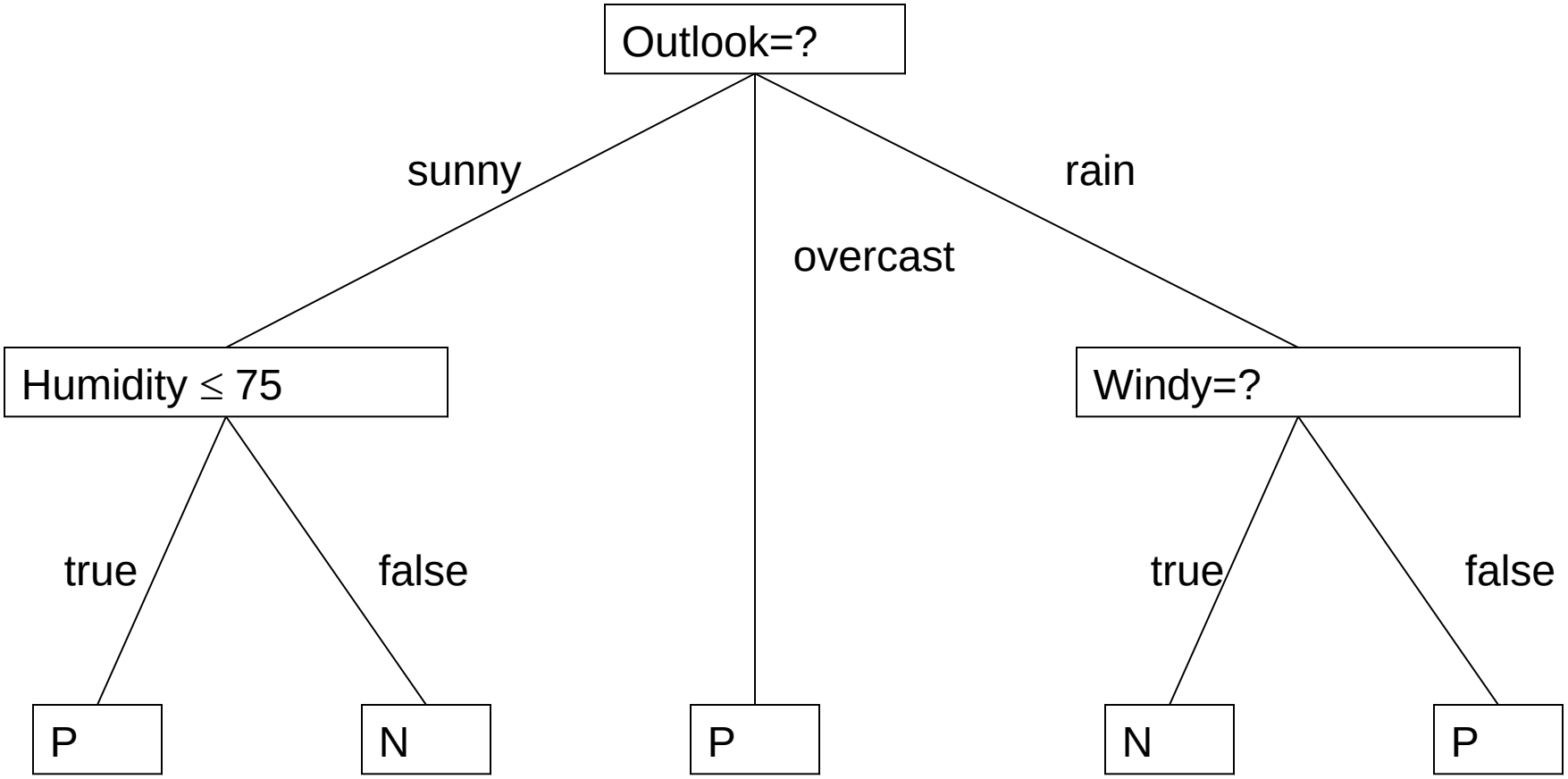
Example

- Instances: Saturday mornings
- Classes:
  - Good day for playing tennis
  - Bad day for playing tennis
- Attributes
  - outlook, discrete, values={sunny,overcast,rain}
  - temperature, continuous
  - humidity, continuous
  - windy, discrete, values={true, false}

Training set

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D1	sunny	75	70	T	P
D2	sunny	80	90	T	N
D3	sunny	85	85	F	N
D4	sunny	72	95	F	N
D5	sunny	69	70	F	P
D6	overcast	72	90	T	P
D7	overcast	83	78	F	P
D8	overcast	64	65	T	P
D9	overcast	81	75	F	P
D10	rain	71	80	T	N
D11	rain	65	70	T	N
D12	rain	75	80	F	P
D13	rain	68	80	F	P
D14	rain	70	96	F	P

Decision Tree



Decision Tree

Outlook=sunny

| Humidity $\leq$ 75: P

| Humidity > 75: N

Outlook=overcast: P

Outlook=rain

| Windy=True: N

| Windy=False: P

Notation

- Let T be the training set,
- Let $\{c_1, c_2, \ldots, c_k\}$ be the set of classes;

Tree Building Algorithm

- build_tree(T) returns a tree:
  - T contains examples from the same class
    - Return a leaf with label the class
  - T contains examples from more than one class
    - T is partitioned into subsets $T_1, T_2, \ldots, T_n$ according to a test on an attribute
    - Call the algorithm recursively on the subsets:
      - $child_i$=build_tree(T_i) for i=1,…n
    - Return a subtree with the root associated to the test and children $child_1, \ldots, child_n$.

Example of build_tree

T=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D1	sunny	75	70	T	P
D2	sunny	80	90	T	N
D3	sunny	85	85	F	N
D4	sunny	72	95	F	N
D5	sunny	69	70	F	P
D6	overcast	72	90	T	P
D7	overcast	83	78	F	P
D8	overcast	64	65	T	P
D9	overcast	81	75	F	P
D10	rain	71	80	T	N
D11	rain	65	70	T	N
D12	rain	75	80	F	P
D13	rain	68	80	F	P
D14	rain	70	96	F	P

Test on Outlook

- T_{sunny}=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D1	sunny	75	70	T	P
D2	sunny	80	90	T	N
D3	sunny	85	85	F	N
D4	sunny	72	95	F	N
D5	sunny	69	70	F	P

- $T_{overcast}$=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D7	overcast	83	78	F	P
D8	overcast	64	65	T	P
D9	overcast	81	75	F	P

Test on Outlook

- T_{rain}=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D10	rain	71	80	T	N
D11	rain	65	70	T	N
D12	rain	75	80	F	P
D13	rain	68	80	F	P
D14	rain	70	96	F	P

build_tree(T_{sunny})

- Test: Humidity $\leq$ 75
- $T_{sunny,\ Humidity \leq 75}$

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D1	sunny	75	70	T	P
D5	sunny	69	70	F	P

- Leaf, P label
- $T_{sunny,\ Humidity > 75}$

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D2	sunny	80	90	T	N
D3	sunny	85	85	F	N
D4	sunny	72	95	F	N

- Leaf, N label

build_tree($T_{overcast}$)

- $T_{overcast}$=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D7	overcast	83	78	F	P
D8	overcast	64	65	T	P
D9	overcast	81	75	F	P

- Leaf, P label

build_tree(T_{rain})

- Test: Windy=?
- $T_{rain,true}$=

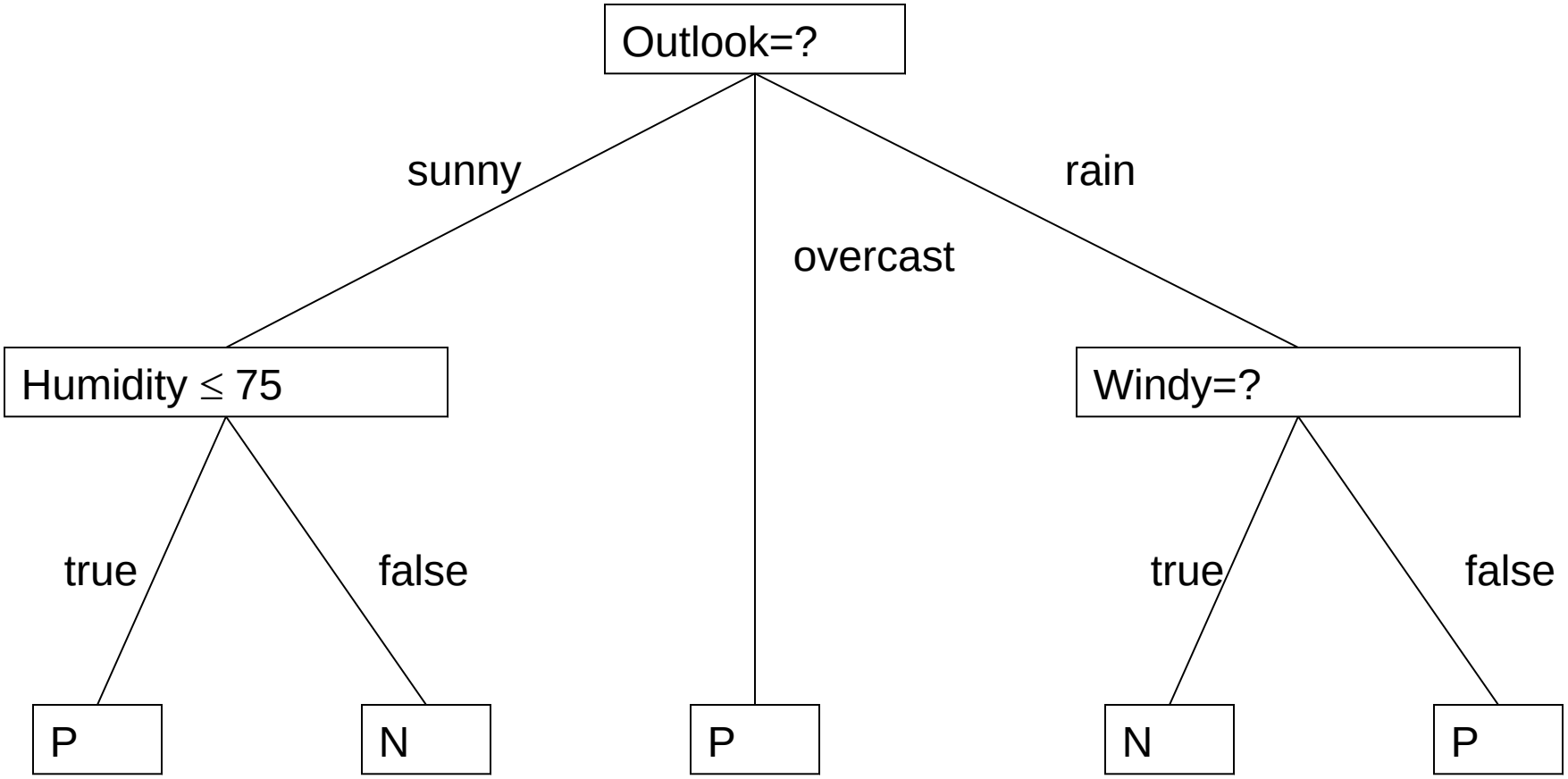
No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D10	rain	71	80	T	N
D11	rain	65	70	T	N

- Leaf, N label
- $T_{rain,false}$=

No	Outlook	Temp (°F)	Humid (%)	Windy	Class
D12	rain	75	80	F	P
D13	rain	68	80	F	P
D14	rain	70	96	F	P

- Leaf, P label

Decision Tree

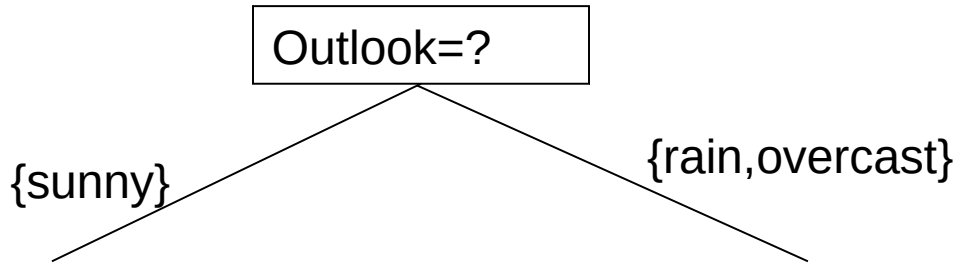


Tests on Attributes

- Discrete attribute X with n possible values $x_1,\ldots,x_n$:
  - Equality with a constant: X=cost, 2 possible outcomes: yes, no
  - Equality test: X=?, n possible outcomes
  - Membership in a set: $X \in S$, 2 possible outcomes: yes, no
  - Membership in a set of a partition of $\{x_1,\ldots,x_n\}$: one outcome per set

Tests on Attributes

- Example of membership in a set of a partition:
  - Attribute Outlook, partition of the set of values {{sunny},{rain,overcast}}



- Continuous attribute X
  - Comparison with a threshold X≤cost, 2 possible outcomes: yes, no

Termination Condition

- c4.5 stops
  - When an uniform set is found
  - When an empty set is found
    - A leaf is returned with label the most frequent class in the father
  - When no test is such that at least two subsets contain a minimum number of cases.
    - The minimum number of cases is a user-defined parameter (e.g. 2)

Building the Tree

- Search in space of all possible trees
  - Once a test is assigned to a node it is possible to backtrack
  - Infeasible
- Greedy search
  - Tests on nodes chosen irrevocably: once a test is assigned to a node it is not possible to backtrack
  - Choice on the basis of a heuristic
  - Most used heuristics
    - Entropy
    - Gini index

Choice of the Test

- Choice of the attribute
- Discrete attributes:
  - Choice of the type of test
  - Possibly choice of the constant or partition
- Continuous attributes
  - Choice of the threshold
- Usually only the equality test X=? is used for discrete attributes
  - Only the attribute must be chosen
- Constraints that the test must satisfy: at least two among $T_1, T_2, \ldots, T_n$ must contain a minimum number of examples

Entropy

- In information theory, entropy is a measure of the uncertainty associated with a discrete random variable.
- Random variable C with k possible values $c_1,\ldots,c_k$, entropy H(C) is given by

$$H(C) = -\sum_{j=1}^{k} P(c_j)\log_2 P(c_j) = E[-\log_2 P(C)]$$

- Also known as Shannon entropy

Information Theory Interpretation

- Suppose you want to transmit a series of messages made of the values of N independent and identically distributed (i.i.d.) random variables
- What is the minimum number of bits necessary to transmit these messages?
- Source coding theorem (Shannon 1948)
  - "The minimum number of bits necessary to encode the values of N i.i.d. random variables with negligible risk of information loss is $N \times H(X)$ as N tends to infinity, where H(X) is the entropy of the random variables"
  - H(X) is the minimum number of bits to encode the value of a random variable X

Application to Decision Trees

- Random variable C: class of an instance randomly selected from a set T
- In this case

$$P(c_j) = \frac{|T_j|}{|T|}$$

- where T_j is the set of examples from T that belong to class c_j
- We can define the entropy of T as the entropy of the random variable C

H(T)=H(C)

Entropy

- H(T) measures the minimum number of bits necessary for encoding, without loss, a message of the form
  - "The present example, randomly selected from the training set, belongs to class c_j"
- H(T) is also called info(T)

Entropy for Two Classes

- In the case of two classes, + and -, with probabilities p_+=P(+) and p_-=P(-)

$$H(T) = -p_+ \times \log_2 p_+ - p_- \times \log_2 p_-$$

- Only one variable is independent: p_-=1-p_+

$$H(T) = -p_+ \times \log_2 p_+ - (1 - p_+) \times \log_2 (1 - p_+)$$

Entropy for Two Classes

- For p_+=0.5: p_-=0.5

$$H(T) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 =$$
$$-0.5 \times (-1) - 0.5 \times (-1) = 1$$

- For p_+=0: p_-=1.0

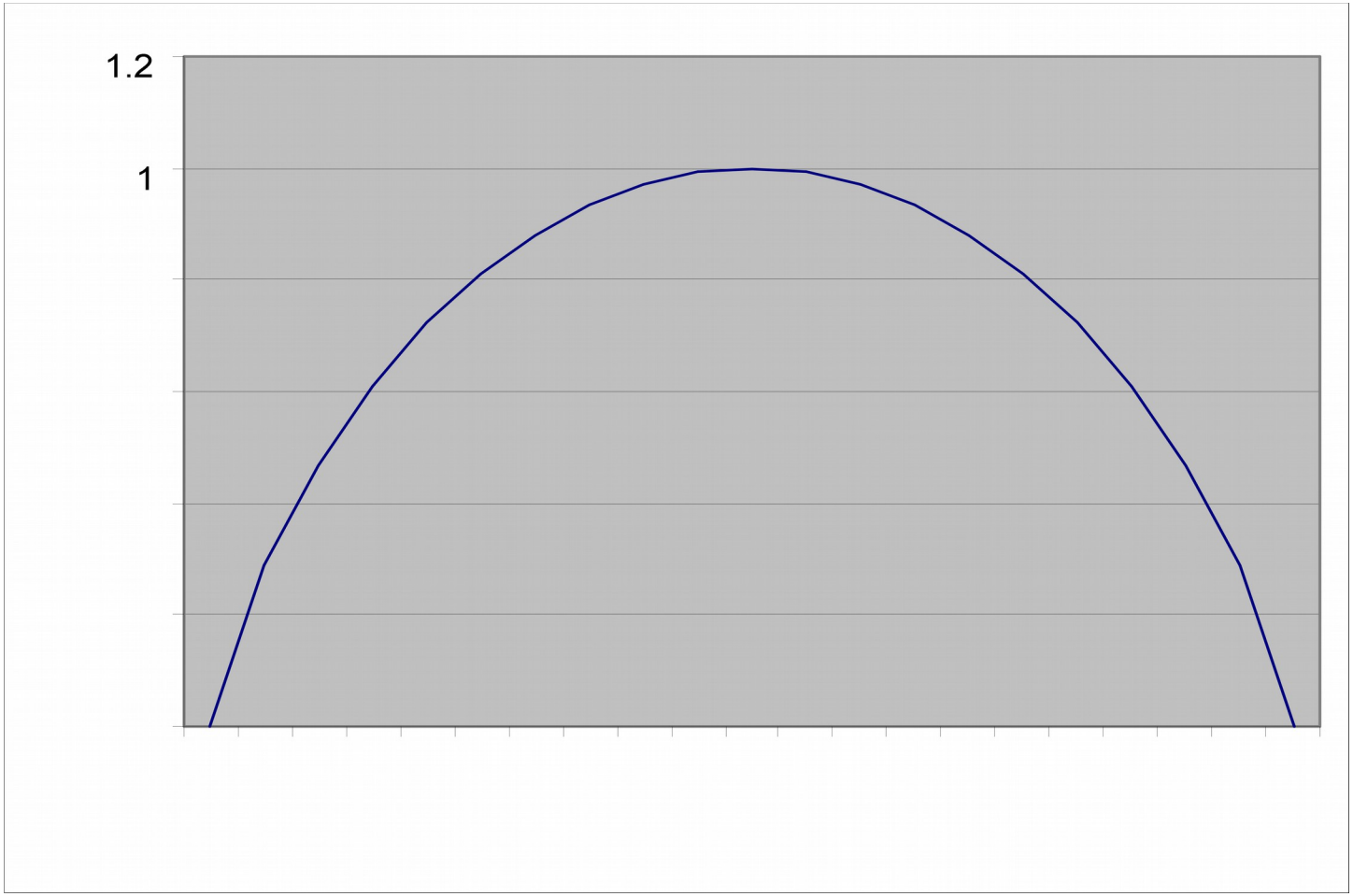
$$H(T) = -0 \times \log_2 0 - 1 \times \log_2 1 = (-0 \times -\infty) - 0$$

- In this case, we define

$$p_+ \times \log_2 p_+ \,|_{p_+=0} \overset{def}{=} \lim_{p_+ \to 0} p_+ \times \log_2 p_+ = 0$$

- So H(T)=0-0=0
- Similarly, if p_+=1 H(T)=0

Entropy for Two Classes



Entropy

- Entropy measures also the non-uniformity or impurity of a set:
  - It is minimal when the set is most pure, i.e., when it contains examples from only one class
  - It is maximal when the set is most impure, i.e., when it contains an equal number of examples of the two classes

Intuition for Two Classes

- If all the examples belongs to the same class, I do not need to communicate the class of a randomly drawn example
- If the examples are uniformly distributed over classes in the training set, I need to use 1 bit on average: message 0 encodes one class, message 1 the other

Example

- Play tennis problem
- Training set T: 14 examples, 9 positive, 5 negative
- Entropy of T
- Remember: $\log_a x=\log_{10} x/\log_{10} a=\ln x/\ln a$
- $\log_{10} 2=0.301$

$$H(T)=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)=$$

$$=-(9/14)\log_{10}(9/14)/0.301-(5/14)\log_{10}(5/14)/0.301=$$

$$=-0.643*(-0.192)/0.301-0.357*(-0.447)/0.301=$$

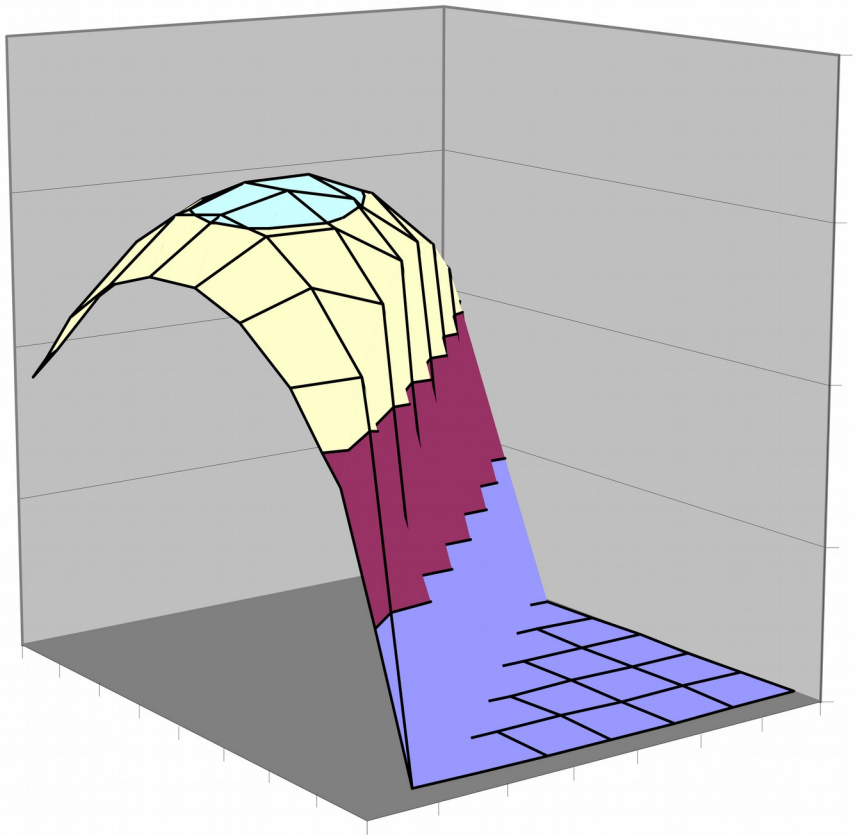
$$=0.410+0.530=0.940$$

Entropy for Three Classes

- Three classes $\{c_1, c_2, c_3\}$:
  - $p_1 = P(c_1)$
  - $p_2 = P(c_2)$
  - $p_3 = P(c_3)$
  - Only two independent variables: $p_3 = 1 - p_1 - p_2$

$$H(T) = -p_1 \times \log_2 p_1 - p_2 \times \log_2 p_2 -$$
$$-(1 - p_1 - p_2) \times \log_2 (1 - p_1 - p_2)$$

Entropy for Three Classes



Maximum of the Entropy

- The maximum of the entropy is obtained for a uniform distribution of the examples over classes
- For three classes: $p_1=p_2=p_2=1/3=0,333$
- $H(T)=-3*1/3*\log_2 1/3=\log_2 3=1,585$
- For k classes, we get the maximum for $P(c_j)=1/k$

$$H_{\max}(T) = -\sum_{j=1}^{k} \frac{1}{k} \times \log_2\left(\frac{1}{k}\right) = k \times \frac{1}{k} \times \log_2 k = \log_2 k$$

c4.5 Heuristic

- **Information gain of a test:** the value given by the decrease of entropy due to the partitioning of the set of examples according to the test
- Test X with n possible outcomes
- Set of examples T is partitioned into n subsets T_1, …,T_n
- Entropy after the partitioning: weighted average entropy in the subsets

$$H_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times H(T_i)$$

Information Gain

$$gain(X) = H(T) - H_X(T)$$

- It is equivalent to the number of bits necessary for encoding the class of examples that we save when we also know the result of the test X on the examples
- Entropy decreases as more and more uniform subsets are obtained

Attribute Choice

- The attribute with the highest information gain is selected for splitting the current set of examples

Example

- Test on the attribute Windy:
- Windy=F => T_F contains 6 positive and 2 negative examples
- Windy=T => T_T contains 3 positive and 3 negative examples
- T=[9,5]
- T_F=[6,2]
- T_T=[3,3]

Example

Windy	Play	Don't Play	Total
F	6	2	8
T	3	3	6
Total	9	5	14

- $H(T_F)=-6/8*\log_2(6/8)-2/8*\log_2(2/8)=0.811$
- $H(T_T)=-3/6*\log_2(3/6)-3/6*\log_2(3/6)=1$

$gain(Windy)=H(T)-(8/14*H(T_F)+6/14*H(T_T))=$

$0.940-(8/14)*0.811-(6/14)*1=0.048$

Example

- Test on the attribute outlook:
- Outlook=sunny => T_{sunny} contains 2 pos and 3 neg
- Outlook=overcast => $T_{overcast}$ contains 4 pos and 0 neg
- Outlook=rain => T_{rain} contains 3 pos and 2 neg
- T=[9,5], T_{rain}=[3,2], T_{sunny}=[2,3], $T_{overcast}$=[4,0]

Example

Outlook	Play	Don't Play	Total
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5
Total	9	5	14

$H(T_{sunny})=-2/5*\log_2(2/5)-3/5*\log_2(3/5)=0.971$

$H(T_{overcast})=-4/4*\log_2(4/4)-0/4*\log_2(0/4)=0$

$H(T_{rain})=-3/5*\log_2(3/5)-2/5*\log_2(2/5)=0.971$

$gain(Outlook)=H(T)-((5/14)*H(T_{sunny})+(4/14)*H(T_{overcast})+(5/14)*H(T_{rain}))=$

$0.940-0.357*0.971-0.286*0-0.357*0.971=0.246$

Problems of Information Gain

- High tendency to favor tests with many results
- Example: test on an attribute that is a key: $|T_i|=1$, $n = |T|$

$$H(T_i) = 0 \forall i \in [1,2,\ldots,n] \Rightarrow H_X(T) = 0$$

- So the gain is maximum:

$$gain(X) = H(T)$$

- Example: diagnosis problem, examples=patients, attribute="Patient's name"
  - Useless attribute but has the highest gain

Normalized Gain

- The gain is normalized with respect to the entropy of the test itself
- The random variable in this case is the value of the attribute X

$$H(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right)$$

- Gain ratio

$$gainratio(X) = \frac{gain(X)}{H(X)}$$

Gain Ratio in the Diagnosis example

- X=patient name

$$gain(X) = H(T) \le \log_2 k$$

$$H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = n \frac{1}{n} \log_2 n = \log_2 n$$

$$gainratio(X) \le \frac{\log_2 k}{\log_2 n}$$

- If n>>k$\Rightarrow$gain ratio(X) is small
- H(X) is also called splitinfo

Example

gainratio(Windy)=gain(Windy)/H(Windy)

H(Windy)=-8/14*$\log_2$(8/14)-6/14*$\log_2$(6/14)=0.985

gainratio(Windy)=0.048/0.985=0.048

gainratio(Outlook)=gain(Outlook)/H(Outlook)

H(Outlook)=-5/14*$\log_2$(5/14)-5/14*$\log_2$(5/14)
-4/14*$\log_2$(4/14)=1.577

gainratio(Outlook)=0.246/1.577=0.156

Gini Index

- Other impurity measure

$$gini(T) = 1 - \sum_{j=1}^{k} P(C_j)^2$$

- For two classes

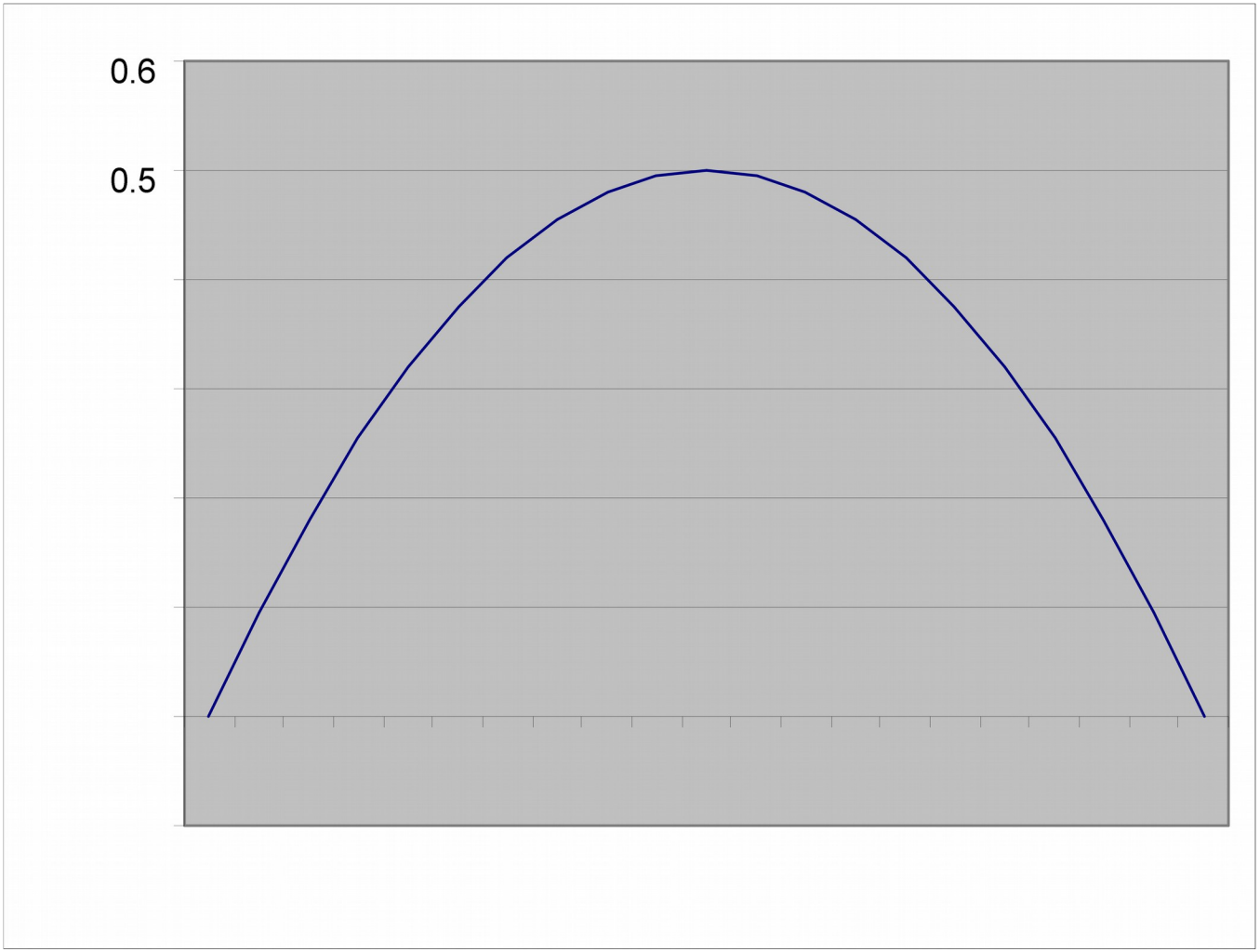
$$gini(T) = 1 - p_+^{\ 2} - (1 - p_+)^2 = 2p_+ - 2p_+^2$$

Gini Index

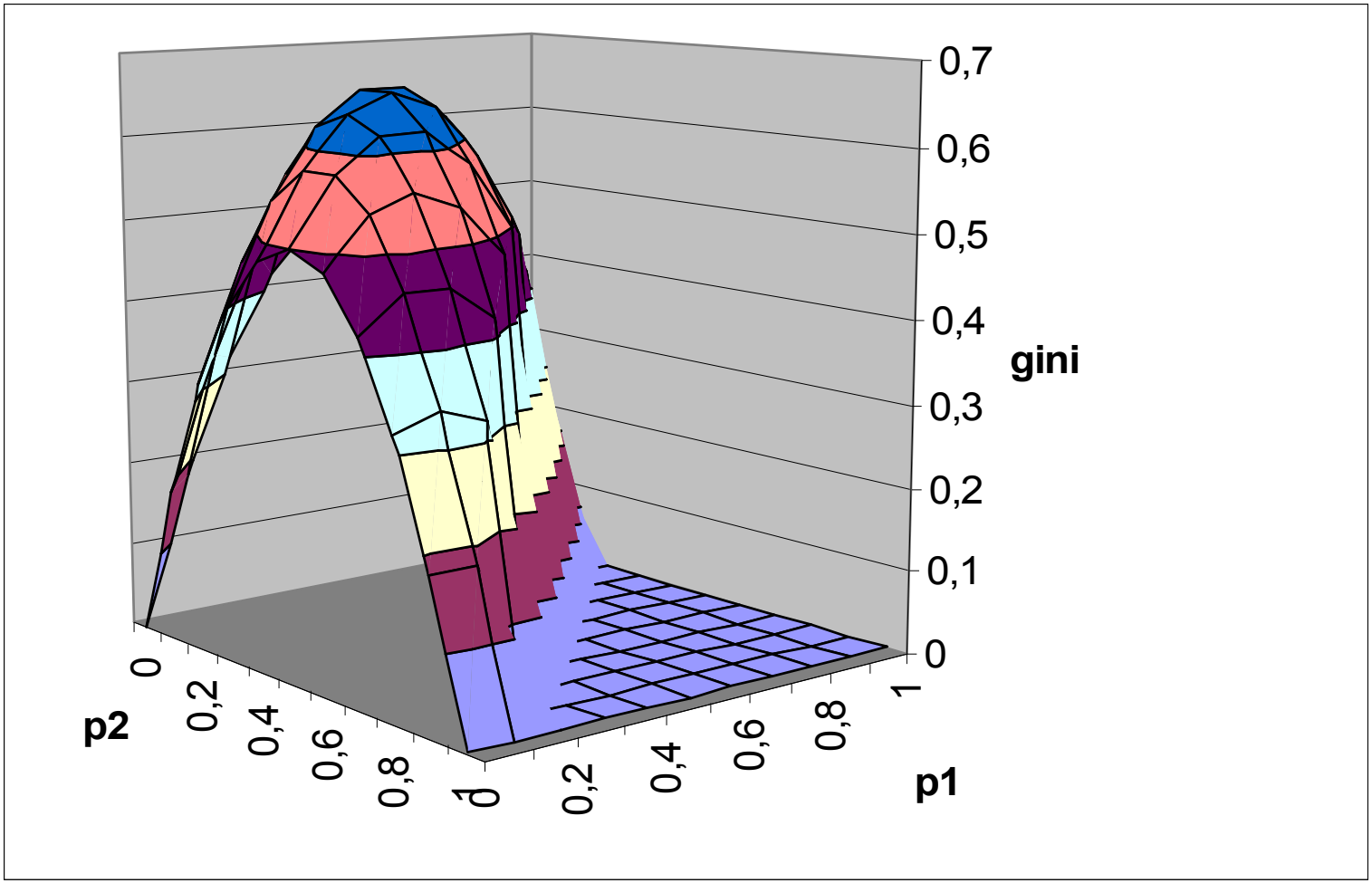
- The max of Gini index is achieved for $p_+=p_-=0,5$: gini(T)=0,5
- The min is achieved for $p_+=0$ or $p_-=0$: gini(T)=0
- For three classes

$$gini(T) = 1 - p_1^{\,2} - p_2^{\,2} - (1 - p_1 - p_2)^2$$

Gini index



Gini Index for Three Classes



Gini Index

- In general, the Gini index has a maximum for $p_1=p_2=...p_k=1/k$

$$gini(T) = 1 - \sum_{i=1}^{k} \left(\frac{1}{k}\right)^2 = 1 - k \times \frac{1}{k^2} = 1 - \frac{1}{k}$$

- and a minimum for $p_1=1$, $p_2=...p_k=0$: gini(T)=0

Gini index

- Gini index after the split on attribute X

$$gini_X(T) = \sum_{j=1}^{n} \frac{|T_i|}{|T|} \times gini(T_i)$$

- The attribute *X* that gives the lowest $gini_X(T)$ is chosen for the split
- Gini index is used by CART

Tests on Continuous Attributes

- Suppose continuous attribute X assumes m values in T
- Let $<V_1,V_2,\ldots,V_m>$ be the values ordered from the smallest to the largest
- The test $X \leq V_i$ divides the m values into two groups:
- $X \leq V_i : \{V_1,\ldots,V_i\}$
- $X > V_i : \{V_{i+1},\ldots,V_m\}$
- So we have m-1 candidates for the threshold: V_1, ..., V_{m-1}
- Each candidate must be evaluated

Tests on Continuous Attributes

- Evaluation of a test:
  - Sort the examples on the basis of the attribute X
  - Count the examples
    - Let $e_{j,i}$ be the number of examples that have value V_i for X and belong to class c_j for j=1,…,k, i=1,…,m-1
    - Let e_j be the number of examples that belong to class c_j for j=1,…,k
  - Each test has two possible outcomes: yes or no

Tests on Continuous Attributes

- $d_{j,yes}$=examples of class j in branch $X \leq V_i$
- $d_{j,no}$=examples of class j in branch $X > V_i$
- For j=1 to k $d_{j,yes}=0$ $d_{j,no}=e_j$
- For i=1 to m-1
  - Let the test be $X \leq V_i$
  - For j=1 to k
    - $d_{j,yes}:=d_{j,yes}+e_{j,i}$
    - $d_{j,no}:=d_{j,no}-e_{j,i}$
  - The heuristic for attribute X, threshold V_i can be computed from the values $d_{1,yes},\ldots,d_{k,yes},d_{1,no},\ldots,d_{k,no}$

Attributes with Unknown Value

- If the training set contains examples with one or more attributes unspecified
- For example

<?,72,90,T,P>

- How do we take into account this example when computing the heuristic and when splitting the example set?

Attributes with Unknown Value

- Test evaluation:
  - Consider a discrete attribute X with n values $\{x_1,\ldots,x_n\}$
  - Let F be the set of examples of T that have X known
  - F provides us with a distribution of examples into values and class $P(x_i,c_j)$
  - We assume that the unknown values have the same distribution $P(x_i,c_j)$ with respect to the attribute and the class
  - Therefore the examples with unknown values do not alter the value of the entropy or of the Gini index

Attributes with Unknown Value

- Entropy of the partitioning: the unknown value is considered as a value of its own
- Let T_{n+1} be the set of examples of T with unknown value for X

$$H(X) = -\sum_{i=1}^{n+1} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right)$$

Attributes with Unknown Value

- c4.5 further penalizes attributes with unknown values by multiplying the gain by the probability that the attribute is known
- If F is the set of examples of T with X known

$$gain(X) = \frac{|F|}{|T|} \times (H(F) - H_X(F))$$

Example

- Suppose that in the example database the case D6 :
  Outlook=overcast, Temperature=72, Humidity=90, Windy=T
- is replaced by
  Outlook=?, Temperature=72, Humidity=90, Windy=T
- The frequencies of Outlook over F are

Outlook	Play	Don't Play	Total
Sunny	2	3	5
Overcast	3	0	3
Rain	3	2	5
Total	8	5	13

Example

Outlook	Play	Don't Play	Total
Sunny	2	3	5
Overcast	3	0	3
Rain	3	2	5
Total	8	5	13

- $H(F)=-8/13*\log_2(8/13)-5/13*\log_2(5/13)=0.961$
- $H_{Outlook}(F)=5/13*(-2/5*\log_2(2/5)-3/5*\log_2(3/5))$
  $+3/13*(-3/3*\log_2(3/3)-0/3*\log_2(0/3))$
  $+5/13*(-3/5*\log2(3/5)-2/5*\log2(2/5))$
  $=0.747$
- gain(Outlook)=13/14*(0.961-0.747)=0.199 (slightly less than before)

Example

- H(Outlook)=-5/14*$\log_2$(5/14)-3/14*$\log_2$(3/14)-5/14*$\log_2$(5/14)-1/14*$\log_2$(1/14)=1.809
- gainratio(Outlook)=0.199/1.809=0.110

Partitioning

- In which subset of a node with test on X do we put an example with X unknown?
- Partitioning is generalized in a probabilistic sense:
  - Each example of T is assigned a weight, initially 1
  - Each example is inserted in every subset T_i with a weight
- If we partition on discrete attribute X
  - If example e, with weight w in T, has $X=x_i$, e is put in T_i with weight w and in T_j with $j \neq i$ with weight 0
  - If example e, with weight w in T, has X=?, e is put in T_i with weight $w \times P(x_i)$
  - $P(x_i)$ can be estimated by relative frequency in F

Example

- Partitioning according to Outlook
- No problem for the 13 cases of F
- Example D6 is assigned to sets T_{sunny}, $T_{overcast}$ and T_{rain} with weights 5/13, 3/13 e 5/13 respectively

Example

- Consider T_{sunny}

No	Outlook	Temp	Humidity	Windy	Class	Weight
D1	sunny	75	70	T	P	1
D2	sunny	80	90	T	N	1
D3	sunny	85	85	F	N	1
D4	sunny	72	95	F	N	1
D5	sunny	69	70	F	P	1
D6	?	72	90	T	P	5/13

- If T_{sunny} is partitioned on Humidity with threshold 75, we get the following distribution:
  - Humidity ≤ 75 2 class P, 0 class N
  - Humidity > 75 5/13 class P, 3 class N

Example

- The second subset still contains examples from two classes but no test produces two subsets with at least two examples=> stop

Outlook=sunny

| Humidity ≤ 75: P (2.0)

| Humidity > 75: N (3.4/0.4)

Outlook=overcast: P (3.2)

Outlook=rain

| Windy=True: N (2.4/0.4)

| Wndy=False: P (3.0)

Output Interpretation

Numbers (A/B) associated to leaves

- A=total weight of examples associated to the leaf
- B=total weight of examples associated to the leaf that are misclassified
- For example

N (3.4/0.4)

- Means that 3.4 cases belong to the leaf of which 0.4 do not belong to class N

Classification of Unseen Cases with All Values Known

- An unseen (new) case e with all attributes known is classified by
  - Traversing the tree from the root by following the branches that correspond to the values of the case
  - Suppose that the leaf

    Pos (A/B)

    is reached, then e is classified as Pos with probability (A-B)/A and Neg with probability B/A
  - In the case of more than two classes, the distribution of examples into classes at the leaf is used

Classification of Unseen Cases with Unkown Values

- T: node with a test on X, e has X unknown
- e is associated with a weight w, initially set to 1
- When traversing the tree, if the attribute at the current node is unknown in e we explore all subtrees.
- Subtree T_i is explored by assigning e to T_i with weight $w \times P(x_j)$
- $P(x_j)$ is estimated by considering by relative frequency over T
- The information about relative frequencies are stored in the leaves that are descendants of T

Classification of Unseen Cases with Unkown Values

- In the end more than one leaf will be reached
- Let L be the set of leaves that are reached
- Let w_l=P(l|e) be the weight of e that reaches l
- Let P(c_j|l) be the probability that an example in l belongs to class c_j

(estimated by relative frequency)

$$P(c_j \mid e) = \sum_{l \in L} P(c_j, l \mid e) = \sum_{l \in L} P(c_j \mid l, e) \times P(l \mid e) =$$

$$\sum_{l \in L} P(c_j \mid l) \times P(l \mid e)$$

Example

- Unseen example e

  Outlook=sunny, Temperature=70, Humidity=?, Windy=F
- Outlook=sunny => first subtree
- Humidity=?=> we cannot determine whether Humidity ≤ 75
- We follow both branches with weights
  - Branch Humidity ≤ 75: w=2.0/5.4=0.370
  - Branch Humidity > 75: w=3.4/5.4=0.630

Example

- Leaf humidity $\leq$ 75:
  - P(P|l)=100%
  - P(N|l)=0%
- Leaf humidity > 75
  - P(N|l)= 3/3.4=88%
  - P(P|l)= 0.4/3.4=12%
- Overall we get
  - P(P|e)=0.370*100%+0.630*12%=44%
  - P(N|e)=0.630*88%=56%

Observations on c4.5 Search (1)

- c4.5 performs a hill-climbing search in the space of all the possible decision trees starting from the simplest hypothesis and going to more complex hypotheses
- Observations:
  - The space of possible decision trees is equivalent to the powerset of X so for c4.5 the hypothesis space surely includes the target concept
  - c4.5 maintains a single hypothesis during the search. This contrasts with Candidate-Elimination that maintains the set of all the hypotheses coherent with the examples.

Observations on c4.5 Search (2)

- c4.5 does not performs backtracking during the search. So it runs the risk of incurring in a solution only locally optimal
- c4.5 uses all the training examples at every step for deciding how to refine the tree, differently from the methods such as Find-S or Candidate-Elimination that take decisions in an incremental way on the basis of individual examples. The result is that the search of c4.5 is less sensible to errors in the single examples.

c4.5 Inductive Bias (1)

- c4.5 inductive bias is [Mit97]:
  - Shortest trees are preferred to the longer ones. The trees that put closer to the root the attributes with the highest information gain are preferred.
- Even if c4.5 hypothesis space is the powerset of X, thanks to this inductive bias, c4.5 is able to generalize anyway
- Differences with Candidate-Elimination: Candidate-Elimination searches in a complete way an incomplete hypothesis space while c4.5 searches in an incomplete way a complete hypothesis space

c4.5 Inductive Bias (2)

- c4.5: preference bias (or search bias)
- Candidate-Elimination: restriction bias (or language bias)

- In general, a preference bias is better because it allows to work with a complete hypothesis space
- Some algorithms use both

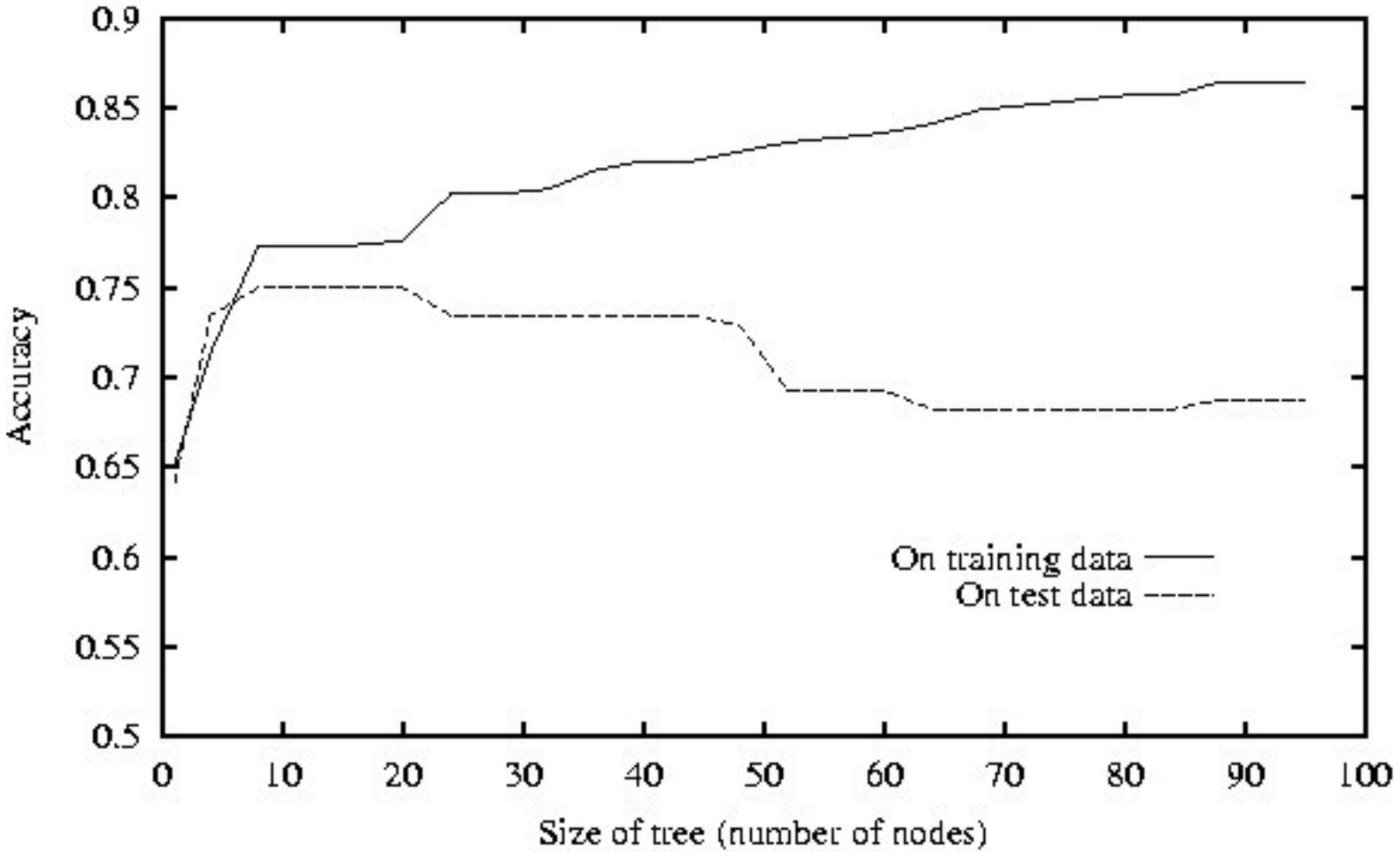
Why preferring shorter hypotheses?

- Question largely debated by philosophers
- William of Occam in 1320 was the first to tackle the problem [Mit97]:
  - Occam's razor: prefer the simplest hypothesis that describes the data
- Is this true?
  - In favor: simplest hypothesis are fewer than more complex ones, so it is more difficult to find a short hypothesis that describes the data by chance.
  - Against: if we define a small set of hypotheses (e. g. the trees with 17 nodes), then we should prefer those.

Overfitting

- A hypothesis overfits the data when there exists a simpler hypothesis less accurate on the training set but more accurate on the universe
- Overfitting may happen when the training set contains errors or when it is small

Overfitting



Overfitting

- Example: consider the previous training set plus

  <sunny,80,70,strong,N>

- which should be positive but it is erroneously classified as negative
- The tree h' learned from the new training set is more complex than the tree h learned from the original training set
- However, since the new example is incorrect, h' will make more mistakes than h on U

Approaches to Avoid Overfitting

- Stop the growth before reaching a tree that perfectly classifies the training examples (prepruning)
- Grow the tree and prune it afterwards (postpruning)

How to Determine the Optimal Dimension of the Tree

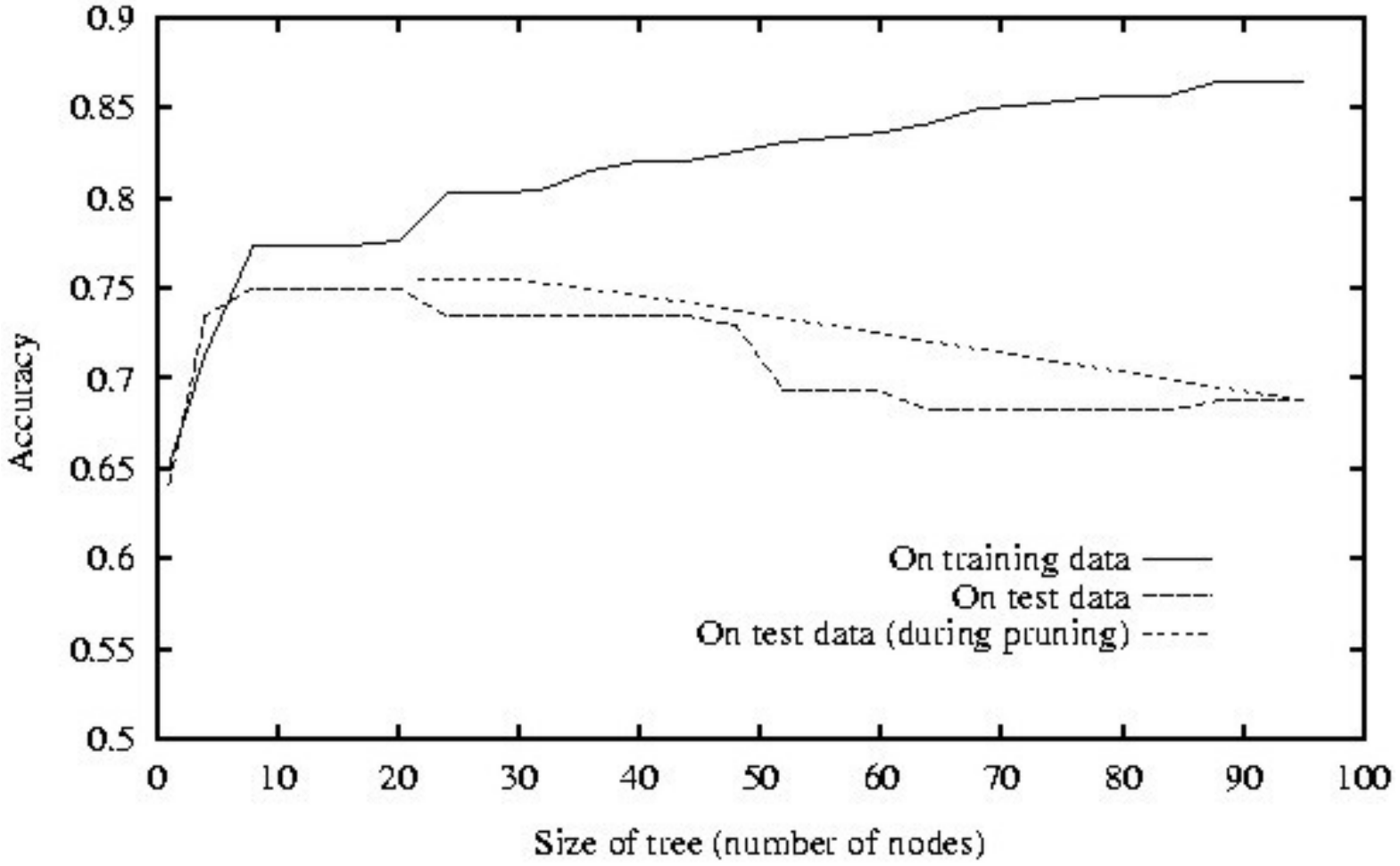
- Using a separate example set for evaluating the tree ("training and validation set" approach)
- Using all the available examples for training but using a statistical test to decide whether to keep a node or not
- Using an explicit measure of the complexity of encoding the tree and the examples that are exceptions and choose a tree that (locally) minimizes this measure (minimum description length principle)

Reduced error pruning

- "Reduced error pruning" is a postpruning and "training and validation set" approach.
- Divide the data into a training and a validation set
- While a further pruning reduces the accuracy on the validation set
  - Evaluate the impact on the validation set of pruning every node
    - Pruning a node means replacing it with a leaf with label the most frequent class in the node
  - Prune the node that gives the best accuracy on the validation set

- Problems when the data are limited

Reduced error pruning



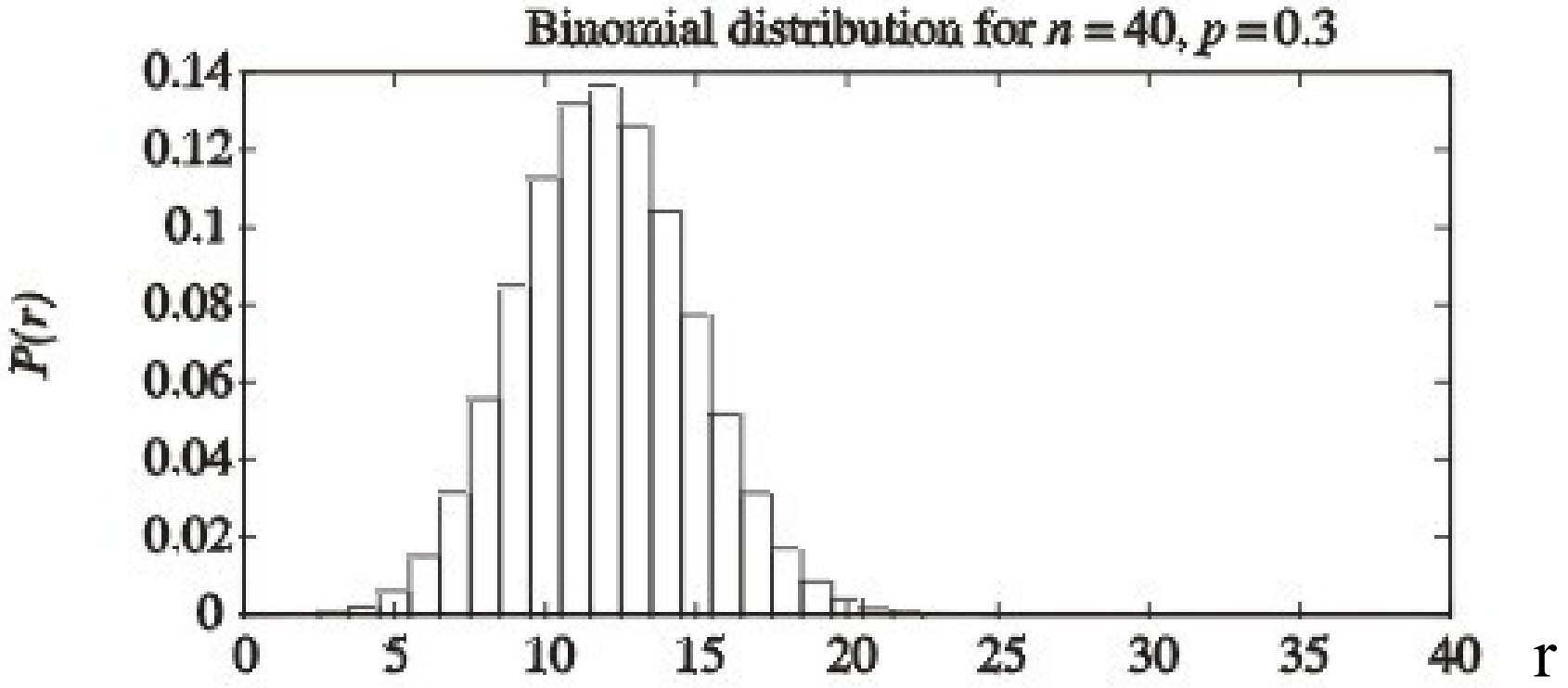
Simplification of the Tree in c4.5

- “Pessimistic pruning”: it is a postpruning that uses a statistical test rather than a testing set
- The tree that is obtained is simpler and is often more accurate when evaluating new cases because it reduces the overfitting
- Pruning by substitution: a single leaf in place of a tree or a sub-tree in place of a tree
- It is based on an estimate of the error rate

Simplification of the Tree in c4.5

- Consider a leaf that covers N training cases of which E erroneously
- The error on the training set is E/N
- Consider E as events in N trials
- We ask what is the probability of the error event on the entire population of cases covered by the leaf
- The probability of the error has a binomial distribution

Binomial Distribution



- The binomial distribution describes the probability that r heads are obtained when throwing n times a coin for which the probability of landing head is p

Binomial Distribution

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r} = \binom{n}{r} p^r (1-p)^{n-r}$$

$$E[r] = np$$

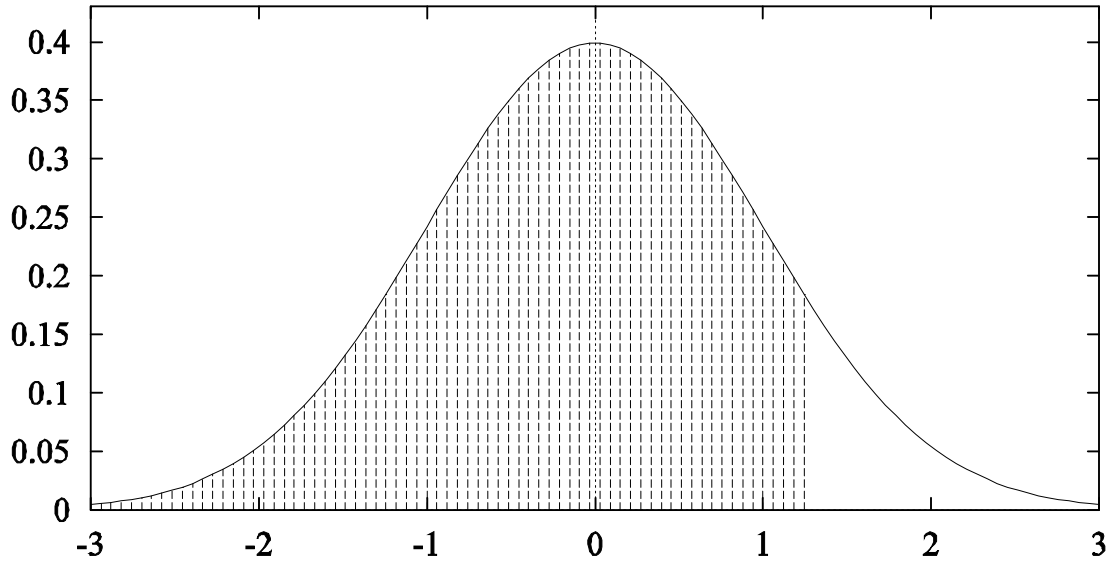
$$Var[r] = np(1-p)$$

Estimate of p

- Suppose you observed r heads in n throws, you want to estimate p
- In particular, you want
  1. an interval [a,b] in which you are sure that p falls with a certain probability N%, i.e. P(a<p<b)=N%
     - [a,b] is called a (two-tailed) confidence interval,
     - N% is called the confidence level, or
  1. a number a such that P(a<p)=L%
     - [a,1] is a one-tailed confidence interval
     - L% is called the probability of the upper tail, or
  1. a number b such that P(p<b)=U%
     - [0,b] is a one-tailed confidence interval
     - U% is called the probability of the lower tail

One-tailed Confidence Interval

- We consider one-tailed confidence of the form [0,b]



Estimate of p

- Given r, n, and N%, L% or U% there are algorithms that give you (a, b), a and b
- Either directly using the binomial distribution or
- By approximating the binomial distribution with a normal distribution
  - This approximation is good when n>30 or when np(1-p)>5

Confidence Interval

- Let's go back to learning
- Suppose you have E error over N examples
- E corresponds to r, N to n and the true error probability to p
- We want to estimate the true error probability
- We use one-tailed confidence intervals [0,b]
- Let CF be the probability of the confidence interval (U %)
- Let $U_{CF}(E,N)$ be the upper bound of the confidence interval, i.e. let $U_{CF}(E,N)=b$

c4.5 Tree Pruning

- c4.5 considers $N*U_{CF}(E,N)$ instead of E as an estimate of the number of errors in a leaf
- c4.5 uses CF=75% by default
- The estimate of the number of errors of a tree is the sum of the estimate of the number of errors in each leaf
- c4.5 replaces a tree with a leaf if the estimated number of errors in the leaf is smaller than those of the tree
- The same for replacing trees with sub-trees

Example: vote Dataset

- Instances: members of the US congress
- Attributes: votes expressed by the member regarding 16 questions
- Classes: democratic, republican

Example: vote Dataset

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151.0)
|   adoption of the budget resolution = u: democrat (1.0)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6.0)
|   |   education spending = y: democrat (9.0)
|   |   education spending = u: republican (1.0)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97.0/3.0)
|   synfuels corporation cutback = u: republican (4.0)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2.0)
|   |   duty free exports = u: republican (1.0)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5.0/2.0)
|   |   |   education spending = y: republican (13.0/2.0)
|   |   |   education spending = u: democrat (1.0)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0.0)
|   water project cost sharing = y: democrat (4.0)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0.0)
|   |   mx missile = y: democrat (3.0/1.0)
|   |   mx missile = u: republican (2.0)
```

Example: vote Dataset

Simplified tree:

```
physician fee freeze = n: democrat (168.0/2.6)
physician fee freeze = y: republican (123.0/13.9)
physician fee freeze = u:
|   mx missile = n: democrat (3.0/1.1)
|   mx missile = y: democrat (4.0/2.2)
|   mx missile = u: republican (2.0/1.0)
```

Example: vote Dataset

- The sub-tree
  education spending = n: democrat (6.0)
  education spending = y: democrat (9.0)
  education spending = u: republican (1.0)
- was replaced by the leaf democrat because
- First leaf E=0 N=6 $U_{75\%}(0,6)=0.206$
- Second leaf E=0 N=9 $U_{75\%}(0,9)=0.143$
- Third leaf E=0 N=1 $U_{75\%}(0,1)=0.750$
- So the estimate of the number of errors for the sub-tree is

$$6*0.206+9*0.143+1*0.750=3.273$$

Example: vote Dataset

- If the sub-tree
  education spending = n: democrat (6.0)
  education spending = y: democrat (9.0)
  education spending = u: republican (1.0)
- was replaced by the leaf democrat, it would cover 16 cases with 1 error so the estimate of the number of errors would be:

$$16*U_{75\%}(1,16)=16*0.157=2.512$$

- Since the estimate is lower for the leaf than for the sub-tree, this is replaced by the leaf

Example: vote Dataset

- The sub-tree immediately above takes the form

adoption of the budget resolution = y: democrat (151.0)
adoption of the budget resolution = u: democrat (1.0)
adoption of the budget resolution = n: democrat (16/1)

- The estimate of the number of errors for the sub-tree is

$$151*U_{75\%}(0,151)+1*U_{75\%}(0,1)+2.512=4.642$$

- If the sub-tree was replaced by the leaf democrat the estimate of the number of errors would be

$$168*U_{75\%}(1,168) =2.610$$

- Therefore it is convenient to perform the substitution

Pruned Trees

- The numbers (N/P) that appear close to the leaves mean:
  - N number of training example covered by the leaf
  - P number of predicted errors if N new cases would be classified by the tree ($N*U_{CF}(E,N)$)
- Estimate of the error on new cases of the pruned trees=errors predicted in the leaves / number of cases in the training set
- In the vote dataset the sum of errors predicted in the leaves is 20.8 over a total of 300 cases in the training set => 6.9% error

Example: vote Dataset

Evaluation on training data (300 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
25	8(2.7%)	7	13(4.3%)	(6.9%) <<

Evaluation on test data (135 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
25	7(5.2%)	7	4(3.0%)	(6.9%) <<

Example: vote Dataset

- The error on the training set is higher for the pruned tree than for the unpruned tree
- **but, as expected, the error on the unseen cases is lower**

Computational Complexity

- Worst case analysis
  - A: number of attributes
  - N: number of examples
  - Hypothesis: an attribute can be used only once

Algorithm	Training
c4.5 (only discrete attributes)	$O(A^2N)$
c4.5 (only continuous attributes)	$O(A^2N\log_2 N)$

Because it is necessary to order the values of continuous attributes

- The average case can be very different

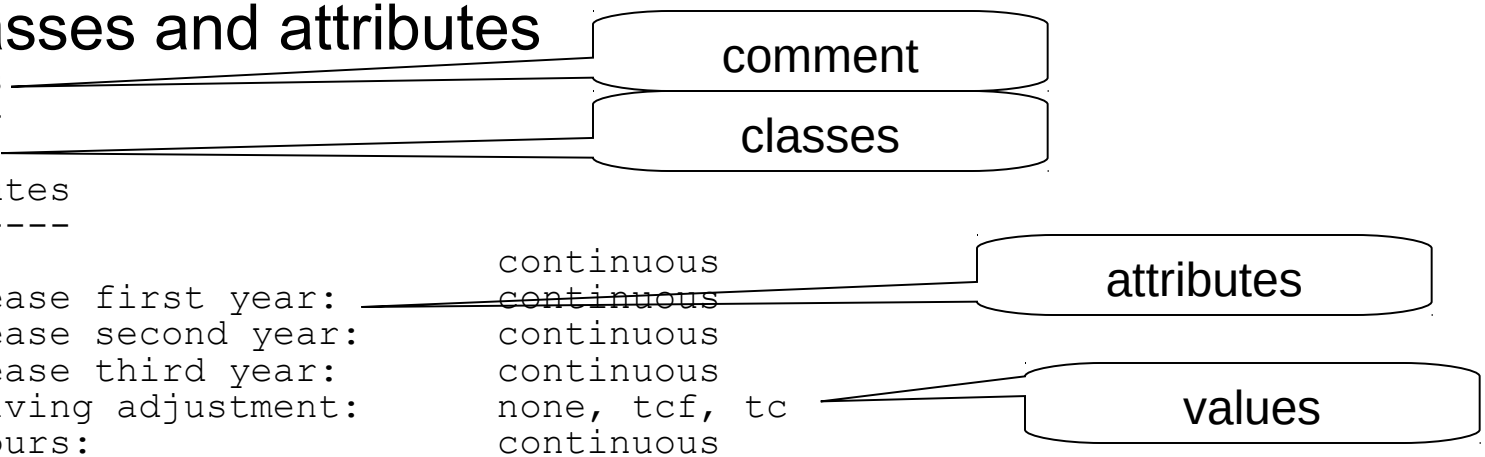
Examples of Usage

- Results of the labor negotiations in Canada in the years 1987--88.
- Concepts: acceptable or unacceptable contract.
- Contract: 16 attributes, not all applicable to all cases.
- Classes: acceptable (good) or not acceptable (bad).
- Aim: obtaining a system able to judge the acceptability of a new contract.

Data Divided in Some File

- labor-neg.names: description of the problem in term of classes and attributes

```
|  Classes
|  -------
good, bad.
|  Attributes
|  ----------
duration:                               continuous
wage increase first year:               continuous
wage increase second year:              continuous
wage increase third year:               continuous
cost of living adjustment:              none, tcf, tc
working hours:                          continuous
pension:                                none, ret_allw, empl_contr
standby pay:                            continuous
shift differential:                     continuous
education allowance:                    yes, no
statutory holidays:                     continuous
vacation:                               below average, average, generous
longterm disability assistance:         yes, no
contribution to dental plan:            none, half, full
bereavement assistance:                 yes, no
contribution to health plan:            none, half, full
```



Training set

- Labor-neg.data: attributes not applicable or unknown: “?”

```
1,5.0,?,?,?,40,?,?,2,?,11,average,?,?,yes,?,good
3,3.7,4.0,5.0,tc,?,?,?,?,yes,?,?,?,?,yes,?,good
3,4.5,4.5,5.0,?,40,?,?,?,?,12,average,?,half,yes,half,good
2,2.0,2.5,?,?,35,?,?,6,yes,12,average,?,?,?,?,good
3,6.9,4.8,2.3,?,40,?,?,3,?,12,below average,?,?,?,?,good
3,3.5,4.0,4.6,none,36,?,?,3,?,13,generous,?,?,yes,full,good
1,3.0,?,?,none,36,?,?,10,no,11,generous,?,?,?,?,good
2,4.0,5.0,?,tcf,35,?,13,5,?,15,generous,?,?,?,?,good
2,2.5,3.0,?,?,40,none,?,?,?,11,below average,?,?,?,?,bad
3,3.5,4.0,4.6,tcf,27,?,?,?,?,?,?,?,?,?,?,good
2,4.5,4.0,?,?,40,?,?,4,?,10,generous,?,half,?,full,good
1,6.0,?,?,?,38,?,8,3,?,9,generous,?,?,?,?,good
2,3.0,3.0,?,none,33,?,?,?,yes,12,generous,?,?,yes,full,good
3,2.0,2.5,?,?,35,none,?,?,?,10,average,?,?,yes,full,bad
3,4.5,4.5,5.0,none,40,?,?,?,no,11,average,?,half,?,?,good
2,2.5,2.5,?,?,38,empl_contr,?,?,?,10,average,?,?,?,?,bad
1,2.0,?,?,tc,40,ret_allw,4,0,no,11,generous,no,none,no,none,bad
3,2.0,2.5,2.0,?,37,empl_contr,?,?,?,10,average,?,?,yes,none,bad
2,4.5,4.0,?,none,40,?,?,4,?,12,average,yes,full,yes,half,good
```

Labor-neg.test: set of new cases, to be used to evaluate the learned tree (same formats of labor-neg.data)

Example

- Command to run c4.5 on the data:

c4.5 –f labor-neg –u

where the –f option is used to indicate the name of the set of files and –u means that the learned tree must be evaluated on the cases in labor-neg.test

Output

- Two decision trees:
  - *complete tree*
  - *pruned tree*

- Evaluation of the trees

Output: Complete Tree

```
C4.5 [release 8] decision tree generator        Sat May 18 17:05:35 1996
----------------------------------------

    Options:
        File stem <labor-neg>
        Trees evaluated on unseen cases

Read 40 cases (16 attributes) from labor-neg.data

Decision Tree:

wage increase first year <= 2.5 :
|   statutory holidays <= 10 : bad (6.0)
|   statutory holidays > 10 :
|   |   working hours <= 38 : good (2.3/1.0)
|   |   working hours > 38 : bad (3.0)
wage increase first year > 2.5 :
|   statutory holidays > 10 : good (21.2)
|   statutory holidays <= 10 :
|   |   wage increase first year <= 4 : bad (4.5/0.5)
|   |   wage increase first year > 4 : good (3.0)
```

Output: Pruned Tree

```
Simplified Decision Tree:

wage increase first year <= 2.5 : bad (11.3/2.8)
wage increase first year > 2.5 :
|   statutory holidays > 10 : good (21.2/1.3)
|   statutory holidays <= 10 :
|   |   wage increase first year <= 4 : bad (4.5/1.7)
|   |   wage increase first year > 4 : good (3.0/1.1)


Tree saved
```

Tree Evaluation

```
Evaluation on training data (40 items):

         Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate

          11    1( 2.5%)      7    1( 2.5%)    (17.4%)   <<

Evaluation on test data (17 items):

         Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate

          11    4(23.5%)      7    3(17.6%)    (17.4%)   <<


           (a)   (b)      <-classified as
          ----  ----
            10     1      (a): class good
             2     4      (b): class bad
```

Confusion matrix

Weka

- Weka is a machine learning suite
- It contains tools for:
  - Learning decision trees
  - Learning production rules
  - Discovering association rules
  - Clustering
- Open source, in Java
- Downloadable from
- http://www.cs.waikato.ac.nz/ml/weka/
- Connected to the book: [Wit05]

Weka

- Contains an implementation of c4.5 Release 8: j4.8
- Uses a format called ARFF (Attribute Relation File Format) for the input files

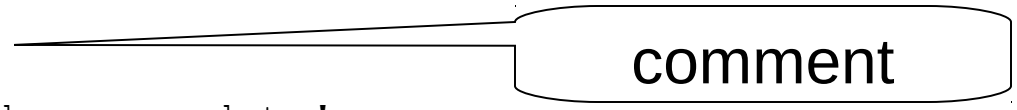
ARFF

- A single file
- Two sections:
  - Header
  - Data
- Header: attribute definition
  - @relation <dataset name>
  - @attribute <attribute name> {<val1>, <val2>, …, <valn>}
    - or
  - @attribute <attribute name> real
- The last attribute is usually used as the class

File labor.arff: Header

```
% labor.arff
@relation 'labor-neg-data'

@attribute 'duration' real
@attribute 'wage-increase-first-year' real
@attribute 'wage-increase-second-year' real
@attribute 'wage-increase-third-year' real
@attribute 'cost-of-living-adjustment' {'none','tcf','tc'}
@attribute 'working-hours' real
@attribute 'pension' {'none','ret_allw','empl_contr'}
@attribute 'standby-pay' real
@attribute 'shift-differential' real
@attribute 'education-allowance' {'yes','no'}
@attribute 'statutory-holidays' real
@attribute 'vacation' {'below_average','average','generous'}
@attribute 'longterm-disability-assistance' {'yes','no'}
@attribute 'contribution-to-dental-plan' {'none','half','full'}
@attribute 'bereavement-assistance' {'yes','no'}
@attribute 'contribution-to-health-plan' {'none','half','full'}
@attribute 'class' {'bad','good'}
```



Data Section

- @data tag followed by the description of the examples, one description per row (ending with a line feed) as in c4.5
- Every example is described by the list of values for every attribute separated by , (comma)
- Each value corresponds to the attribute in the same position in the header
- The declarations @relation, @attribute and @data are case insensitive.

Labor.arff: Data

```
@data
1,5,?,?,?,40,?,?,2,?,11,'average',?,?,'yes',?,'good'
3,3.7,4,5,'tc',?,?,?,?,'yes',?,?,?,?,'yes',?,'good'
2,2,2.5,?,?,35,?,?,6,'yes',12,'average',?,?,?,?,'good'
3,6.9,4.8,2.3,?,40,?,?,3,?,12,'below_average',?,?,?,?,'good'
...
```

The single quotes (‘) are necessary only if the values contain spaces

ARFF

- Other formats for the attributes
  - integer, string, date
- integer: the attribute can take only integer values
- string: the attribute can take arbitrary string values
  - String attributes are treated as nominal attributes

ARFF

- date: date attributes. Format:

  @attribute <name> date [<date-format>]

- where <date-format> is an optional string that specifies how the values are to be interpreted and printed. The default format is the ISO-8601 format "yyyy-MM-dd'T'HH:mm:ss“
- Example

```
@RELATION Timestamps
@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"
@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

ARFF

- date: date attributes. Format:

  @attribute <name> date [<date-format>]

- where <date-format> is an optional string that specifies how the values are to be interpreted and printed. The default format is the ISO-8601 format "yyyy-MM-dd'T'HH:mm:ss“
- Example

```
@RELATION Timestamps
@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"
@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

Weka Execution

- Double click on Weka (weka.jar)
- Click on Explorer
- Click on Open file… : select the file to be opened, for example labor.arff
- Click on Classify
- Choice of the algorithm: j48
- Setting the test options: for example, select Percentage Split to select a fraction of the examples to be used for testing
- Click on Start

Output

J48 pruned tree

---

wage-increase-first-year <= 2.5: bad (15.27/2.27)
wage-increase-first-year > 2.5
| statutory-holidays <= 10: bad (10.77/4.77)
| statutory-holidays > 10: good (30.96/1.0)

Number of Leaves : 3

Size of the tree : 5

Time taken to build model: 0.08 seconds

Output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	17	85	%
Incorrectly Classified Instances	3	15	%
Kappa statistic	0.6875		
Mean absolute error	0.2806		
Root mean squared error	0.3729		
Relative absolute error	58.3712 %		
Root relative squared error	72.0645 %		
Total Number of Instances	20		

Output

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.667	0	1	0.667	0.8	bad
1	0.333	0.786	1	0.88	good

=== Confusion Matrix ===

a	b	<-- classified as
6	3	a = bad
0	11	b = good

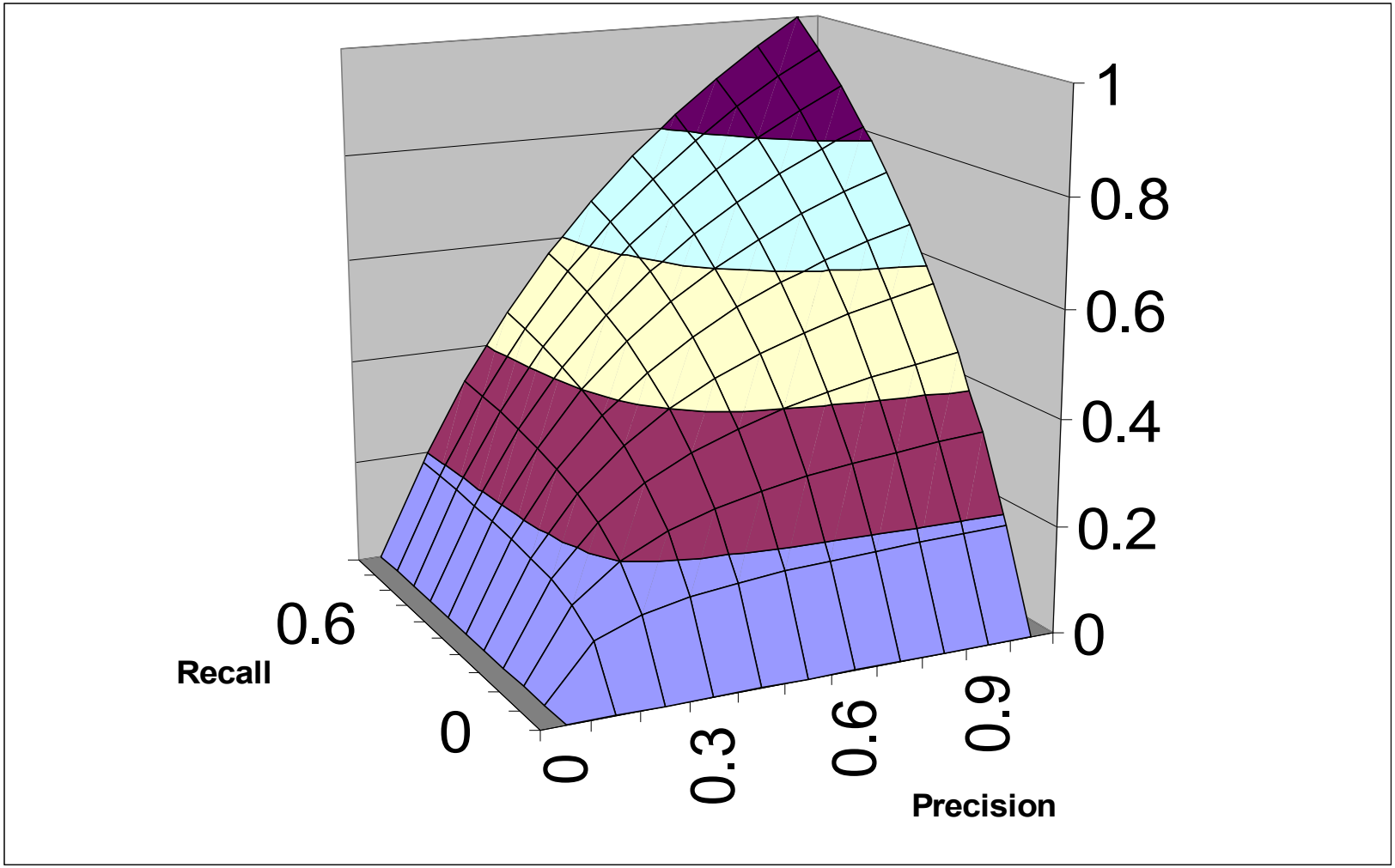
Output Explanation

- Confusion matrix:

pos	neg	<-- classification
TP	FN	P pos
FP	TN	N neg

- TP=true positive, FN=false negative, FP=false positive, TN=true negative, P=TP+FN=positive, N=FP+TN=negative
- **Accuracy**=(TP+TN)/(TP+TN+FN+FP)
- **Error rate**=(FP+FN)/(TP+TN+FN+FP)=1-Accuratezza
- **TP Rate(TPR)**=**Sensitivity=Recall=**TP/(TP+FN)=TP/P
- **FP Rate(FPR)**=FP/(FP+TN)=FP/N
- **FN Rate(FNR)**=FP/(TP + FN) =FP/P=1-TPR
- **TN Rate(TNR)**=**Specificity**=TN/(FP+TN)=TN/N=1–FPR
- **Precision**=TP/(TP+FP)
- Couples: TPR-FPR, Sensitivity-Specificity, Recall-Precision
- **F-meaure**=2*Precision*Recall/(Precision+Recall)

F-measure



Useful Links

- C4.5

  http://www.rulequest.com/Personal/

  C source code

- Weka: open source suite of machine learning algorithms written in Java

  http://www.cs.waikato.ac.nz/ml/weka/

References

[Mit97] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

[Qui93b] J. R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann Publishers, San Mateo, California, 1993

[Qui96] J. R. Quinlan, *Improved Use of Continuous Attributes in C4.5*, Journal of Artificial Intelligence Research, 4, pag. 77--90, 1996. ftp://ftp.cs.cmu.edu/project/jair/volume4/quinlan96a.ps

[Wit05] I.H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Second Edition, Morgan Kaufmann, 2005.