MACHINE LEARNING

- Definition 1:
  - Learning is constructing or modifying representations of what is being experienced [Michalski 1986], p. 10
- Definition 2:
  - Learning denotes changes in the system That are adaptive in the sense That they enable the system to do the same task or tasks drawn from the same population blackberries efficiently and more effectively the next time [Simon 1984], p. 28

USES OF ML

- A) knowledge extraction
  - to be used for the operation of knowledge-based systems (for example, for classification systems)
  - for scientific purposes, for the discovery of new facts and theories through observation and experimentation
- B) improving the performance of a machine
  - for example, improving the motion and cognitive abilities of a robot
  - for game playing

ML techniques

- symbolic techniques
  - different types of representation
    - value attribute representation
    - representation of the first order
- Statistical techniques
- Neural networks

INDUCTIVE LEARNING

- Inductive learning: the system starts from observations coming from a trainer or from the surrounding environment and generalizes, gaining knowledge that, hopefully, is also valid for not yet observed cases (**induction**).
- Two types of inductive learning:
  - **Learning from examples**: the knowledge is gained from a set of positive examples that are instances of the concept to be learnt and negative examples which are non-instances of the concept
  - **Learning regularity**: there is not a specific concept to be learned, the goal is to find regularities (i,e. common features, patterns) in data

LEARNING FROM EXAMPLES
MAIN CONCEPTS

- **Universe** U: set of all domain objects
- **Concept** C: subset of objects in the domain $C \subseteq U$
- A **description language for objects** Lo
- A **description language for concepts** Lc
- A procedure that interprets both languages and checks whether the description Dc of the concept C is satisfied by the description Dx of an object x (**Dc covers Dx**).

LEARNING FROM EXAMPLES MAIN CONCEPTS

- Informally:
  - Learning a concept C means finding a description of C that allows to tell if an object $x \in U$ is an instance of C, i,e. if $x \in C$.

LEARNING FROM EXAMPLES
MAIN CONCEPTS

- **Fact**: description of an object
- **Example of a concept C**: labeled fact. The label is + if the object is an instance of C, while the label is - if the object is not an instance of C.
- **Training set**: set E of examples (labeled facts). E is composed by all the positive examples E + and all the negative examples E-

LEARNING FROM EXAMPLES MAIN CONCEPTS

- Hypothesis: we have a description of the concept to be learned
- If an fact satisfies the hypothesis we say that the hypothesis **covers** the fact.
- Function for the test coverage:

  covers (H, e)

  returns true if e covered by H and false otherwise
- Extension to sets of examples:

  covers (H, E) = {e ∈E | covers (H, e) = true}

LEARNING FROM EXAMPLES
MAIN CONCEPTS

LEARNING PROBLEM DEFINITION

- Given a set E of positive and negative examples of a concept C, expressed in an object description language Lo,
- Find a hypothesis H, expressed in a given concept description language Lc, such that:
  - every positive example e+ $\in$ E is covered by H
  - no negative example e- $\in$ E- is covered by H

COMPLETENESS and CONSISTENCY

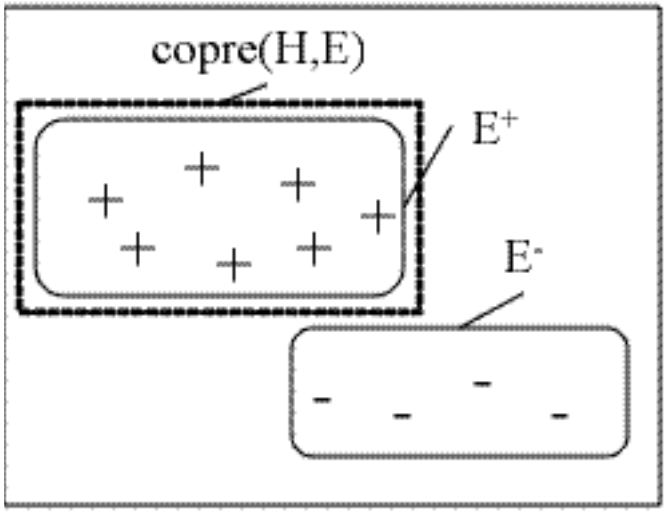
- A hypothesis H is **complete** if it covers all positive examples

$$covers(H, E+) = E+$$

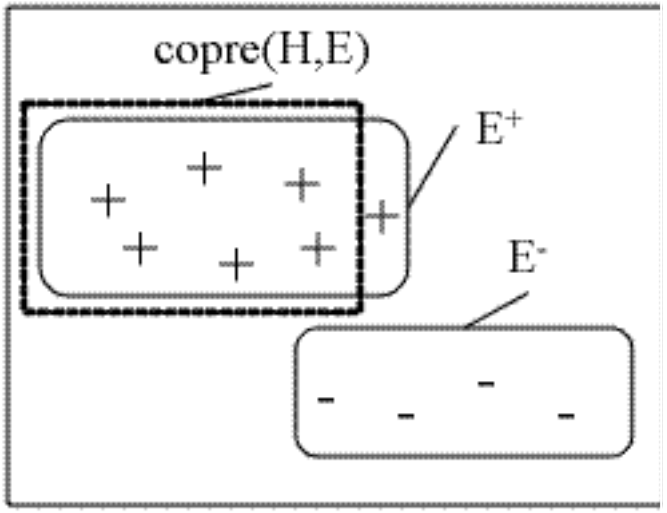
- A hypothesis H is **consistent** if does not cover any negative example

$$covers(H, E-) = \varnothing$$

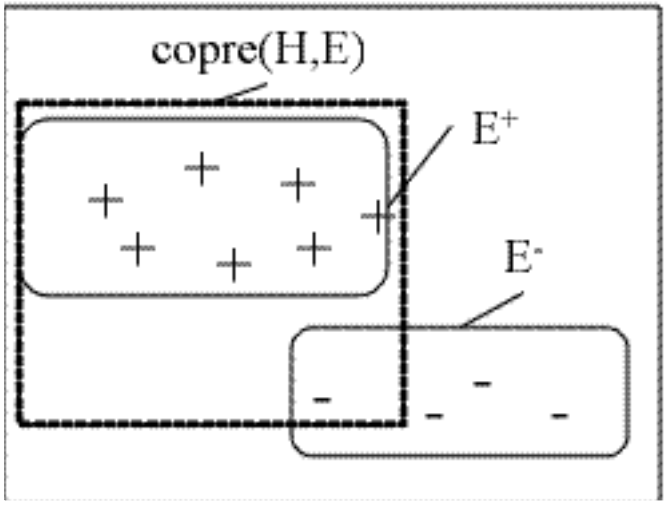
H: completa, consistente



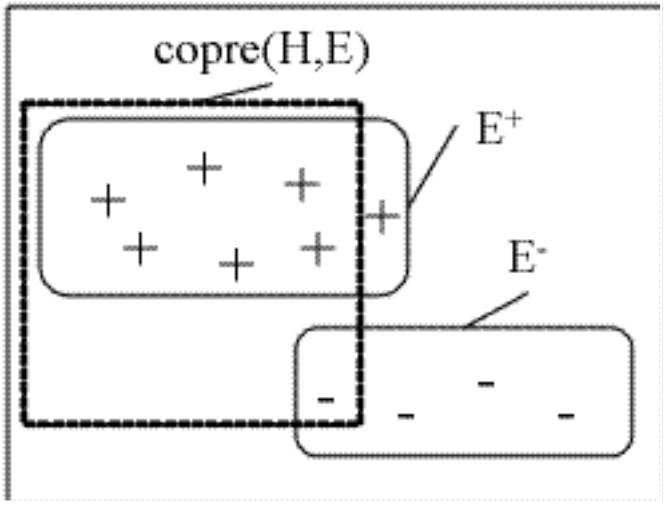
H: incompleta, consistente



H: completa, inconsistente



H: incompleta, inconsistente



KNOWLEDGE REPRESENTATION

- Languages used for the representation of examples and concepts:
  - attribute-value languages
  - relational languages
  - first order languages

- The richer the language, the more complex the learning problem

ATTRIBUTE-VALUE LANGUAGES

OBJECT DESCRIPTION LANGUAGE

- Fixed number of attributes for each instance
- Each instance is described by values assumed by the set of attributes.
- Attributes can be:
  - Boolean or binary
  - nominal
  - ordinal
  - Numeric
- if k are the attributes, each instance can be represented as a point in a k-dimensional space

EXAMPLE

- Universe of athletes
- Instances described by attributes
  - height, weight, body-type

- Example instance

  height = 1.85m, weight = 110kg, body-type = robust

ATTRIBUTE VALUE LANGUAGES

- Equivalent to propositional logic:
  - any equality or inequality can be seen as a proposition (no variables and no terms)
  - there are no variables, quantifiers and predicates with arity > 1

PRODUCTION RULES

CONCEPT DESCRIPTION LANGUAGE

- Production rules
  - Antecedent: conjunctions and disjunctions of equalities or inequalities between an attribute and a value,
  - Consequent: the concept (class)
- Example:
- Concept: soccer player
- Example description of the concept

weight> 100 and (bosy-type= normal or body type = robust) → soccer_player

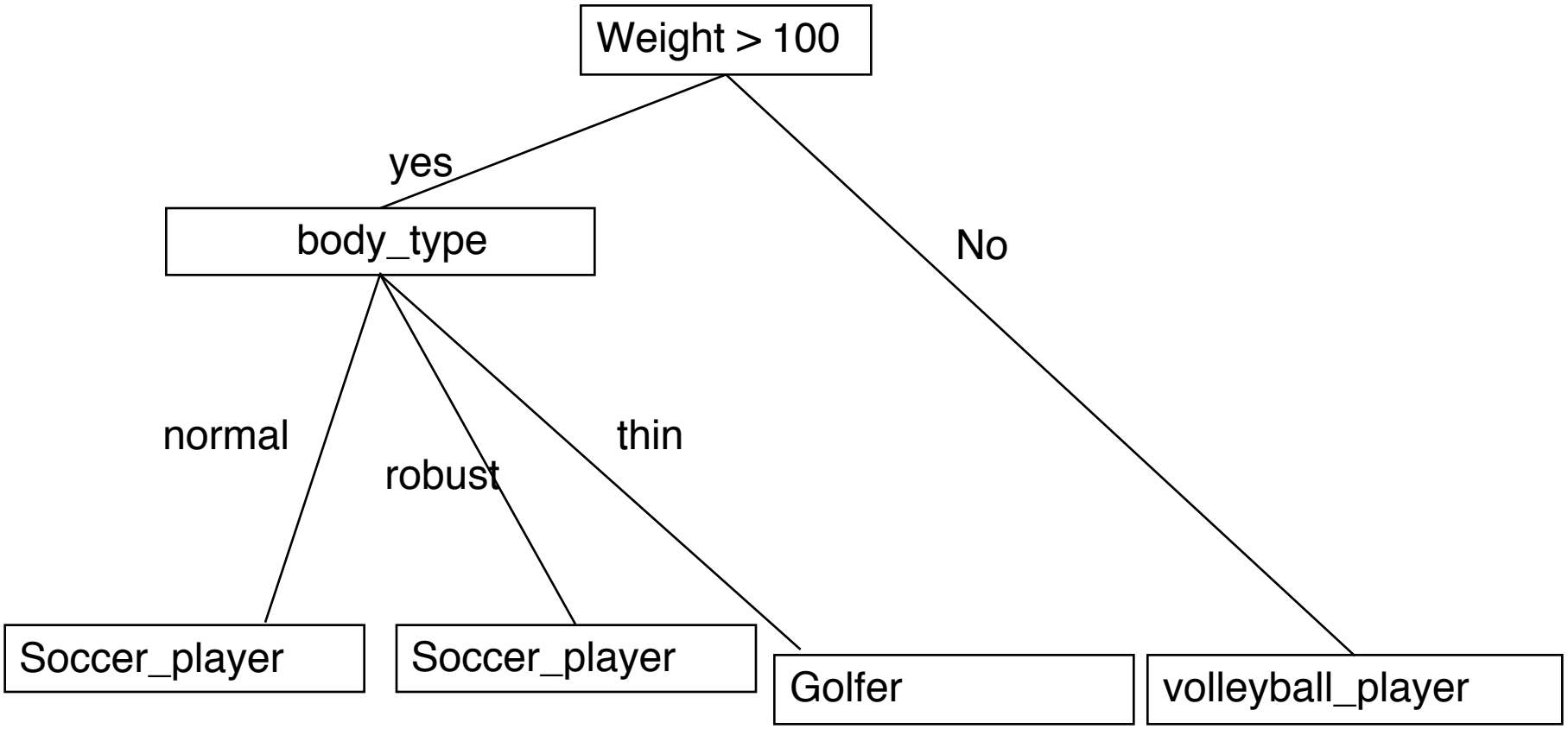
DECISION TREES

CONCEPT DESCRIPTION LANGUAGE

- Decision trees
  - each node corresponds to a test on an attribute (equality, inequality) and each branch that starts from the node is labeled with the test result
  - each leaf corresponds to a class (concept)

- Fully equivalent to production rules

DECISION TREE (example)



RELATIONAL LANGUAGES

OBJECT DESCRIPTION LANGUAGE

- Attribute-value languagesare not suited to represent instances composed of subparts.
- Example: jones family, components:
  - name: dave, son: mike, father: ron
  - name: mike, son: junior, father: dave
  - name: junior, father: mike
- The description of the family could be transformed into a list of attribute value pairs providing a number of attributes equal to the product of the maximum number of components for the maximum number of attributes per component: waste of memory

RELATIONAL LANGUAGES

OBJECT AND CONCEPT DESCRIPTION LANGUAGE

- The attribute-value languages are therefore inefficient to represent these universes. Relational languages are used instead (see example above).

- These languages allow descriptions of concepts that may contain variables and quantifiers (non-propositional languages).

- Description of the family concept with a grandparent

$$\forall x, y, z \text{ (son(x) = y and son (y) = z)}$$

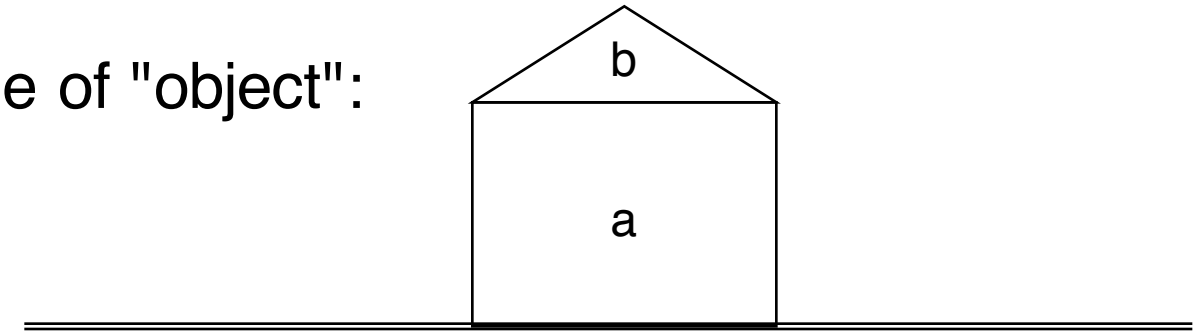
FIRST ORDER LANGUAGES

OBJECT DESCRIPTION LANGUAGE

- First-order logic languages.
  - Very rich
  - Objects are described as ground facts
  - Each attribute of a component becomes a predicate.
  - We can express also relations between components of the same object

Example: world block

•Example of "object":



Instance in attribute-value form: components:

- name: a, shape = square, size = large, on-table = yes
- name: b, shape = triangle, size = small, on-table = no

Instance in first-order logic:

- object (s, a), object (s, b),
- square (a), triangle (b), large(a), small(b), on-table (a)

- **on (b, a)** *Relation between parts*

FIRST ORDER LANGUAGES

CONCEPT DESCRIPTION LANGUAGE

- Concepts are described through logic clauses
- Compared to relational languages, they also allow:
  - to define the concepts recursively

```
ancestor(X, Y):- father(X, Z), ancestor(Z, Y)
```

- They are a deeply studied and formal language. Use of logic programming for the representation of objects and concepts: *inductive logic programming*

LEARNING TECHNIQUES

- Learning from value attribute objects:
  - Decision trees
  - Production rules

- Learning from first order logic facts:
  - Inductive Logic Programming

DECISION TREES LEARNING

- Systems that learn decision trees: CLS, IDR, C4, ASSISTANT, ID5, C4.5 etc.
- Appropriate problems:
  - instances are represented by attribute value pairs
  - the target function has discrete values
  - disjunctive descriptions of concepts may be required
  - the set of training data may contain errors
  - the set of training data may contain missing data

C4.5 ALGORITHM

- c4.5 [Qui93b, Qui96], designed by Quinlan is an algorithm for learning decision trees
  - Ranked #1 in the *Top 10 Algorithms in Data Mining* pre-eminent paper published by Springer LNCS 2008
- Evolution of ID3 by the same author
- Inspired by one of the first systems: CLS (Concept Learning Systems) by E.B. Hunt
- **J48** is an open source Java implementation implementation of the C4.5 algorithm in the Weka data mining tool.

TREE GENERATION ALGORITHM

- T: set of examples – **Training set**
- $\{C_1, C_2, ..., C_k\}$: Set of classes
- Consider the set T:
  - If T contains one or more examples, all belonging to the same class ➔ **single leaf** labeled with the class
  - T contains examples that belong to multiple classes ➔ **partition T** in subsets according to a test on an attribute. We have a node associated with the test, with a sub-tree for each possible test result. **Recursively call the algorithm** on each node created by the partition.
  - If T contains no example (empty set) ➔ **single leaf** labeled the class more' frequently in'both father

TERMINATION CONDITION

- In principle the algorithm stops when all leaves contain homogeneous examples.
- Indeed, we have less tight termination conditions
  - C4.5 does stops even if:
    - No test exists such that at least two subsets contain a minimum number of cases
    - By default the number of cases is 2

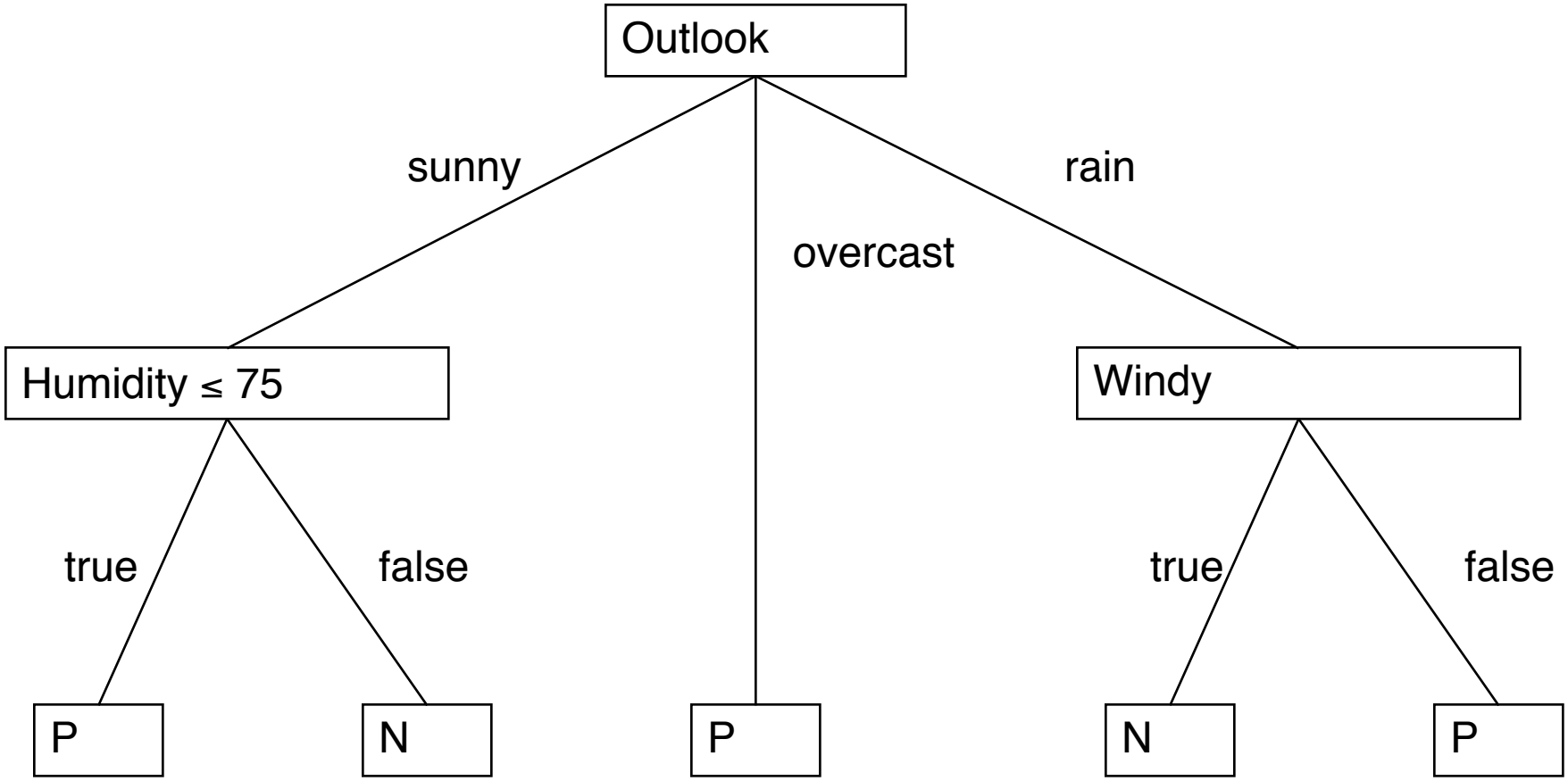
EXAMPLE

- Instances: Saturday morning
- Concepts: Saturday suitable for playing tennis and
  Saturday not suitable for playing tennis
- attributes:
  - outlook with values {sunny, overcast, rain}
  - temperature with numerical values
  - humidity with numerical values
  - windy with 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≤ 75: P

| humidity> 75: N

Outlook = overcast: P

Outlook = Rain

| windy = True: N

| windy = False: P

ATTRIBUTE CHOICE

- How to choose the test (attribute) at every step?
- Generation of the smallest possible trees: computationally intractable.
- It is then chosen based on a greedy heuristic (non-backtrackable)
- The choice is designed to **minimize the depth** of the final tree
  - The perfect attribute divides examples into sets that are all positive or all negative
  - A useless attribute leaves the example set with roughly the same proportion of positive and negative examples as the original set

ATTRIBUTE CHOICE

- We need a formal measure of the attribute "performance":
  - The measure should have its max with perfect attributed and min with useless ones.
  - Takes inspiration from information theory, namely the amount of information provided by an attribute.
  - The proportion of positive and negative example in a set T is a good estimation of this:
    - ***How much information we still need after the attribute test (the lower the better)***
    - ***Based on the concept of Entropy***

ENTROPY OF AN EXAMPLE SET

- Entropy of a set of examples: Given the set of samples S and the class C_j

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \times \log_2 \left(\frac{freq(C_j, S)}{|S|} \right)$$

- Entropy of the training set

info (T)

It can also be seen as a measure of non-uniformity of an arbitrary collection of examples: the lower the better

EXAMPLE

- Case of only two classes, p^+ and p^- are the proportion of positive and negative examples in the training set ($p^- = 1-p^+$).

- The entropy is given by:

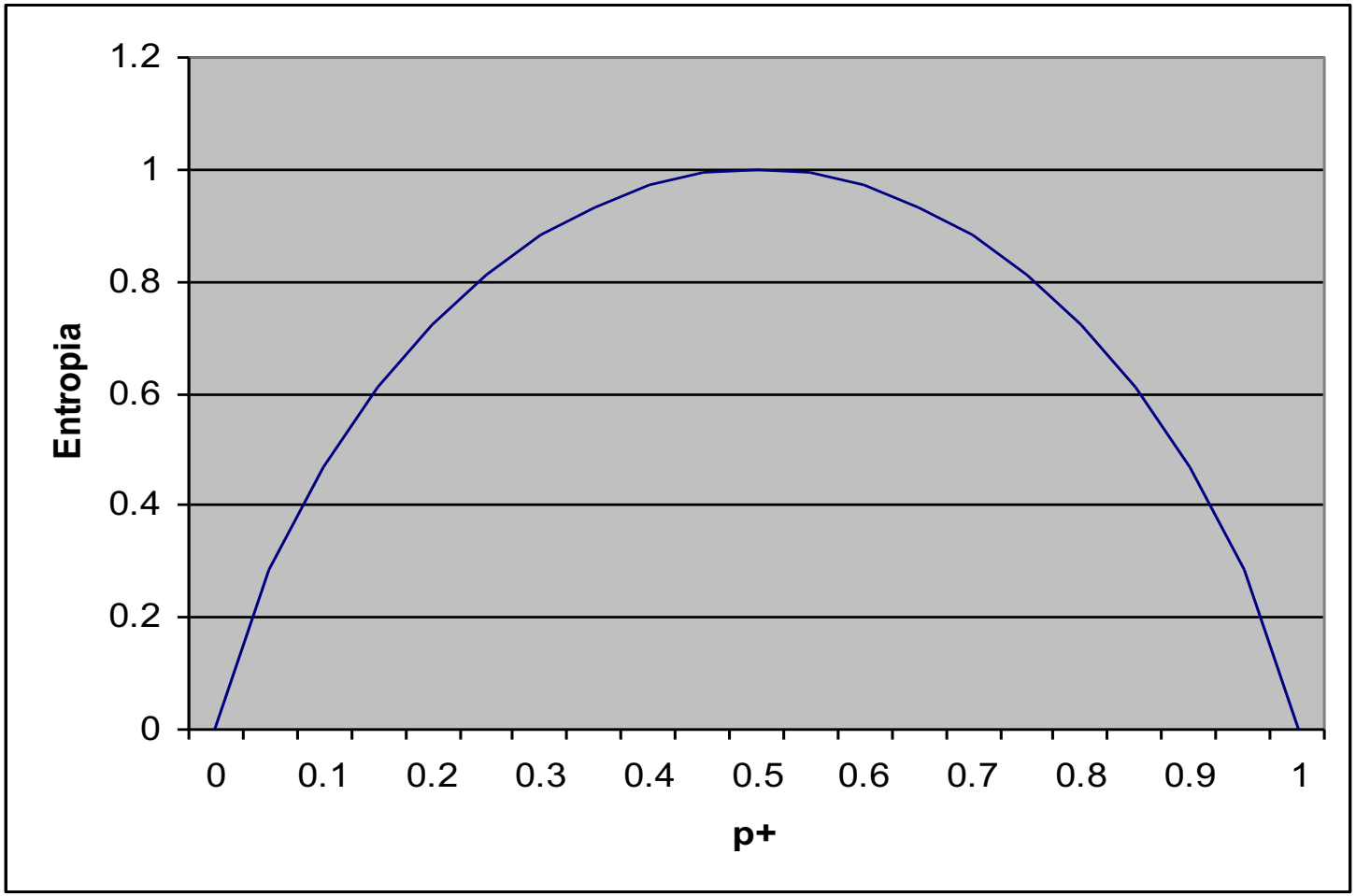
$$info(T) = -p^+ \times \log_2 p^+ - p^- \times \log_2 p^-$$

EXAMPLE

- The entropy is minimum when all the examples in T belong to the same class:
  - $p^+ = 0$ (or $p^+ = 1$) ➔ info (T) = 0
    - $0 * \log_2 0 = 0$

- The entropy is maximum when half of the examples belong to a class, and the other half to the other class:
  - $p^+ = 0.5$ ➔ info (T) = 1

ENTROPY



GAIN OF AN ATTRIBUTE CHOICE

- The gain of an attribute X is given by the difference between the original information and the one after the partition on X

- Entropy after partitioning (according to the X test): weighted average of the entropies of the individual sub-units

$$info_X(T) = -\sum_{j=1}^{n} \frac{|T_i|}{|T|} \times info(T_i)$$

- Information gain

$$gain\ (X) = info\ (T) - info_X(T)$$

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

EXAMPLE

- Consider the problem of the decisio: P or N
- In the original training set T we have 14 examples:
  - 9 positive
  - 5 negative

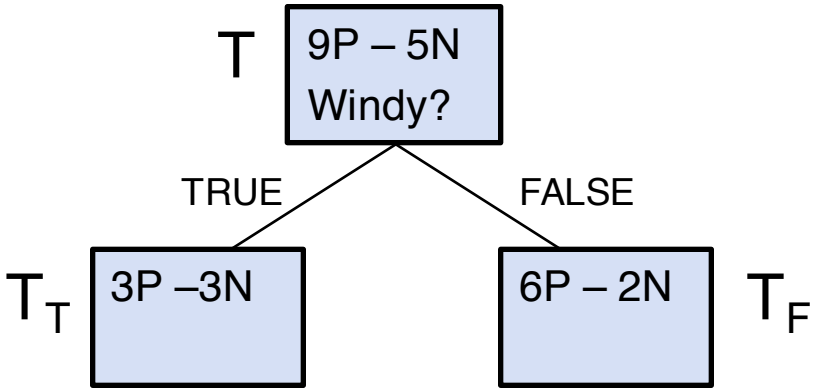
- Entropy of T

$$info(T) = -p^{+} \times \log_2 p^{+} - p^{-} \times \log_2 p^{-}$$

- $info\ (T) = -\ (9/14)\ log_2(9/14) - (5/14)\ log_2(5/14) = 0.940$

EXAMPLE

- Test on the attribute wind:



$info\ (T_F) = -\ 6/8 * log_2(6/8) - 2/8 * log_2(2/8) = 0.811$

$info\ (T_T) = -\ 3/6 * log_2(3/6) - 3/6 * log_2(3/6) = 1$

Should be weigthed summed

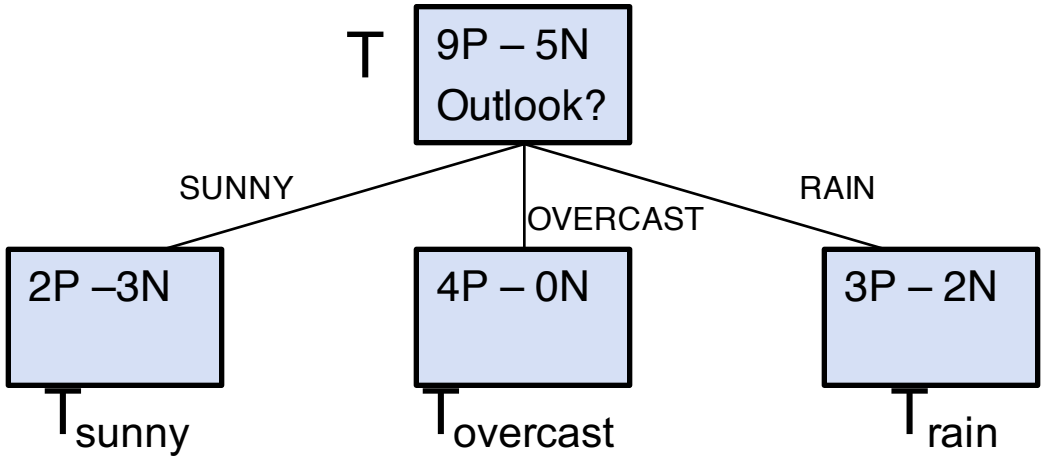
$info_{wind}(T) = 8/14 * info\ (T_F) + 6/14 * info\ (T_T)$

$gain(wind) = info\ (T) - info_{wind}(T) =$

$= 0.940 - (8/14) * 0.811 - (6/14) * 1 = 0,048$

EXAMPLE

- Test on the attribute outlook



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

info $(T_{overcast}) = 0$

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

$info_{outlook}(T) = 5/14 * info(T_{sunny}) + 4/14 * info(T_{overcast}) + 5/14 * info(T_{rain})$

gain (outlook) = info(T) – $info_{outlook}(T) = 0.940-(0.357* 0.971+ 0.286 * 0 +$ *0357*0971)= 0.246*

TEST GENERATION

- **Problem:** generation of the set of possible tests.
- A priori definition of the allowed forms
- Three possible types of tests
  - Discrete attribute: a result for each value.
  - Discrete attribute: a result for each group of values.
  - Continuous attribute: two possible outcomes (binary test).
- Additional constraint: at least two of $T_1, T_2, \ldots, T_n$ must contain a minimum number of examples.

TESTS ON DISCRETE ATTRIBUTES

When testing on discrete attributes, we can have:

- A result for each value found in the training set
- A result for each group of values: we must determine how many and which groups to consider.
  - Domain knowledge: a priori identify significant groups (new values).
  - Testing of all possible partitions of the set of values.

TEST ON CONTINUOUS ATTRIBUTES

- A continuous attribute A takes m distinct values in T $\{V_1, V_2...., V_m\}$ ordered from the smallest to the largest
- Select as threshold $Z = V_k$ that splits the m values into two groups:
  - $A \leq Z \{V_1, ..., V_k\}$
  - $A > Z \{V_{i+1}, ..., V_m\}$
- We have m-1 candidates: $V_1, ..., V_{m-1}$
- Evaluation of each of the m-1 candidates in terms of entropy.

ATTRIBUTES WITH MISSING VALUES

- In many domains not all attribute values are known for every example. How to compute the gain?
- Gain: we call $F \subseteq T$ the subset of T for which A is known and X is a test on A

$$info(T) = info(F)$$

$$info_X(T) = info_X(F)$$

- The test of A for examples in T \ F does not provide any information on the class.

Gain (X) = (Probability that A is known) x (info (T) -info$_X$(T))
+ (Probability that A is unknown) x 0 =

$$= \frac{|F|}{|T|} \times (info(F) - info_X(F))$$

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	**?**	**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

EXAMPLE

- In the new training set T, we want to test the partition on the attribute Outlook.
  - We have that Outlook is known in 13 cases out of 14.
- Considering the 13 cases for which Outlook is know you have the following frequencies

Outlook	P	N	Total
Sunny	2	3	5
Overcast	3	0	3
Rain	3	2	5
Total	8	5	13

EXAMPLE

- The entropy of the set F (composed by 13 examples where outlook is known) is

  $info\ (F) = -\ 8/13 * log_2(8/13)\ -5\ /\ 13 * log_2(5/13) = 0.961$

- The information on the test on Outlook is a weighted sum

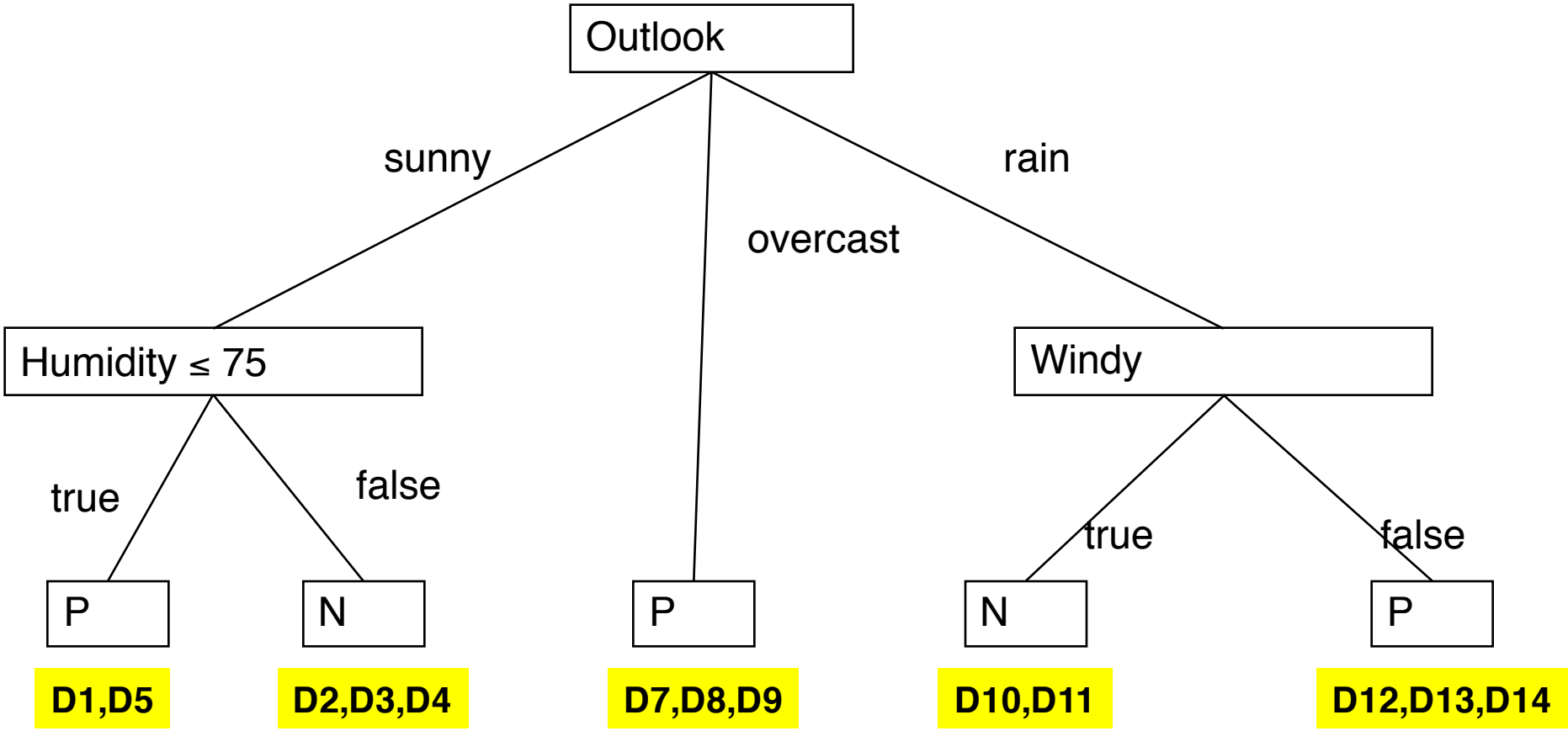
$$info_{Outlook}(T) = 5/13 * (-\ 2/5 * log_2(2/5)\ -3\ /\ 5 * log_2(3/5))$$
$$+\ 3/13 * (-\ 3/3 * log_2(3/3)\ -0\ /\ 3 * log_2(0/3))$$
$$+\ 5/13 * (-\ 3/5 * log2\ (3/5)\ -2/5 * log2\ (2/5))$$
$$= 0.747$$

$$gain\ (Outlook) = 13/14 * (0961\text{-}0747) = 0.199$$

  – A bit lower then before

PARTITIONING

- Completely known examples fall entirely in one leaf
- What happens to those examples that have a unknown value for an attribute used as a test? **D6?**



PARTITIONING

- Probabilistically generalized: each instance is associated with a weight w initially set to 1
- Consider a test on A
  - if an example e, with weight w in T, has A = V_i (known) assign it to T_i with weight w and to T_j ($j \neq i$) with weight 0
  - if an example e, with weight w in T, has A unknown we partition it in each T_j with a weight of $w*P_{vj}$ where P_{vj} is the probability of the value V_j in T
  - P_{vj} can be estimated as the sum of the weights of the instances in T that have A = V_j divided by the sum of all the weights of the instances in T that have A known

EXAMPLE

- Partitioning according to the attribute Outlook
- The 13 cases for which Outlook is known are easy
- The remaining case is assigned to all blocks, corresponding to the values sunny, overcast and rain, with weights 5/13, 3/13 and 5/13 respectively

Example

- Consider the subset corresponding to outlook = sunny

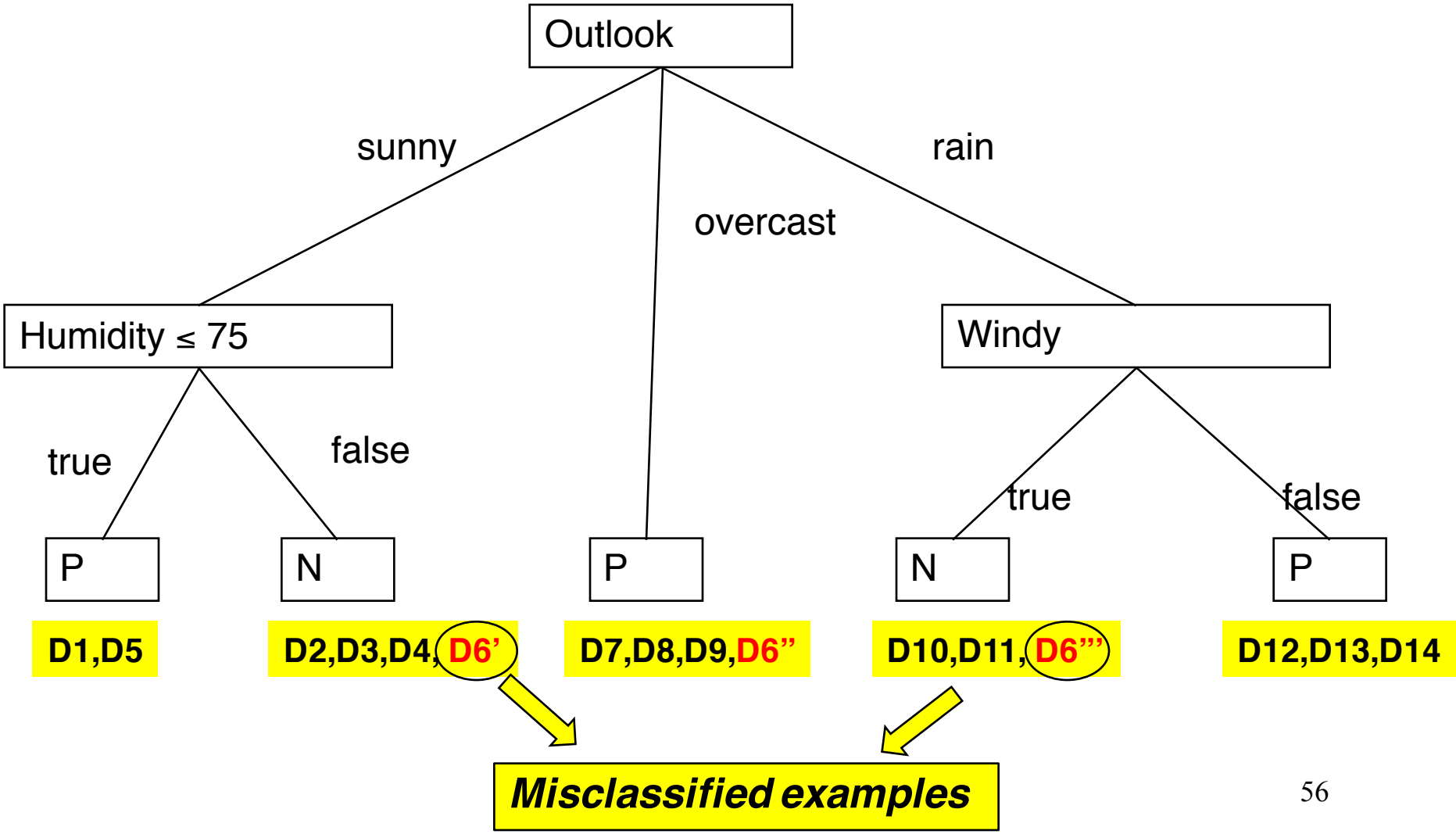
No	Outlook	Temp	Humidity	Windy	Classe	Weight
D1	sunny	75	70	T	P	1
D2	sunny	80	90	T	N	1
D3	sunny	85	85	F	N	1
D4	sunny	72	95	F	N	1
D5	sunny	69	70	F	P	1
D6'	**sunny**	**72**	**90**	**T**	**P**	**5/13**

- If this set is further partitioned on the same tests on humidity, the distribution of classes in subsets are
  - *Humidity ≤ 75* *2 class P, 0 class N*
  - *Humidity> 75* *5/13 class P, 3 class N*

TRAINING SET

No	Outlook	Temp (° F)	Humid (%)	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'	**sunny**	**72**	**90**	**T**	**P**	**5/13**
D6"	**overcast**	**72**	**90**	**T**	**P**	**3/13**
D6"'	**rain**	**72**	**90**	**T**	**P**	**5/13**
D7	overcast	83	78	F	P	1
D8	overcast	64	65	T	P	1
D9	overcast	81	75	F	P	1
D10	rain	71	80	T	N	1
D11	rain	65	70	T	N	1
D12	rain	75	80	F	P	1
D13	rain	68	80	F	P	1
D14	rain	70	96	F	P	1

COMPLETE PARTITIONING



EXAMPLE

- We have no longer homogeneous classes so why not partition again?
  - No test produces two subsets with at least two examples of each

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)**

| windy = False: P **(3.0)**

OUTPUT INTERPRETATION

Numbers (A, B) associated with each leaf

- A = total number of training examples sets associated to the leaf
- B = number of misclassified examples of the training sets associated to the leaf

N (3.4 / 0.4)

- It means that 3.4 cases belong to the leaf of which 0.4 does not belong to the class N

CLASSIFICATION OF NEW CASES

- The main purpose of any ML model is to be able to classify unseen examples.
- Easy for examples with all attributes known.
- Classification of an example e with A unknown:
  - initially assign e weight w=1
  - If a node with a test on A is found, we partition it in all possible sub-branches. In each sub-branch for V_j assigning e the weight P_{vj}
  - We eventually reach more leaves:
    - we obtain a distribution of classes instead of a single class. The probability of each class is the weight of the examples that reached the leaf
    - if two leaves are associated to the same class C, the probability of C is given by the sum of the probabilities of the example weights in the two leaves

Example

- You want the classify the example
  - Outlook = sunny, temperature =70, Humidity = ?, Windy = F
  - Outlook = sunny => first subtree
  - Humidity =? It does not allow us to determine whether humidity ≤ 75

- It descends along the two branches by assigning them two fractional weights:
  - branch humidity ≤ 75 with weight 2.0 / 5.4 = 0.370
  - branch humidity > 75 with weight 3.4 / 5.4 = 0630

Example

- The example part assigned to the branch humidity ≤ 75 is classified as P with probability 100%
- The example part assigned to the branch humidity > 75 is classified as
  - N with probability 3 / 3.4 = 88%
  - P with probability 0.4 / 3.4 = 12%
- Overall, the relative distribution of all classes is
  - P: 0.370 * 0.630 * 100% + 12% = 44%
  - N: 0630 * 88% = 56%

EVALUATION OF A DECISION TREE

- A learning algorithm is good if the produced hypothesys are good at predicting the classification of unseen cases.
- We divide the example set into two disjoint sets: the training set and the test set
- Design the decision tree on the training set
- Measure the percentage of examples in the test set that are correctly classifies
- Repeat the above steps with different sizes of training sets that are randomly selected

NOISE AND OVERFITTING

- In **overfitting**, a learnt model describes random error or noise instead of the underlying relationship.
- Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- A model that has been overfit has poor predictive performance.
- It overreacts to minor fluctuations in the training data.
- Two ways to prevent overfitting:
  - **Tree pruning**: a posteriori removes splits on irrelevant attributes
  - **Cross-validation**: repeatedly split the known data in training and test set with the results averaged.

REMARKS

- C4.5 performs a hill-climbing search in the space of all possible decision trees

- Remarks:
  - The space of possible decision trees is equivalent to powersets of all possible examples
  - Too large
  - C4.5 only keeps a single case when searching. It performs a committment each time it selects a test attribute.

REMARKS

- C4.5 does not perform backtracking. Therefore it runs the risk of incurring an locally optimal solutions
- C4.5 uses all training examples at every step to decide how to refine the tree.
- On the contrary, other methods make decisions incrementally based on individual examples.
- The result is that the search of C4.5 is less sensitive to errors in the individual examples.

- Finally, one important feature of C4.5 is that it selects informative features out of the training set.

DECISION TREES VARIANTS

- Based on outcome
  - Classification trees: predicted outcome is a class
  - Regression trees: predicted outcome is a real number
  - Some similarities and differences (ex. procedure to determine where to split)
- Ensamble methods: construct more than one tree
  - Bagging trees: builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction
  - Random Forests: uses a number of decision trees in order to improve the classification rate.
  - Boosted trees: used for classification and regression

Bibliography

[Mit97] TM Mitchell, *Machine Learning*, McGraw-Hill, 1997

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

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

[Wit99] IH Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.