Introduction to Decision Trees Learning
[IIA – Lect._____]

Alessio Micheli

micheli@di.unipi.it

Dipartimento di Informatica
Università di Pisa - Italy

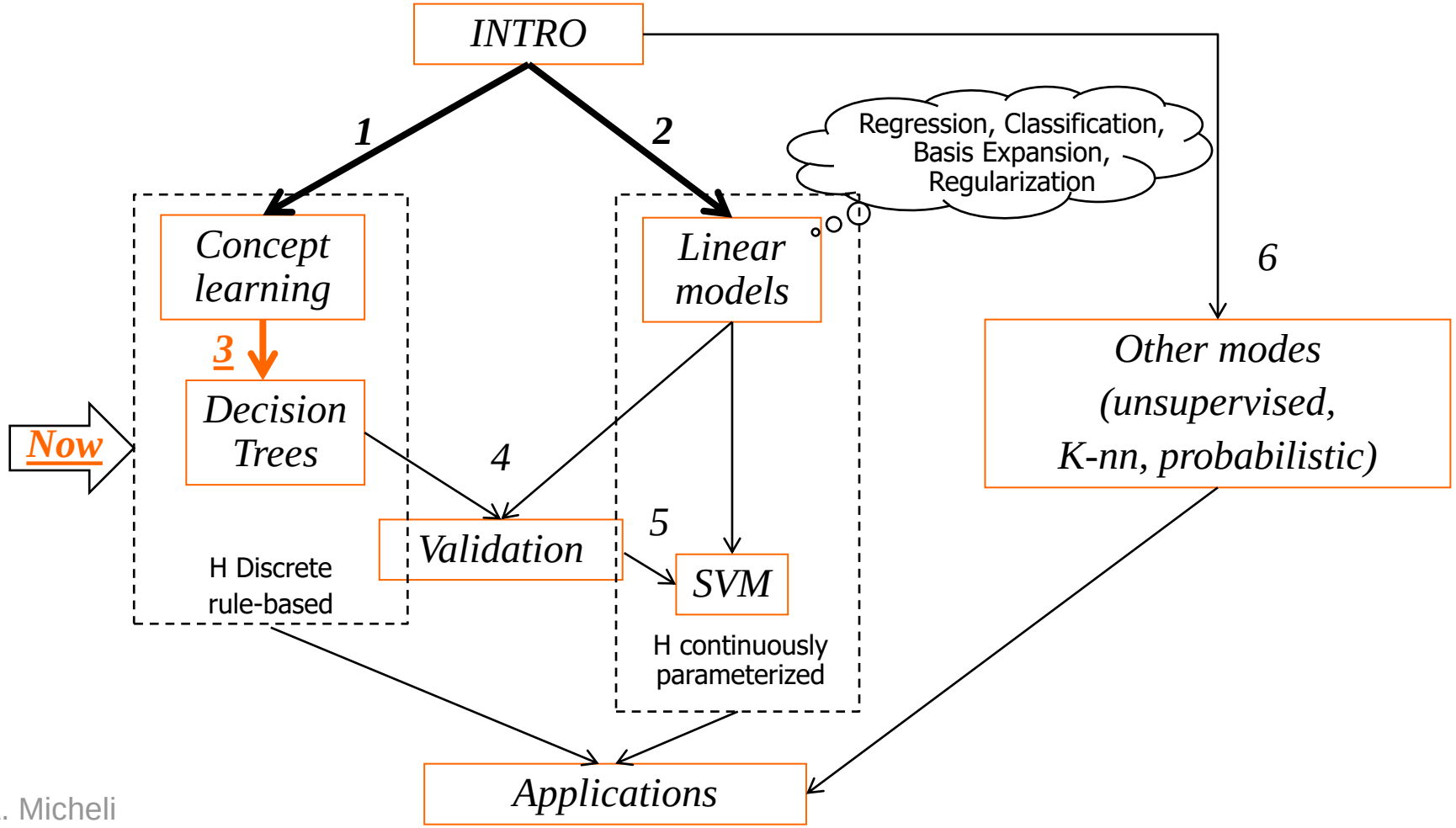
Computational Intelligence & Machine Learning Group

DRAFT, please do not circulate! *2023*

In the course flow

Dip. Informatica
University of Pisa

In the course: example for *numerical eq*. language (H continuous space), now we return to *symbolic rules* language (by *DT*) [see 2 parallel tracks]



Coming from the Concept Learning lecture

Dip. Informatica
University of Pisa

Overcome the *conjunctive restriction* toward *flexible* models:

Many approaches in ML, e.g.:

[in the class of symbolic - rule based approaches]

- **Decision trees** (next slides) (chap. 3 Mitchell book)
  (a popular approach in ML/DM)

- Genetic Algorithms:
  - encode each rule set as a bit string and uses genetic search operator to explore this H
- Inductive Logic Programming (cap 10 Mitchell):
  - First-order logic rules that contain variables (much more expressive than propositional rules)
  - Automatic inferring *Prolog* programs from examples (collection of Horn clauses)

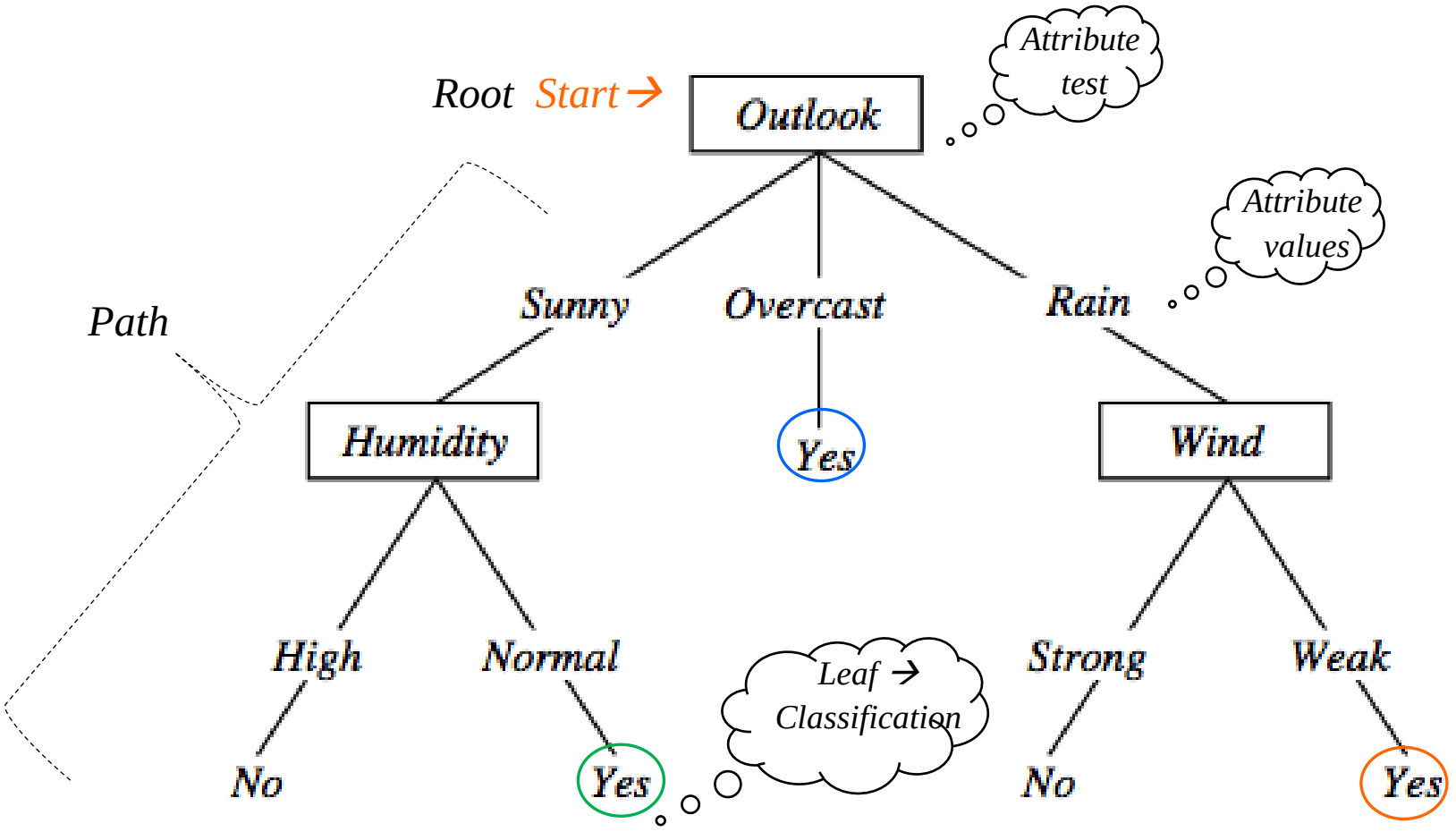
Play Tennis
Training set

Dip. Informatica
University of Pisa

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

Decision trees (e.g. play tennis)

Dip. Informatica
University of Pisa



***Exercise**: classify the instance*

<Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong>

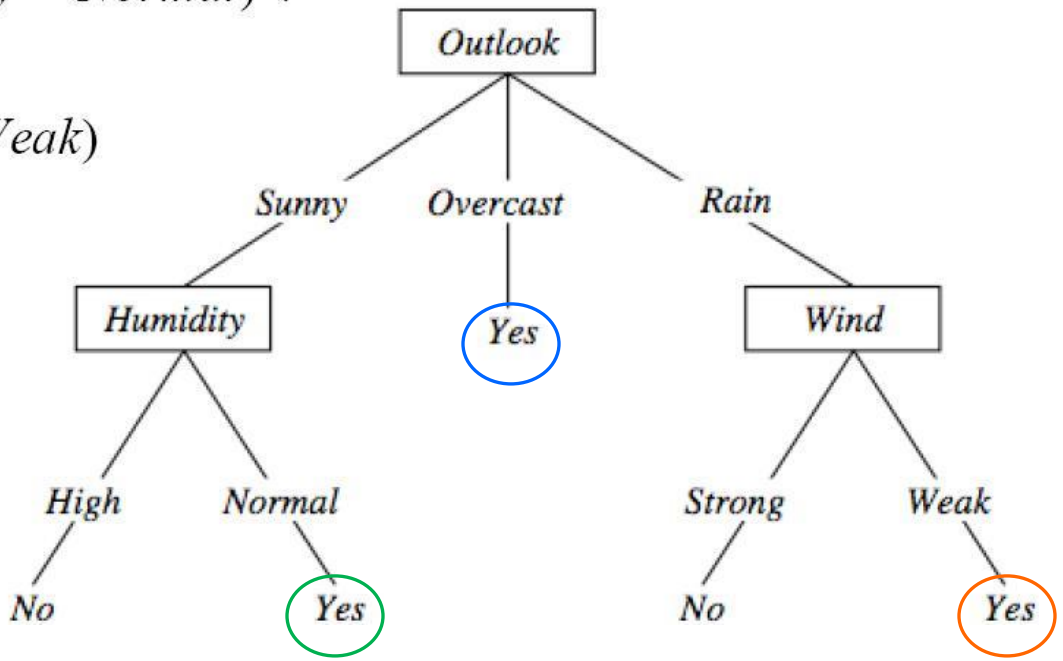
Decision trees: expressive power

Dip. Informatica
University of Pisa

- Decision trees represent a disjunction of conjunctions of constraints on the value of attributes:

1. $(Outlook = Sunny \wedge Humidity = Normal) \vee$
2. $(Outlook = Overcast) \vee$
3. $(Outlook = Rain \wedge Wind = Weak)$

or *if-then-else* rules



H of D.T. capable of expressing any finite discrete-valued function (propositional)

Top-down induction of Decision Trees

Dip. Informatica
University of Pisa

- **ID3** (Quinlan, 1986) is a basic algorithm for **learning** DT's
- Given a training set of examples, the algorithms for building DT performs *search* in the space of decision trees
- The construction of the tree is *top-down*. The algorithm performs a *greedy* search.
- The fundamental question is "*which attribute should be tested next*"? Which question gives us more *information gain*?"
  - Select the *best* attribute
  - A descendent node is then created for each possible value of this attribute and examples are partitioned according to this value
  - The process is repeated for each successor node until all the examples are classified correctly or there are no attributes left

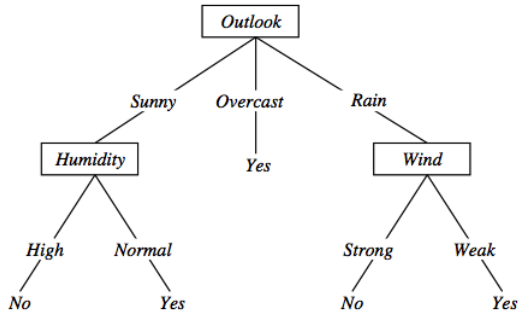
ID3: algorithm

Dip. Informatica
University of Pisa

ID3(*X*, *T*, *Attrs*) *X*: training examples

T: target attribute (e.g. *PlayTennis*),

Attrs: other attributes, initially all attributes



Create *Root* node

If all X's are +, *return* *Root* with class +

If all X's are –, *return* *Root* with class –

If *Attrs* is empty *return* *Root* with class most common value of *T* in *X*

else

$A \leftarrow$ best attribute; decision attribute for Root $\leftarrow A$

For each possible value v_i of *A*:

- add a new branch below *Root*, for test $A = v_i$
- $X_i \leftarrow$ *subset* of *X* with $A = v_i$
- *If* X_i is empty *then* add a new leaf with class the most common value of *T* in *X*

  else add the subtree generated by **ID3**(X_i, *T*, *Attrs* – {*A*})

return *Root*

Note: restricted to subset of examples X_i *and to other attributes*

Selecting the best attribute: entropy

Dip. Informatica
University of Pisa

- We use the notion of *entropy*, commonly used in information theory.
- Entropy measures the *impurity* of a collection of examples.
  It depends on the distribution of the random variable *p.*
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

Def $\quad \textbf{Entropy}\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [\text{assume } 0 \log_2 0 = 0]$

$Entropy\ ([14+, 0–]) = -14/14\ log_2\ (14/14) - 0\ log_2\ (0) = \mathbf{0}$

$Entropy\ ([9+, 5–]) = -9/14\ log_2\ (9/14) - 5/14\ log_2\ (5/14) = \mathbf{0{,}94}$

$Entropy\ ([7+, 7–]) = -7/14\ log_2\ (7/14) - 7/14\ log_2\ (7/14) =$

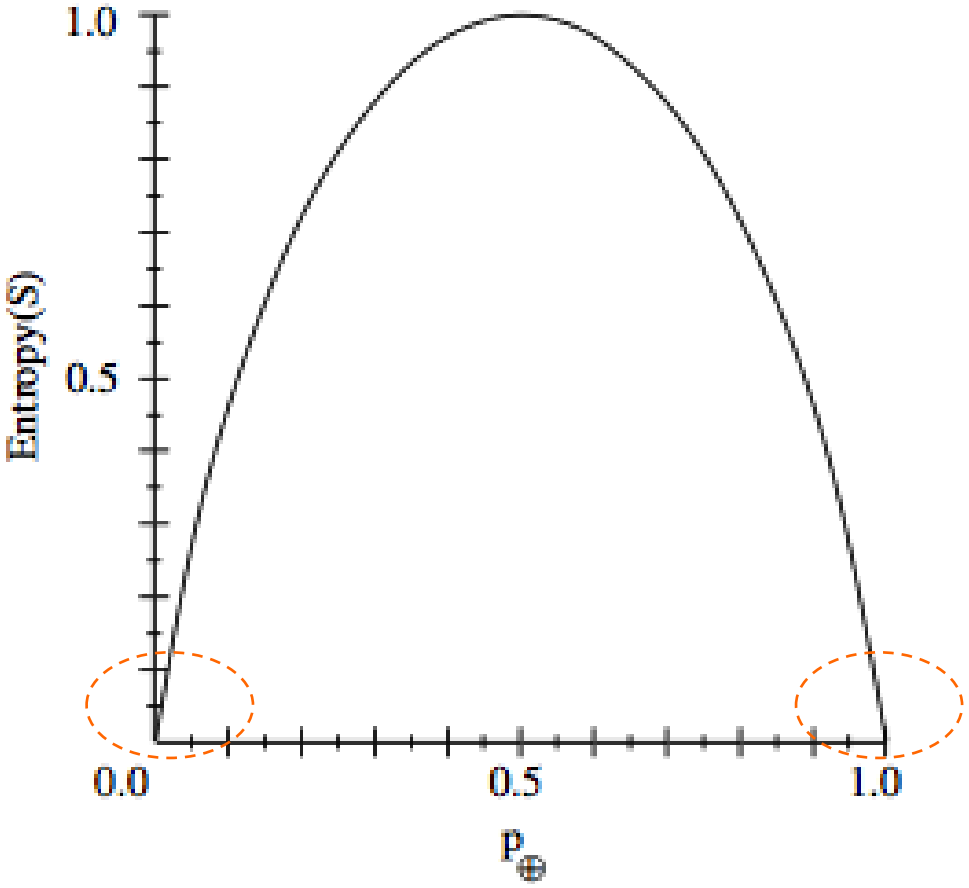
$= 1/2 + 1/2 = \mathbf{1} \qquad [log_2 1/2 = -1]$

High impurity
(low homogeneity)
No good ;-)

Note: $0 \le p \le 1,\ 0 \le entropy \le 1$

Entropy (of S for the Target)

Dip. Informatica
University of Pisa



Good for us

All +1 or All -1 → 0 entropy, "no information" . Max for S with ½ + and ½ - examples.

Selecting the best attribute: information gain definition

Dip. Informatica
University of Pisa

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- Expected *reduction* in entropy knowing *A*

Def

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

$Values(A)$ possible values for A

Sv subset of examples of S for which A has value v *(weighted sum)*

- The higher the information gain the more effective the attribute in classifying training data (higher variation in the homogeneity of the class distribution over the sub sets, max separation of classes).
- ***Homogeneous, as*** [14+, 0–] or [0+,7-] allow us "clear" classification

Selecting the best attribute: information gain: why?

Dip. Informatica
University of Pisa

- Entropy to measure the homogeneity (indeed impurity) of the class of the subset of examples, hence:
- **Select A that maximize:**

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

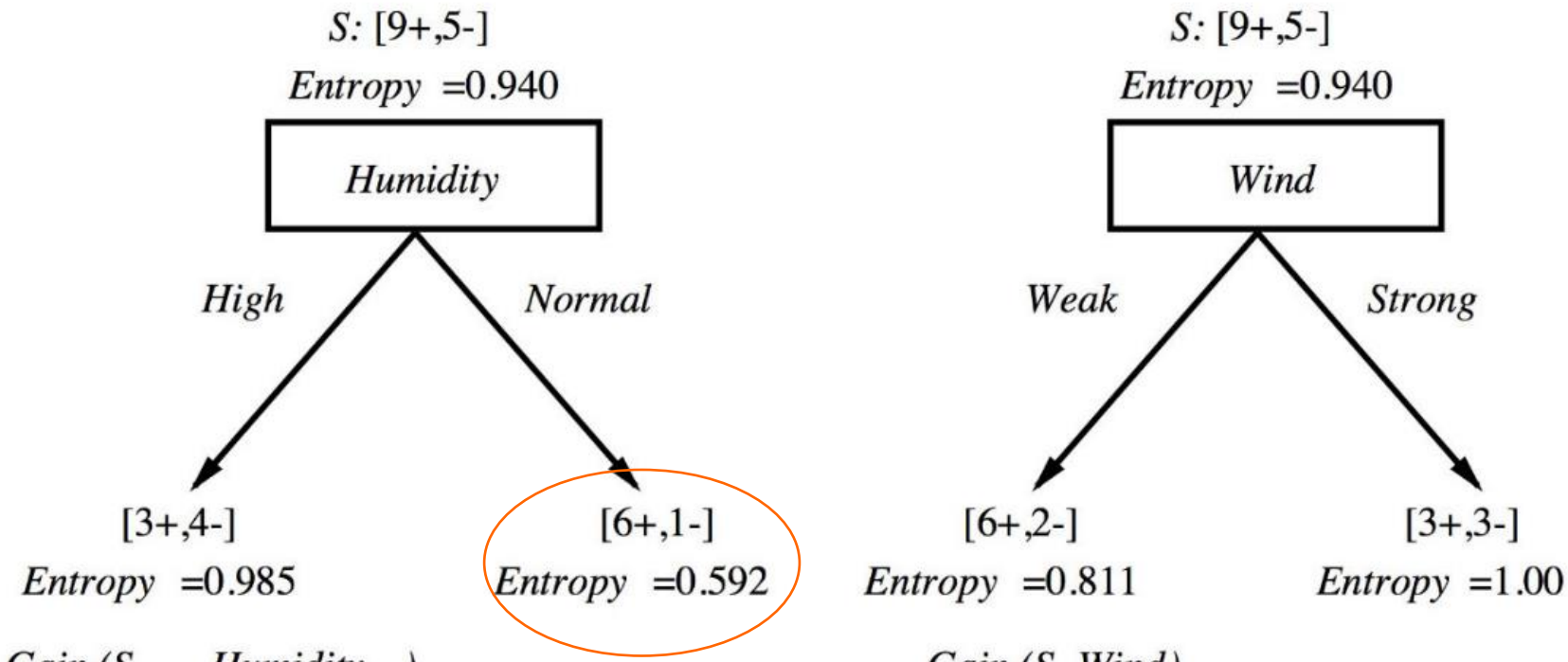
After splitting, lower values (more homogeneous targets) in each subsets → higher Gain

- The aim is to separate the examples on the basis of the target, finding the attribute that discriminates examples that belong to different target classes
  - *e.g. all the – on the left, all the + on the right*

Example

Dip. Informatica
University of Pisa

Select the best attribute



Favorable case

Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

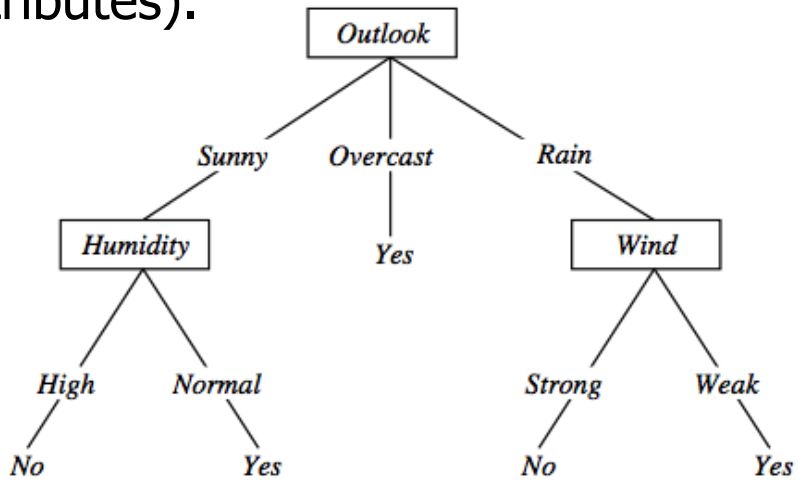
THE WINNER

Exercise: compute the values

Going on

Dip. Informatica
University of Pisa

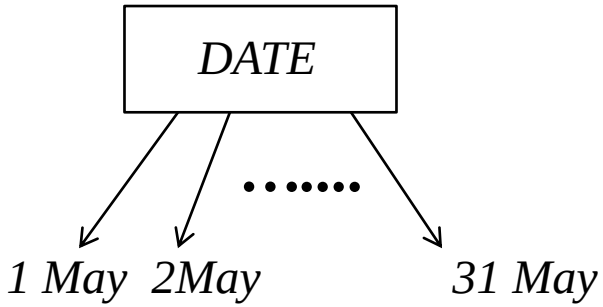
- Exercise: (example in Mitchell book, sect. 3.4.2)
  - The information gain is computed for the all the attributes and the one with highest inf. gain is selected as the first decision node (e.g. *outlook*)
  - The training data are partioned for values of *outlook* (and associated to following nodes)
  - If entropy is not zero in a set, the tree continues to grow there (with such data and remaining attributes).
  - Obtaining the flowing DT:



Problems with *information gain*

Dip. Informatica
University of Pisa

- The information gain favors attributes with many possible values.
- Consider the attribute *Date* in the *PlayTennis* example.
  - "Date" has the maximum information gain: Every day correspond to a different subset which is pure: [1+, 0-] or [0+, 1-] → 0 entropy
  - Perfect fit (separation) of TR data
  - But it is not significant: not useful for unseen instances!!!



- How to avoid generation of many small subsets?

An alternative measure: *gain ratio*

Dip. Informatica
University of Pisa

Def

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

Where

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

- S_i are the sets obtained by partitioning on value v_i of A, up to c values
- *SplitInformation* measures the entropy of S with respect to the values of A. The more uniformly dispersed the data the higher it is.

- *GainRatio* penalizes attributes that split examples in many small classes such as *Date*. Let $|S|=n$, *Date* splits examples in n subsets.
  - $SplitInformation(S, Date) = -[(1/n \log_2 1/n)+\ldots+ (1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$ *which is* > 1 *for* $n>2$ → *we reduce the GainRatio*
- Compare with an A which splits data in two even classes (with $(n/2)/n = ½$):
  - $SplitInformation(S, A) = -[(½ \log_2 ½)+ (½ \log_2 ½)] = -[-½ -½] = 1$ → *no reduct.*

Adjusting *gain-ratio*

Dip. Informatica
University of Pisa

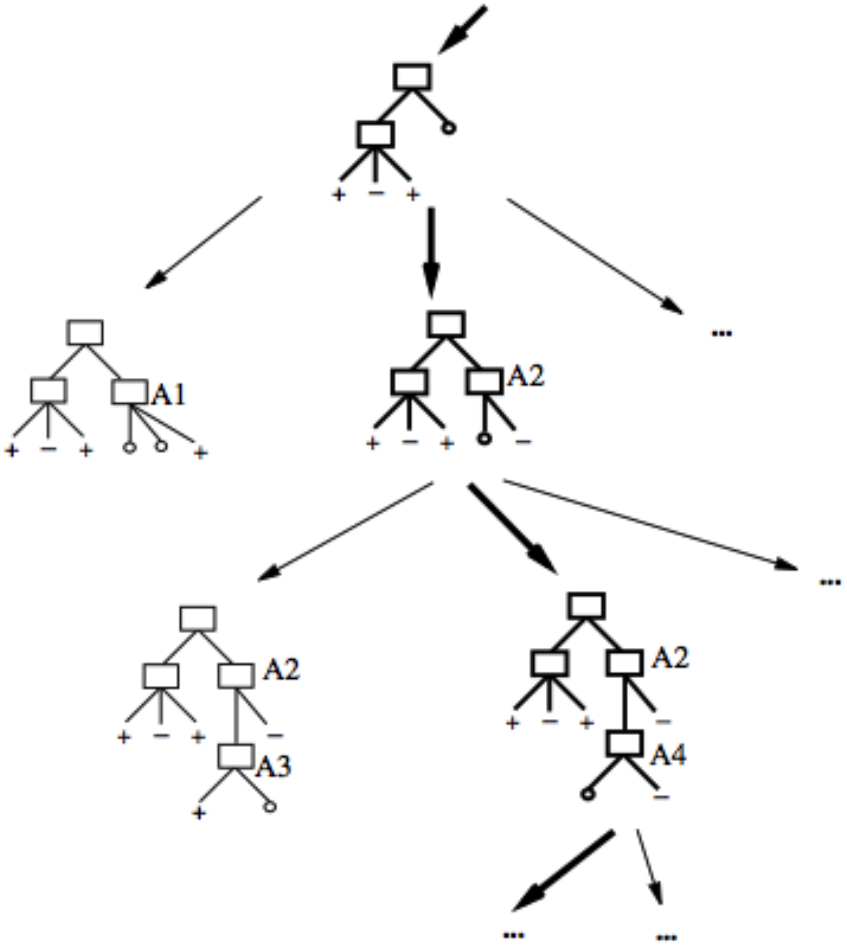
- Problem: $SplitInformation(S, A)$ can be zero or very small when $|S_i| \approx |S|$ for some value i *[log 1 =0]*
  - *E.g. An extreme case with* $|S_1|=0$ $|S_2|=n$
  - *SplitInformation = - [0/n * log 0/n + n/n * log n/n] = - 0 - 0 = 0*
  - *E.g. an attribute with the same value for the examples, not good indeed (not informative)!*

- To mitigate this effect, the following heuristics has been used:
  1. compute *Gain* for each attribute
  2. apply *GainRatio* only to attributes with *Gain* above average

- Other measures have been proposed, …see Mitchel book

Hp Space Search in DT learning

Dip. Informatica
University of Pisa



Hp space search by ID3,

Search (*hill-climbing*) through the space of possible DT from simplest to increasingly complex

W.r.t. to previous alg. (Cand. Elimin.):

- The hypotheses space is complete (represents all discrete-valued functions)
- The search maintains a single current hypothesis
- No backtracking; no guarantee of optimality (local optima)
- It uses all the available examples (not incremental)
- May terminate earlier, accepting noisy classes

Inductive bias in DT learning

Dip. Informatica
University of Pisa

- What is the **inductive bias** of DT learning?

1. *Shorter trees are preferred over longer trees*

   Due to simple-to-complex search, incremental.

   Not enough. This is the bias exhibited by a simple breadth first algorithm generating all DT's and selecting the shorter consistent one

2. *Prefer trees that place high information gain attributes close to the root*

- *Note*: DT's are not limited in representing all possible functions.
  - The restriction is *not* over the Hp space, it is on the *search strategy*.

Two kinds of Biases

Dip. Informatica
University of Pisa

Def

- Preference or **search biases** (due to the search strategy)
  - ID3 searches a *complete* hypotheses space; the search strategy is *incomplete*

Def

- Restriction or **language biases** (due to the set of hypotheses expressible or considered)
  - *Candidate-Elimination* searches an *incomplete* hypotheses space; the search strategy is *complete* (all the consistent hp, VS)

- A combination of biases in learning a linear models (Why? Exercise).

Why the search bias can be preferred over the language bias?

- In ML typically uses ***flexible*** *approaches* (universal capability of the models), without excluding a priori the unknown target function
- Of course, *flexible* → take care of the overfitting issue !

Prefer shorter hypotheses: Occam's razor

Dip. Informatica
University of Pisa

- Why prefer shorter hypotheses?
- Again …. The **Occam** (Ockham) razor (1300!)…
  "The simplest explanation is more likely the correct one" Or *"prefer the simplest hypothesis that fits the data"*
  - Lex parsimoniae ("law of parsimony", "law of economy", or "law of succinctness")
  - The term razor refers to the act of *shaving away* unnecessary assumptions to get to the simplest explanation.

- Remember the "*control of model complexity*" by *regularization* for linear models
- Again …This fundamental concept in ML will be found again and we will "rationalize" it at the end (to quantify it see #ML)

Issues in decision trees learning

Dip. Informatica
University of Pisa

- Overfitting *
  - Early stopping
  - Reduced error pruning
  - Rule post-pruning

** A general concept (hence more relevant)*

- Extensions (specific for DT)
  - Alternative measures for selecting attributes [done]
  - Continuous valued attributes
  - Handling training examples with missing attribute values
  - Handling attributes with different costs
  - Improving computational efficiency

Most of these improvements in C4.5 (Quinlan, 1993)

Overfitting: definition

Dip. Informatica
University of Pisa

- Building trees that "adapt too much" to the training examples may lead to "overfitting".
- Consider error of hypothesis h over
  - training data (also TR): $error_D(h)$ empirical error
  - entire distribution X of data: $error_X(h)$ expected or true error

Def

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

$$error_D(h) < error_D(h') \quad \text{and}$$

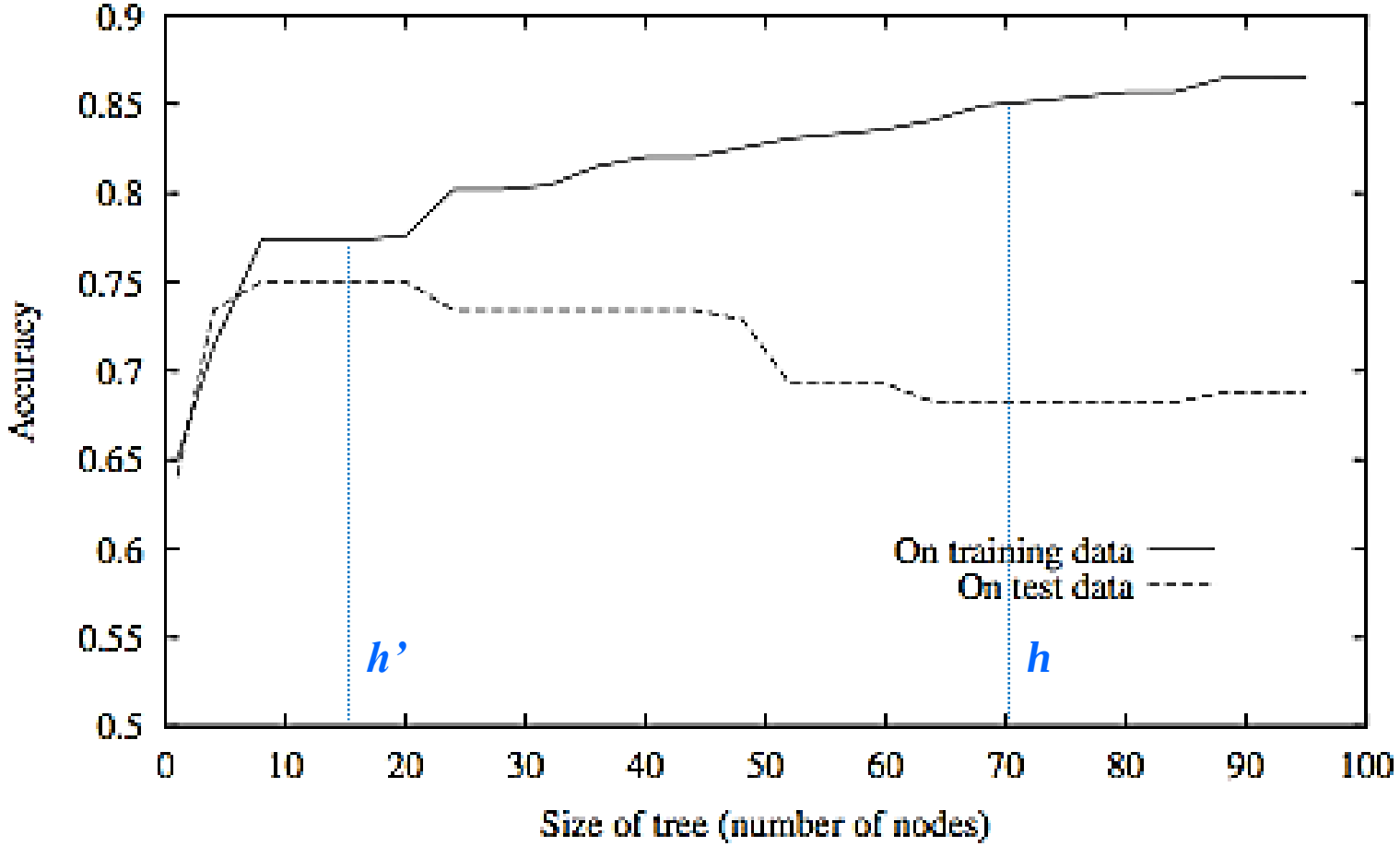
$$error_X(h') < error_X(h)$$

i.e. h' behaves worse on TR data, better over unseen data

Flexible approaches can easily encounter overfitting (if used without special care)

Overfitting in decision tree learning

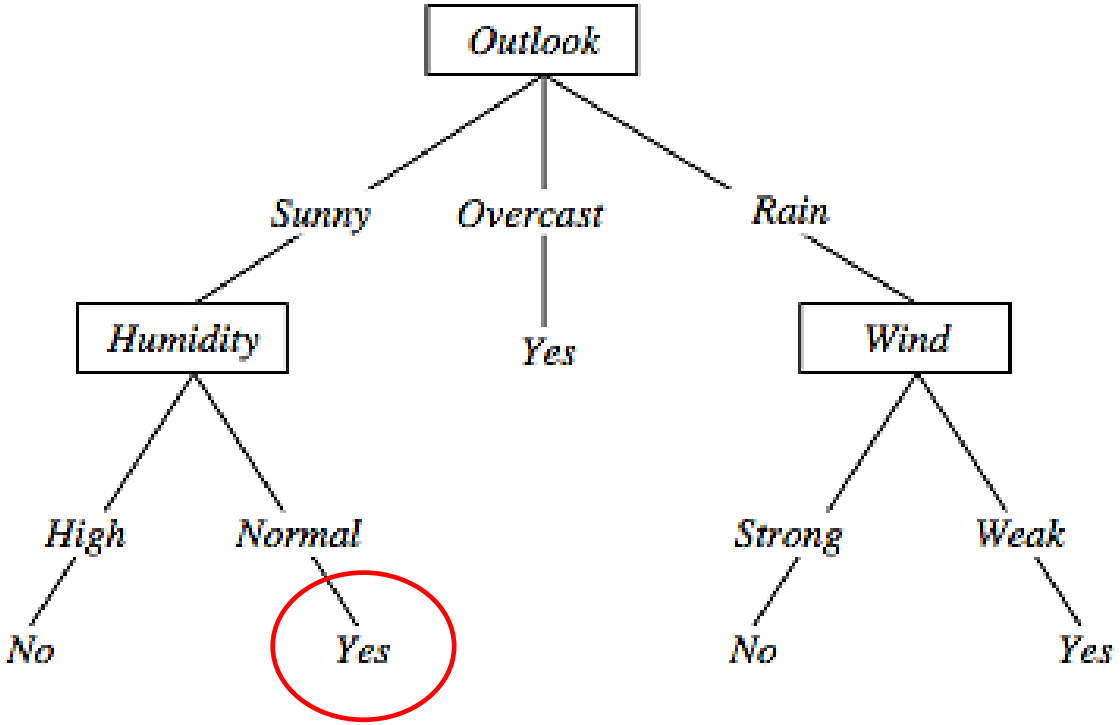
Dip. Informatica
University of Pisa



i.e. model complexity

An example with noisy data

Dip. Informatica
University of Pisa



⟨*Outlook=Sunny, Temp=Hot, Humidity=Normal, Wind=Strong, PlayTennis=No* ⟩

Immage this new noisy example causes splitting of 2nd leaf node
→ DT grows (its complexity grows)

Fitting even the noise, the new DT is perfect on TR data, not for new data

"Avoiding"* overfitting in DT

Dip. Informatica
University of Pisa

Two strategies:

1. Stop growing the tree earlier, before perfect classification ("*early stopping*", see the plot before)
2. Allow the tree to *overfit* the data, and then *post-prune* the tree

How to assess the effect?

- Training and validation set
  - split the training set in two parts (*training* and *validation* sets) and use the validation set to assess the utility of 1 and 2
- Other approaches
  - Use a statistical test to estimate effect of expanding or pruning
  - *Minimum description length principle*: uses a measure of complexity of encoding the DT and (misclassified) examples, and halt growing the tree when this encoding size is minimal

* "Avoiding" is not correct, we can see the methods to mitigate the overfitting, but we have still to control it by a conscious management (see the next lecture)

Reduced-error pruning (Quinlan 1987)

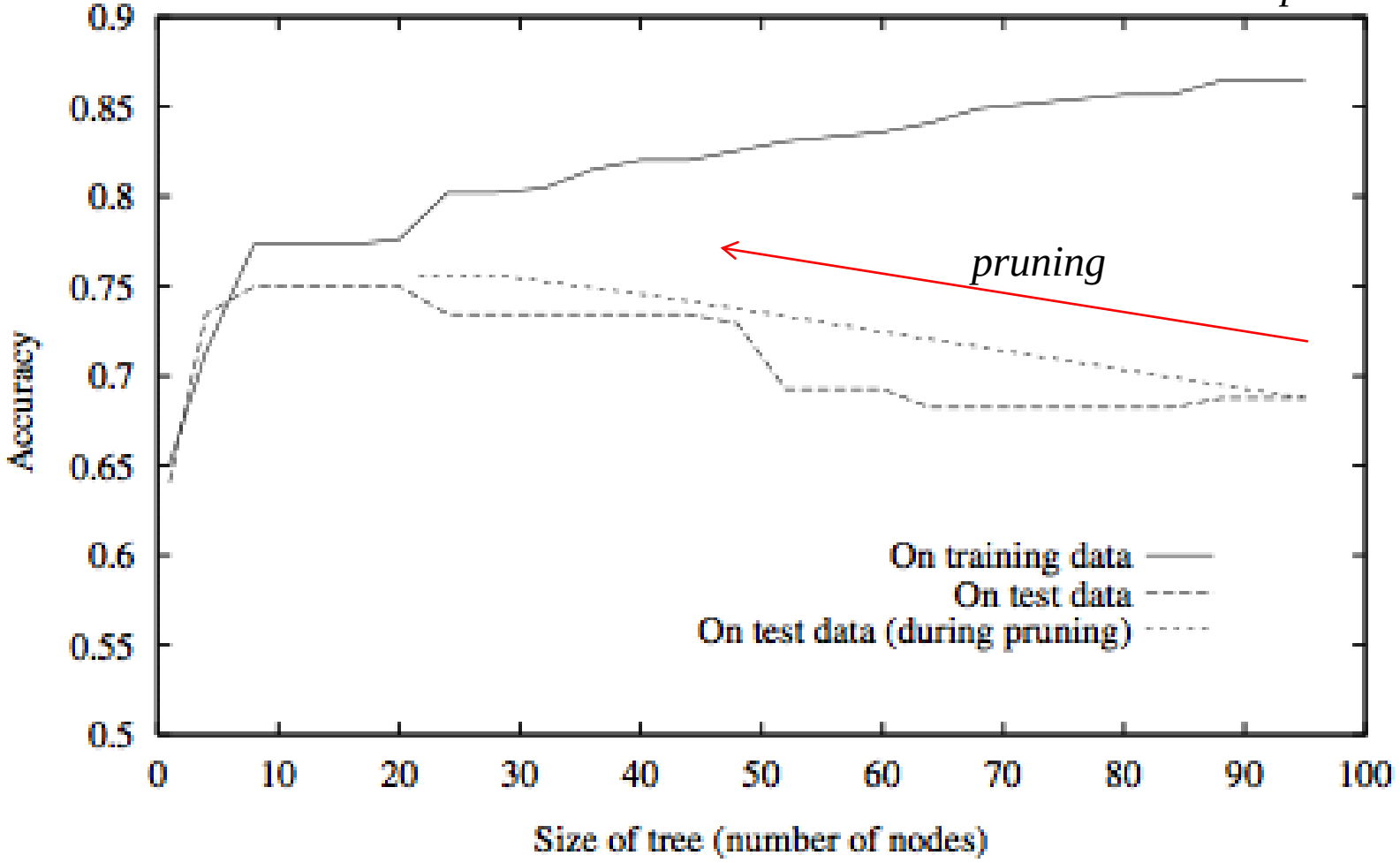
Dip. Informatica
University of Pisa

- Each node is a candidate for pruning
- *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
- Nodes are removed only if the resulting tree performs no worse on the validation set.
- Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned
- Pruning stops when no pruning increases accuracy

Effect of *reduced error* pruning

Dip. Informatica
University of Pisa

***Note**: Validation set results are not plotted here*



Rule post-pruning

Dip. Informatica
University of Pisa

1. Create the decision tree from the training set

2. Convert the tree into an equivalent set of rules
   - Each *path* corresponds to a rule
   - Each *node* along a path corresponds to a pre-condition
   - Each leaf classification to the post-condition

   e.g. (*Outlook=Sunny*)$\wedge$(*Humidity=High*) $\Rightarrow$ (*PlayTennis=No*)

3. Prune (generalize) each rule by removing those preconditions whose removal improves accuracy …
   - … over validation set
   - … over training with a pessimistic, statistically inspired, measure (heuristic)

4. Sort the rules in estimated order of accuracy, and consider them in sequence when classifying new instances

Why converting to rules?

Dip. Informatica
University of Pisa

- Each distinct path produces a different rule: a condition removal may be based on a local (contextual) criterion.
- Pruning of preconditions is rule specific (path), node pruning is global and affects all the rules (sub-tree)
- Converting to rules improves *readability* for humans (assuming a limited number of rules)

- Big topic: Explainable ML/AI , or Interpretable ML/AI → Trustworthy AI

Dealing with continuous-valued attributes

Dip. Informatica
University of Pisa

- So far discrete values for attributes and for outcome.
- Given a continuous-valued attribute A, dynamically create a new attribute A_c

$$A_c = \text{True } if\ A < c, \text{ False } otherwise$$

- How to determine threshold value c ?
- Example: *Temperature* in the *PlayTennis* example
  - Sort the examples according to *Temperature*

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

  - Determine candidate thresholds by averaging consecutive values where there is a change in classification: (48+60)/2=54 and (80+90)/2=85
  - Evaluate candidate thresholds (attributes) according to information gain. The best is *Temperature*>54.
    Then, the new attribute competes with the other ones

Handling incomplete training data (*imputation*)

Dip. Informatica
University of Pisa

- How to cope with the problem that the value of some attribute may be **missing**?
  - *Example*: Blood-Test-Result in a medical diagnosis problem
- The strategy: use other examples to guess attribute (within training data at a given node). *Imputation:*
  1. *Most common*. Assign the value that is most common among all the training examples at the node or those in the same class
  2. Assign a probability p_i to each value v_i, based on frequencies, and assign values to missing attribute, according to this probability distribution (adding more examples weighted by the prob., i.e. assigning fraction p_i of example to each descendant tree)
  3. Classify the new example in the same fashion (weighting): and the most probable classification is chosen (C4.5 algorithm)

Handling attributes with different costs

Dip. Informatica
University of Pisa

- Instance attributes may have an associated cost: we would prefer decision trees that use low-cost attributes
- ID3 can be modified to take into account costs:

1. Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(A)}$$

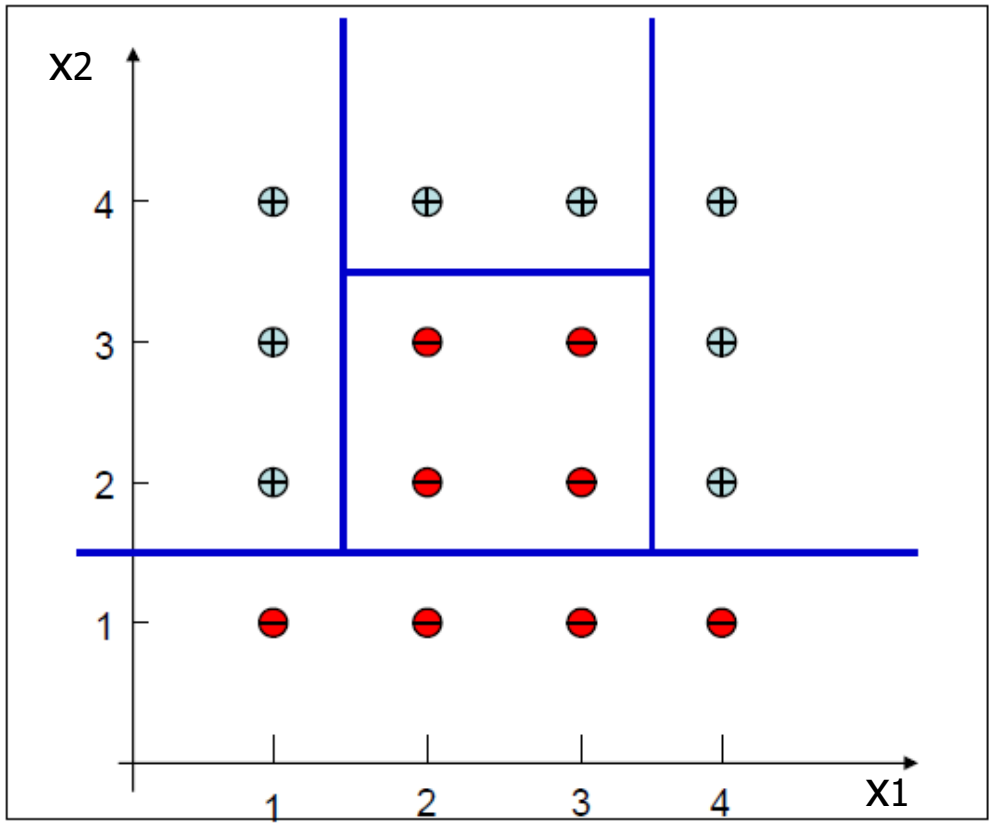
2. Nunez (1988)

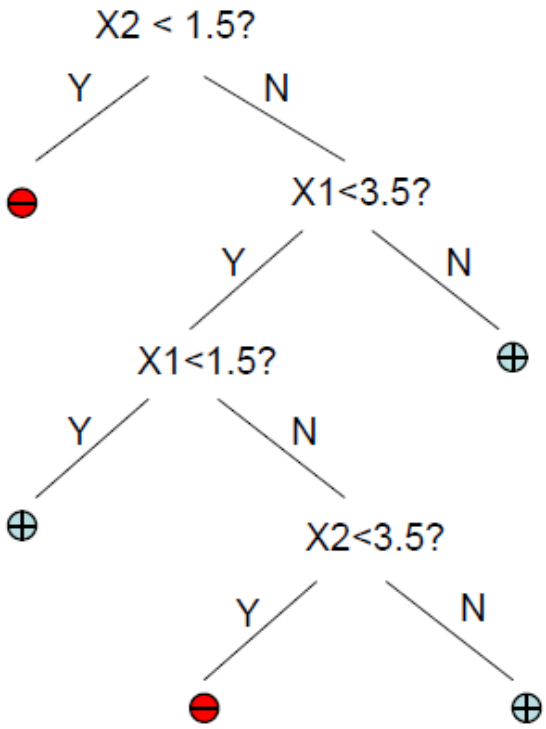
$$\frac{2^{Gain(S, A)} - 1}{(Cost(A) + 1)^w} \qquad w \in [0,1]$$

A geometrical view

Dip. Informatica
University of Pisa

- Decision boundaries that can be produced by a DT:
  Decision Trees divide the input space into axis-parallel rectangles and label each rectangle with one of the K classes (leaf of the tree)





Conclusions

Dip. Informatica
University of Pisa

- DT is a popular approach for classification in a discrete number of classes.
  - Expressive hypotheses space in the area of propositional - rule based approaches.
  - Robust to noisy data. Deal with Missing attribute values.
  - **Easy to be understood (explicit if-then rules,** unless they are a huge amount ;-) **)**
  - Many extensions to the basic scheme …
- *Language* and *search* biases: ID3 searches a complete hypothesis space, with a greedy incomplete strategy
- Flexible approach: Overfitting is an important issue, tackled by early stopping or post-pruning and generalization of induced rules

Past and Next lectures

Dip. Informatica
University of Pisa

Parallel development (description) of continuous (linear) and discrete (DT) hypotheses spaces:

- AND rules → extended to DT → add flexibility but overfitting issue
- Linear models → extended by phi (basis expansion) → add flexibility but overfitting issue

- It is time to see the common concept (theory) underlying these phenomena: next lecture

Bibliographic references (this lecture)

Dip. Informatica
University of Pisa

- T. M. Mitchell, *Machine learning*, McGraw-Hill, 1997: **chap 3**

- Other: AIMA: 18.

- In any ML/DM book.
- Many many papers and even books on DT !!!

DRAFT, please do not circulate!

For information

http://www.di.unipi.it/~micheli/DID

Alessio Micheli

micheli@di.unipi.it

www.di.unipi.it/groups/ciml

Dipartimento di Informatica
Università di Pisa - Italy

Computational Intelligence & Machine Learning Group