

Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

3. Decision Trees

L. Iocchi, F. Patrizi

Overview

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Inductive Bias
- Overfitting and pruning

References

T. Mitchell. Machine Learning. Chapter 3

Concept Learning as search

Problem: Given a training set D for a target function c , compute the "best" consistent hypothesis wrt D .

Solution approach

- Define hypothesis space H
- Implement an algorithm to find $h^* \in H$,
"best" hypothesis consistent with D

Decision Trees

Consider a discrete input space described with m attributes

$X = A_1 \times \dots \times A_m$, with A_i finite

and a classification problem for $f : X \rightarrow C$

Hypothesis space H : set of decision trees.

Decision tree

Tree with the following characteristics:

- Each internal node tests an attribute A_i
- Each branch denotes a value of an attribute $a_{i,j} \in A_i$
- Each leaf node assigns a classification value $c \in C$

Example *PlayTennis*

Instances $X = Outlook \times Temperature \times Humidity \times Wind$

$Outlook = \{Sunny, Overcast, Rain\}$

$Temperature = \{Hot, Mild, Cold\}$

$Humidity = \{Normal, High\}$

$Wind = \{Weak, Strong\}$

Classification values:

$PlayTennis = \{Yes, No\}$

Learning problem:

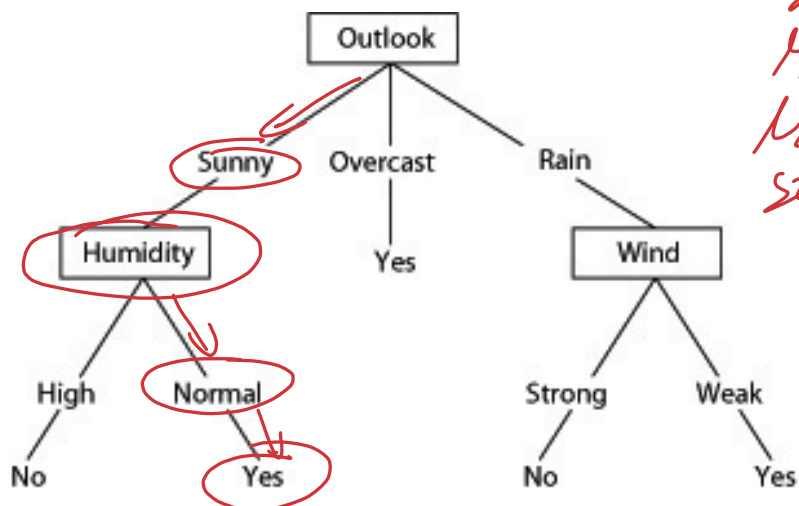
$f : Outlook \times Temperature \times Humidity \times Wind \rightarrow PlayTennis$

Example *PlayTennis*: Training data

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	<u>Rain</u>	Cool	Normal	<u>Weak</u>	<u>Yes</u>
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	<u>Sunny</u>	Mild	<u>High</u>	Weak	<u>No</u>
D9	<u>Sunny</u>	Cool	<u>Normal</u>	Weak	<u>Yes</u>
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 Tree for *PlayTennis*

This tree is one particular hypothesis



Input:
Sunny
Mild
Normal
Strong

Decision Tree for *PlayTennis*

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

*All the
yes combinations.*

$$\begin{aligned} & (Outlook = Sunny \wedge Humidity = Normal) \vee \\ & (Outlook = Overcast) \vee \\ & (Outlook = Rain \wedge Wind = Weak) \end{aligned}$$

Disjunction of conjunctions of all the paths to positive (true) leaf nodes.

Converting a Tree to Rules

A rule is generated for each path to a leaf node.

IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

Decisions are made explicit.

In this case:

		yes
		yes
		yes
		yes

→ always +

ID3 Algorithm

given training set
Used to build dec. tree.

Input: *Examples*, *Target_attribute*, *Attributes*

Output: *Decision Tree*

- 1 Create a Root node for the tree
- 2 if all Examples are positive, then return the node Root with label +
- 3 if all Examples are negative, then return the node Root with label - → the opposite.
- 4 if Attributes is empty, then return the node Root with label = most common value of Target_attribute in Examples
- 5 Otherwise ...

dataset with only positive.
(you play always, no matter what).

See example ch
10: 46

ID3 Algorithm

5. Otherwise

- $A \leftarrow$ the "best" decision attribute for Examples
- Assign A as decision attribute for Root
- For each value v_i of A
 - add a new branch from Root corresponding to the test $A = v_i$
 - $Examples_{v_i}$ = subset of Examples that have value v_i for A
 - if $Examples_{v_i}$ is empty then add a leaf node with label = most common value of Target_attribute in Examples
 - else
 - add the tree $ID3(Examples_{v_i}, Target_attribute, Attributes - \{A\})$

Use decision tree why is the best

See example
ch 10: 55.

If we don't have an attribute it means that that attribute is not important for the decision, we have decided it

Which is the best attribute to choose?

Output tree depends on attribute order

Example (dataset different than previous one):

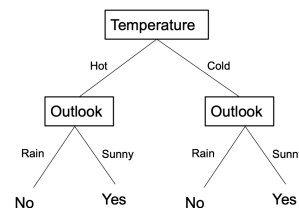
Day	Outlook	Temperature	PlayTennis
D1	Sunny	Hot	Yes
D2	Sunny	Cold	Yes
D3	Rain	Hot	No
D4	Rain	Cold	No

It can change dimension and speed of arriving to the end

Outlook first:



Temperature first:



Which is the best attribute to choose?

Information gain measures how well a given attribute *separates* the training examples according to their target classification.

Day	Outlook	Temperature	PlayTennis
D1	Sunny	Hot	Yes
D2	Sunny	Cold	Yes
D3	Rain	Hot	No
D4	Rain	Cold	No

Day	Outlook	Temperature	PlayTennis
D1	Sunny	Hot	Yes
D2	Sunny	Cold	Yes
D3	Rain	Hot	No
D4	Rain	Cold	No

Day	Outlook	Temperature	PlayTennis
D1	Sunny	Hot	Yes
D3	Rain	Hot	No
D2	Sunny	Cold	Yes
D4	Rain	Cold	No

Given dataset entropy tells us how unpu

ID3 selects the attribute that induces **highest information gain**.

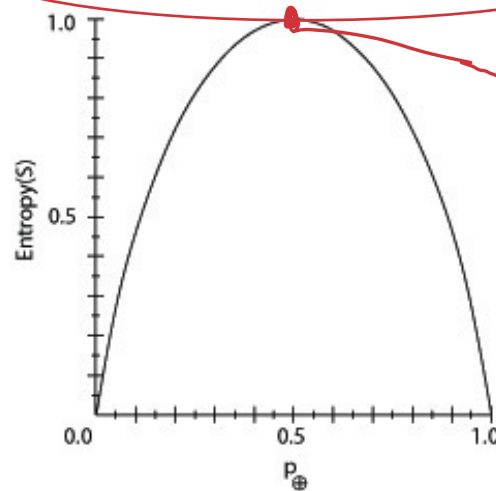
Information gain measured as reduction in **entropy**.

How much a dataset is impure

Entropy

- p_{\oplus} is the proportion of positive examples in S $\rightarrow \frac{+}{N}$
- $p_{\ominus} (= 1 - p_{\oplus})$ is the proportion of negative examples in S $\rightarrow \frac{-}{N} = 1 - \frac{+}{N}$
- Entropy measures the *impurity* of S

$$\text{Entropy}(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



Maximum of impurity
↓

Entropy example

Consider the set $S = [9+, 5-]$ (9 positive examples, 5 negative examples)

Proportion $\text{Entropy}(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$ *Good.*

Note: we define $0 \log_2 0 = 0$

Maximum entropy when $p_{\oplus} = 0.5$, minimum when $p_{\oplus} = 0$ or 1 .

In case of multi-valued target functions (c -wise classification)

$$\text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

\rightarrow Proportion of examples classified as c .

Information Gain

$Gain(S, A)$ = expected reduction in entropy of S caused by knowing the value of attribute A .

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

$Values(A)$: set of all possible values of A
 $S_v = \{s \in S | A(s) = v\}$

Information Gain example

$Values(Wind) = \{Weak, Strong\}$

$S = [9+, 5-]$

$S_{Weak} = [6+, 2-]$

$S_{Strong} = [3+, 3-]$

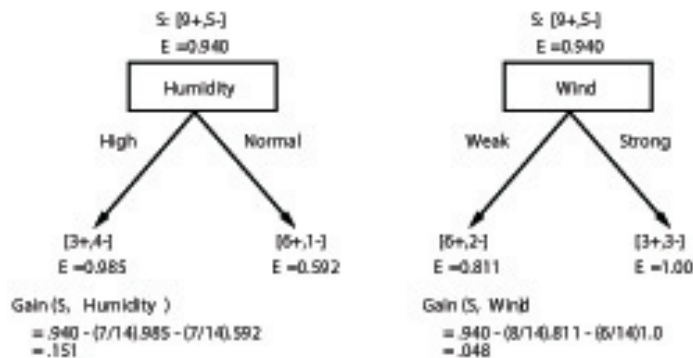
knowing Wind

Wind

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \left[\frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong}) \right] \\ &= 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1.00 \\ &= 0.048 \end{aligned}$$

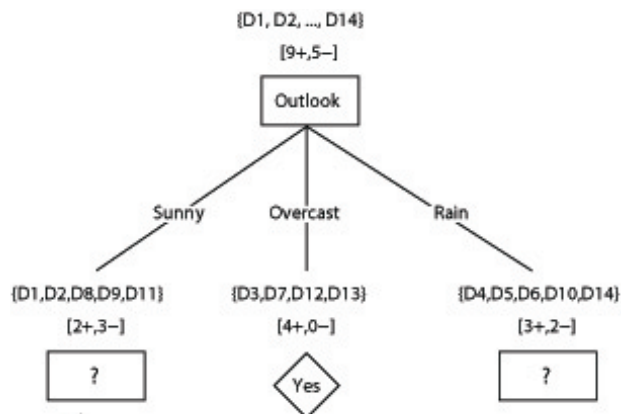
Selecting the Next Attribute

Which attribute is the best classifier?



$$\begin{aligned} \text{Gain}(S, \text{Outlook}) &= 0.246 \\ \text{Gain}(S, \text{Humidity}) &= 0.151 \\ \text{Gain}(S, \text{Wind}) &= 0.048 \\ \text{Gain}(S, \text{Temperature}) &= 0.029 \end{aligned}$$

Selecting the Next Attribute



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

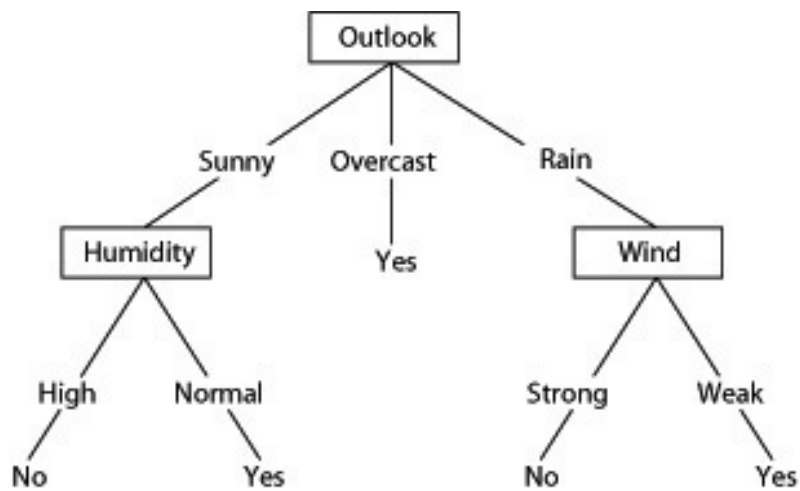
$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) \cdot 0.0 - (2/5) \cdot 0.0 = .970$$

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

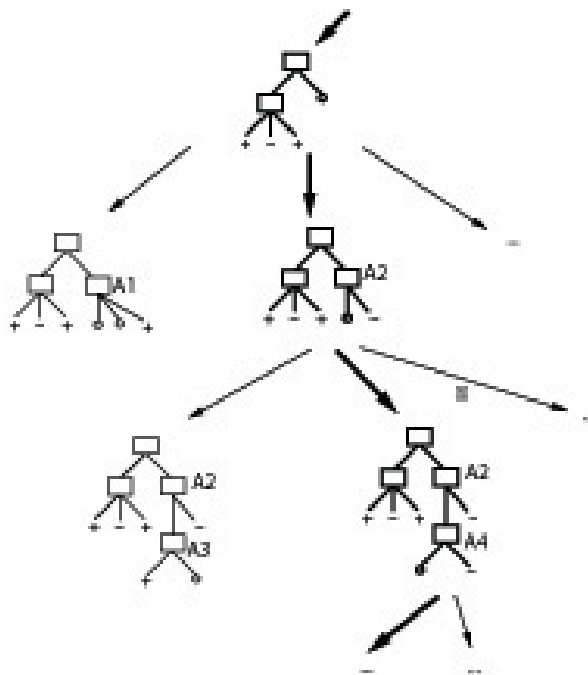
$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) \cdot 1.0 - (3/5) \cdot 0.18 = .019$$

the next will be humidity

Decision Tree for *PlayTennis*



Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

Hyp space is Complete when it contains all possible hypothesis

- Hypothesis space is complete (target concept is there!)
- Outputs a single hypothesis (cannot determine how many DTs are consistent!)
- No back tracking (local minima!) *never move back.*
- Statistically-based search choices (robust to noisy data!)
- Uses all the training examples at each step (not incremental!)

Issues in Decision Tree Learning

If you end before the leaf you will choose with less accuracy

- Determining how deeply to grow the DT
- Handling continuous attributes
- Choosing appropriate attribute selection measures
- Handling training data with missing attribute values
- Handling attributes with different costs

Overfitting in Decision Trees

Consider a new data set $D' = D \cup d_{15}$ adding a training example:

$$d_{15} = \langle \text{Sunny}, \text{Hot}, \text{Normal}, \text{Strong}, \text{PlayTennis} = \text{No} \rangle$$

ID3 will generate a different tree T'

Note: T is consistent with D and T' is consistent with D' (i.e., accuracy = 100%)

Is T' in general a better solution for our learning problem?

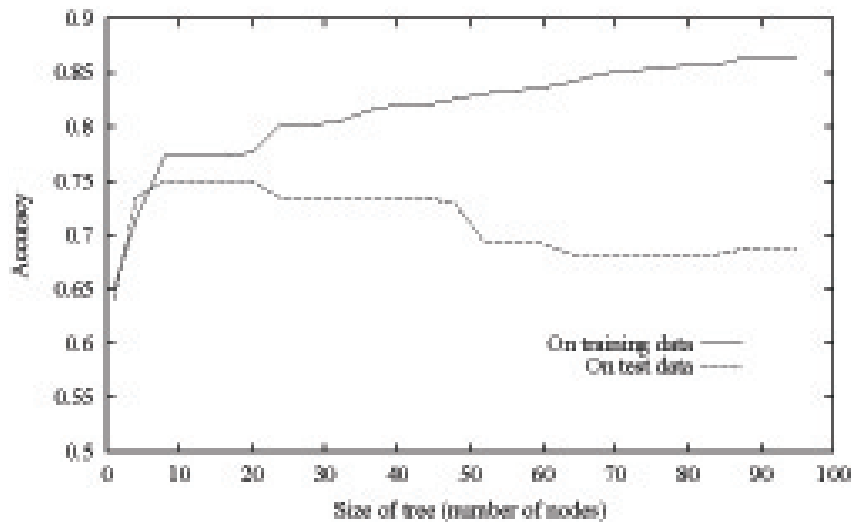
Overfitting in Decision Trees

Let's consider a more complex problem with $|D| \gg 15$ and containing noisy data and two decision trees T and T' obtained with different configuration of an ID3-like algorithm.

$$accuracy_D(T') > accuracy_D(T)$$

Is T' in general a better solution for our learning problem?

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

To determine the correct tree size

- use a separate set of examples (distinct from the training examples) to evaluate the utility of post-pruning
- apply a statistical test to estimate accuracy of a tree on the entire data distribution
- using an explicit measure of the complexity for encoding the examples and the decision trees.

Reduced-Error Pruning

See screenshot at 12:45

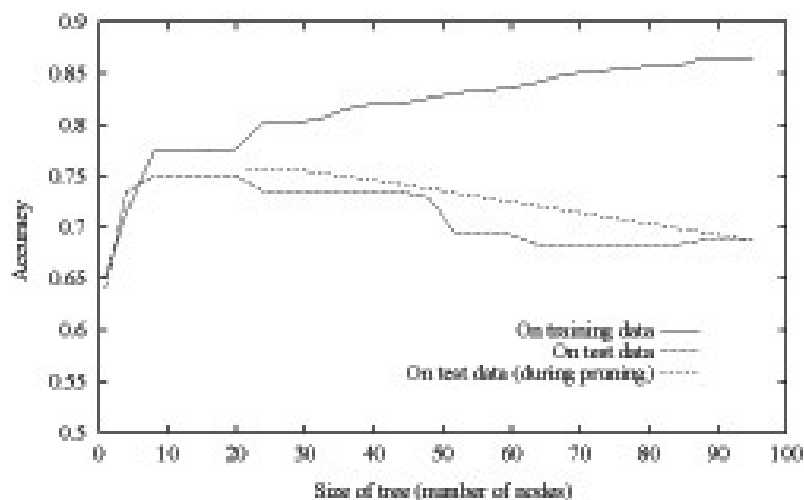
Split data into *training* and *validation* set

degradation in accuracy.

Do until further pruning is harmful (decreases accuracy):

- ① Evaluate impact on *validation* set of pruning each possible node (remove all the subtree and assign the most common classification)
- ② Greedily remove the one that most improves *validation* set accuracy

Effect of Reduced-Error Pruning



Remarks on Reduced-Error Pruning

- it produces smallest version of most accurate subtree (removing sub-trees added due to coincidental irregularities).
- When data set is limited, reducing the set of training examples (used as validation examples) can give bad results.

Rule Post-Pruning (C4.5)

- ① Infer the decision tree allowing for overfitting
- ② Convert the learned tree into an equivalent set of rules
- ③ Prune (generalize) each rule independently of others
- ④ Sort final rules into desired sequence for use

Continuous Valued Attributes

Create a discrete attribute to test continuous variables

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

Attributes with Many Values

Problem:

- If attribute has many values, *Gain* will select it
- Imagine using $Date = Jun_3_1996$ as attribute

One approach: use *GainRatio* instead

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

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

where S_i is subset of S for which A has value v_i

Attributes with Costs

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, *Width_from_1ft* has cost 23 sec.

How to learn a consistent tree with low expected cost?

Replace gain by

- Tan and Schlimmer (1990)

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

- Nunez (1988) ($w \in [0, 1]$ determines importance of cost)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

Unknown Attribute Values

What if some examples missing values of A ?

Use training example anyway, sort through tree

- If node n tests A , assign most common value of A among other examples sorted to node n
- assign most common value of A among other examples with same target value
- assign probability p_i to each possible value v_i of A
 - assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion

Other algorithms based on Decision Trees

Random Forest: ensemble method that generates a set of decision trees with some random criteria and integrates their values into a final result.

Random criteria: 1) random subsets of data (bagging), 2) random subset of attributes (feature selection), ...

Integration of results: majority vote (most common class returned by all the trees).

Random Forests are less sensitive to overfitting.

Summary

- Decision Trees can represent classification function by **making decisions explicit**
- Learning as search in the hypothesis space with heuristics based on information gain
- Statistical method (some robustness to noisy data)
- Overfitting and pruning
- Used as basis of randomized ensemble methods