Methods for Intelligent Systems

Lecture Notes on Machine Learning

Matteo Mattecci

matteucci@elet.polimi.it

Department of Electronics and Information

Politecnico di Milano

Classification Algorithms

– Decision Trees –

What is Classification?

Whenever we build a **model** from data we can apply it to new (unforeseen) examples to get the desired output. According to such an output we have:

Classification: The desired outputs y_i are discrete class labels. The goal is to classify new inputs correctly.

Regression: The desired outputs y_i are continuous valued. The goal is to predict the output accurately for new inputs.

The information used by the model for classification can be composed by:

Categorical Attribute: an attribute which takes on two or more discrete values (a.k.a. symbolic attributes)

Real Attribute: an attribute which takes real values

Decision Trees are just one of such models!

Decision Trees: Definition

A Decision Tree is an arrangement of tests that prescribes an appropriate test at every step in an analysis:

- a method for approximating discrete-valued target functions,
- capable of learning disjunctive expressions,
- robust to noisy data.

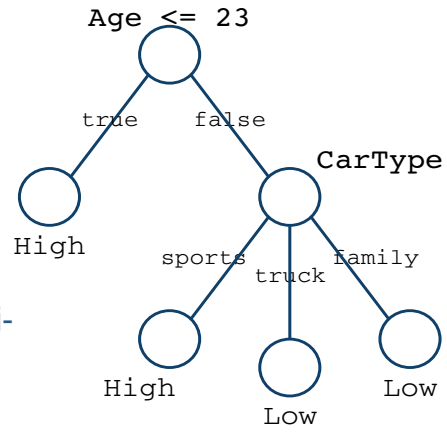
This lectures has been (heavily) inspired by:

- `www.autonlab.org/tutorials/`
- `www.cs.uregina.ca/~hamilton/courses/831/index.html`
- T. Mitchell, “Decision Tree Learning”, in T. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc., 1997, pp. 52-78
- P. Winston, “Learning by Building Identification Trees”, in P. Winston, Artificial Intelligence, Addison-Wesley Publishing Company, 1992, pp. 423-442.

Decision Trees: Representation

Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

- A Node represent a test on attributes' values
- An Arch represent the result of the test
- Nodes with no outgoing arches are called Leaves
- Leaves represent the resulting classification



Decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides instance classification.

When should I use a Decision Tree?

Decision tree learning is generally best suited to problems with:

- Instances represented by attribute-value pairs.
  - Records are described by a fixed set of attributes (e.g., temperature) and their values (e.g., hot).
  - Each attribute takes on a small number of disjoint possible values (e.g., hot, mild, cold).
  - Extensions to the basic algorithm allow handling real-valued attributes as well (e.g., a floating point temperature).
- The target function has discrete output values.
  - A decision tree assigns a classification to each example.
  - Extensions allow learning target functions with real-valued outputs, although the application of decision trees in this setting is less common.
- The training data may contain errors or missing attribute values
- Disjunctive descriptions may be required

A (Very) Small Example Dataset

Suppose you work for *TenenTel Insurance Ltd.* as a consultant:

Age	CarType	Risk
17	sprots	High
43	family	Low
68	family	Low
32	truck	Low
23	family	High
18	family	High
20	family	High
45	sports	High
50	truck	Low
64	truck	High
46	family	Low
40	family	Low

- An expert already assigned clients to a “Risk” level
- The company would like an automatic procedure to obtain the same result on new records
- The company is also looking for a compact representation of the classification process to be used by a computer

Can we learn a Decision Tree out of these data?

Learning Decision Trees: A Simple Idea

A record is classified by starting at the root node of the decision tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute.

It is computationally impractical to find the smallest possible Decision Tree, so we use a procedure that tends to build small trees.

- Start from the root
- Select an attribute
- Split the data according to that attribute
- Recurse . . .

How should I choose the attribute for splitting?
Information Gain! (That’s too obvious)
What about real values?
Oppps, this is less obvious . . .
When should I stop recursion?
Hmmm . . . let me think a little bit

Dealing With Real Attributes

Use a thresholded split to compute Entropy

- Suppose X is the real-valued attribute
- Define $IG(Y|X:t) = H(Y) - H(Y|X:t)$
- Define

$$\begin{aligned} H(Y|H:t) &= H(Y|X \leq t)P(X \leq t) + \\ &\quad + H(Y|X > t)P(X > t) \end{aligned}$$

- Define $IG^*(Y|X) = \max_t IG(Y|X:t)$
- For each real-valued attribute, use $IG^*(Y|X)$ for selecting the split

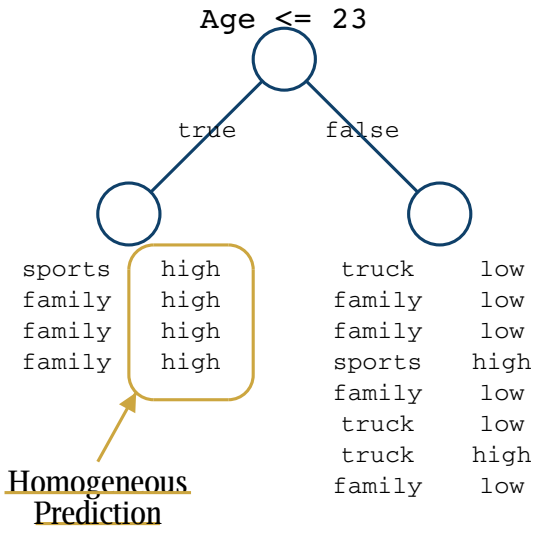
Consider each test on a real-valued attribute as a boolen test on the attribute being less or equal than such threshold value.

When should I stop splitting?

Stopping Recursion: Case Base 1

Each split generates new datasets:

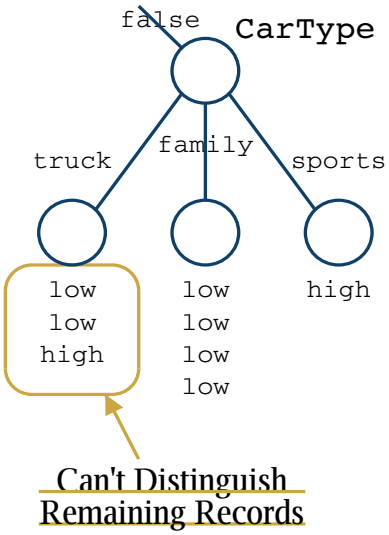
- stop if the new dataset has a homogeneous prediction
- otherwise recurse on the reduced dataset



Stopping Recursion: Case Base 2

Suppose, for this example, you treat real-valued attributes as categorical ones and you remove them when splitting:

- stop if you can't distingush the records in the new dataset
- predict the most common class or randomly if equally likely



Stopping Recursion: "What if Information Gain is 0?"

Consider the following example: $Y = A$ xor B

$$
\begin{aligned}
H(Y) &= 1 \\
H(Y|A) &= P(\bar{A})H(Y|\bar{A})P(A)H(Y|A) \\
&= 1/2 \times 1 + 1/2 \times 1 = 1 \\
H(Y|B) &= P(\bar{B})H(Y|\bar{B})P(B)H(Y|B) \\
&= 1/2 \times 1 + 1/2 \times 1 = 1
\end{aligned}
$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Should I stop Recursion?

- if I stop recursion when Information Gain is zero
- randomly predict one of the output
- 50% Error Rate

Stopping Recursion: “What if Information Gain is 0?”

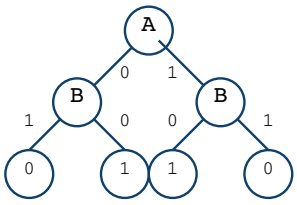
Consider the following example: $Y = A$ xor B

$$
\begin{aligned}
H(Y) &= 1 \\
H(Y|A) &= P(\bar{A})H(Y|\bar{A})P(A)H(Y|A) \\
&= 1/2 \times 1 + 1/2 \times 1 = 1 \\
H(Y|B) &= P(\bar{B})H(Y|\bar{B})P(B)H(Y|B) \\
&= 1/2 \times 1 + 1/2 \times 1 = 1
\end{aligned}
$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Should I stop Recursion?

- if I randomly split when Information Gain in zero
- 0% Error Rate



Learning Decision Tree: The Algorithm

`Node BuildTree(Dataset, Output)`

- `if` all output values are the same in Dataset, `return` a leaf node that says “predict this unique output”
- `if` all input values are the same, `return` a leaf node that says “predict the majority output”
- `else` find attribute X with Highest Information Gain
- Suppose X has n_X distinct values
  - Create and return a node with n_X children
  - The i^{th} children is given by

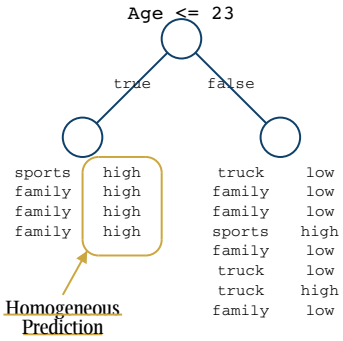
`BuildTree(Dataset`$_i$`, Output)`

Decision Trees: An Example (I)

Build the Decision Trees to classify the `Risk` level from `Age` and `CarType`.

Age	CarType	Risk
17	sprots	High
43	family	Low
68	family	Low
32	truck	Low
23	family	High
18	family	High
20	family	High
45	sports	High
50	truck	Low
64	truck	High
46	family	Low
40	family	Low

- What's the Information Gain for `CarType` Attribute?
- What's the Information Gain for `Age` Attribute?
- Which attibute should I split first?

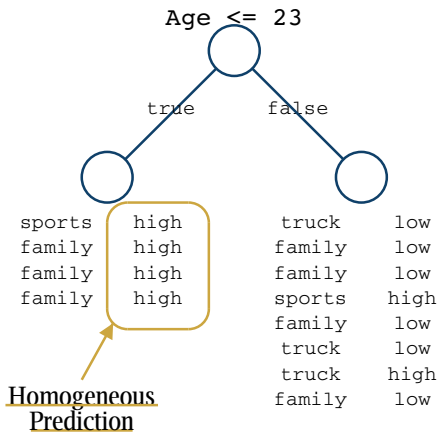


Decision Trees: An Example (II)

Build the Decision Trees to classify the `Risk` level from `Age` and `CarType`.

Age	CarType	Risk
17	sprots	High
43	family	Low
68	family	Low
32	truck	Low
23	family	High
18	family	High
20	family	High
45	sports	High
50	truck	Low
64	truck	High
46	family	Low
40	family	Low

- Should I compute the Information Gain?
- Should I split the `CarType` Attribute?

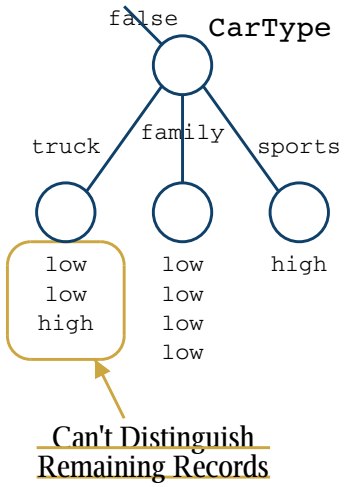


Decision Trees: An Example (III)

Build the Decision Trees to classify the `Risk` level from `Age` and `CarType`.

Age	CarType	Risk
17	sprots	High
43	family	Low
68	family	Low
32	truck	Low
23	family	High
18	family	High
20	family	High
45	sports	High
50	truck	Low
64	truck	High
46	family	Low
40	family	Low

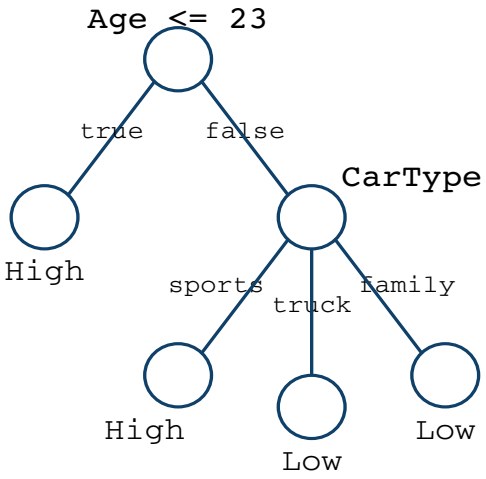
- Can I split again?



Decision Trees: An Example (IV)

Build the Decision Trees to classify the `Risk` level from `Age` and `CarType`.

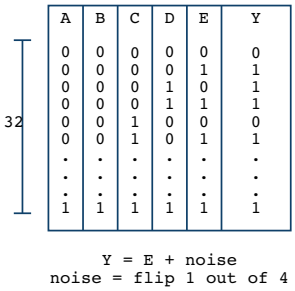
Age	CarType	Risk
17	sprots	High
43	family	Low
68	family	Low
32	truck	Low
23	family	High
18	family	High
20	family	High
45	sports	High
50	truck	Low
64	truck	High
46	family	Low
40	family	Low



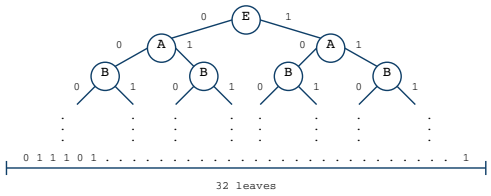
Overfitting in Decision Trees

Consider the following artificial dataset:

- the outpt Y is a copy of E plus a 25% noise
- attributes A, B, C, D are irrelevant to predict Y



We can learn a Decision Tree perfetly predicting the output Y, but it will be corrupted by noise 25% of the times



What if we get new data from the same process?
You'll get 1 out of 4 errors!

Overfitting

Overfitting: we have overfitting when the model we learnt fits the noise in the data, thus it does not generalize on new samples (it just memoryzes the training set).

Can we measure generalization?

- Hide some data before learning the tree (*Test Set*)
- Estimate how well the tree predict on "new" data (*Test Set Error*)

Can we avoid overfitting?

- Statistical tests on the data
- Use cross-validation techniques
- Introduce a bias into the model

General rule: Use Occam's razor!
"Entia non sunt multiplicanda praeter necessitatem"

Avoiding Overfitting

How can we avoid overfitting in Decision Trees?

- Statistical Test or Cross-Validation
  - Stop growing the tree when data split not statistically significant
  - Stop growing the tree when generalization error increase
- Post Pruning
  - Grow full tree, then post-prune the tree
  - Grow full tree, transform it into decision rules, post-prune the rules

Converting a Decision Tree to rules before pruning has some advantages:

- Big Decision Trees might be difficult to understand from a human user
- They can be translated into an equivalent representation known as Decision Rules
  - `IF Age<=23 OR CarType IS sports`
    `THEN Risk IS High`
  - `IF Age>23 AND CarType IS family OR truck`
    `THEN Risk IS Low`

Rule Generation Advantages

To generate rules, trace each path in the decision tree, from root node to leaf node, recording the test outcomes as antecedents and the leaf-node classification as the consequent. This has three advantages:

- Allows distinguishing among the different contexts in which a decision node is used
  - pruning decision regarding an attribute test can be made differently for each path.
  - if the tree itself were pruned, the only two choices would be remove the decision node completely, or retain it
- Removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves (i.e., avoiding to reorganize the tree if the root node is pruned while retaining part of the subtree below this test)
- Converting to rules improves readability.

Rule Pruning

Once a rule set (i.e., all the paths to the leaves) has been devised:

1. Eliminate unecessary rule antecedents to simplify the rules
   - To simplify a rule, eliminate antecedents that have no effect on the conclusion reached by the rule
   - Independence from an antecendent is verified using a test for independency
2. Eliminate unecessary rules to simplify the rule set.
3. Replace those rules that share the most common consequent by a default rule that is triggered when no other rule is triggered.

We can use the following independence tests:

- Chi-Square (cell frequencies $m > 10$): $\chi^2 = \sum_i \sum_j \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$
- Yates' Correction ($5 \leq m \leq 10$): $\chi^2 = \sum_i \sum_j \frac{(|o_{ij} - e_{ij}| - 0.5)^2}{e_{ij}}$
- Fisher's Exact Test ($m < 5$): *see Winston, pp. 437-442*

Contingency Table

A contingency table is the tabular representation of a rule:

	C_1	C_2	
R_1	x_{11}	x_{12}	$R_{1T} = x_{11} + x_{12}$
R_2	x_{21}	x_{22}	$R_{2T} = x_{21} + x_{22}$
	$C_{T1} = x_{11} + x_{21}$	$C_{T2} = x_{12} + x_{22}$	$T = x_{11} + x_{12} + x_{21} + x_{22}$

- R_1 and R_2 represent the Boolean states of an antecedent for the conclusions C_1 and C_2 (C_2 is the negation of C_1)
- x_{11}, x_{12}, x_{21}, and x_{22} represent the frequencies of each antecedent-consequent pair.
- R_{1T}, R_{2T}, C_{T1}, and C_{T2} are the marginal sums of the rows and columns, respectively.

The marginal sums and T, the total frequency of the table, are used to calculate expected cell values in test for independence.

Test for Independence

Given a contingency table of dimensions r by c (rows x columns):

1. Calculate the marginal sums.
2. Calculate the total frequency, T, using the marginal sums.
3. Calculate the expected frequencies for each cell: $e_{ij} = R_{iT} \cdot C_{Tj}/T$
4. Select the test to be used based on the highest expected frequency m
5. Calculate χ^2 using the chosen test
6. Calculate the degrees of freedom: $df = (r-1)(c-1)$
7. Use a chi-square table with χ^2 and df to determine if the conclusions are independent from the antecedent at the selected level of significance α (usually $\alpha = 0.05$).
   - $\chi^2 > \chi^2_\alpha$: reject the null hypothesis of independence (keep the antecedents)
   - $\chi^2 \leq \chi^2_\alpha$: accept the null hypothesis of independence (discard the antecedents)

A Complete Example

Suppose you get made and interview among you friends after their Summer holidays to classify sunburn risk factors:

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

- Can you set up a Decision Tree to predict if some one will get a sunburn? Can you prune the resulting tree into a few simple rules?
- Can you do that without using the **C4.5 free software**?