Decision Trees

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

This lecture is largely based on the book
Artificial Intelligence: A Modern Approach, 3rd ed.
by Stuart Russell and Peter Norvig (Prentice Hall, 2010).

What is a decision tree?

Decision tree

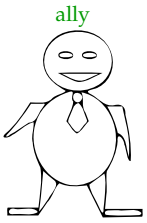
- ✔ is a function that
  - ✘ takes a vector of attribute values as its input, and
  - ✘ returns a "decision" as its output.
  - ✘ Both input and output values can be measured on a nominal, ordinal, interval, and ratio scales, can be discrete or continuous.
- ✔ The decision is formed via a sequence of tests:
  - ✘ each internal node of the tree represents a test,
  - ✘ the branches are labeled with possible outcomes of the test, and
  - ✘ each leaf node represents a decision to be returned by the tree.

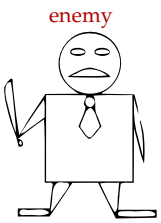
Decision trees examples:

- ✔ classification schemata in biology (určovací klíče)
- ✔ diagnostic sections in illness encyclopedias
- ✔ online troubleshooting section on software web pages
- ✔ ...

Attribute description

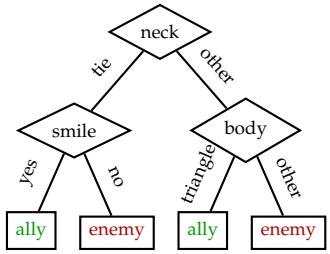
Example: A computer game.
The main character of the game meets various robots along his way. Some behave like allies, others like enemies.





head	body	smile	neck	holds	class
circle	circle	yes	tie	nothing	ally
circle	square	no	tie	sword	enemy
...	...	...	...	...	...

The game engine may use e.g. the following tree to assign the ally or enemy attitude to the generated robots:



Expressiveness of decision trees

The tree on previous slide is a Boolean decision tree:

- ✔ the decision is a binary variable (true, false), and
- ✔ the attributes are discrete.
- ✔ It returns ally iff the input attributes satisfy one of the paths leading to an ally leaf:

$$ally \Leftrightarrow (neck = tie \wedge smile = yes) \vee (neck = \neg tie \wedge body = triangle),$$

i.e. in general

- ✘ $Goal \Leftrightarrow (Path_1 \vee Path_2 \vee \ldots)$, where
- ✘ $Path$ is a conjuction of attribute-value tests, i.e.
- ✘ the tree is equivalent to a DNF of a function.

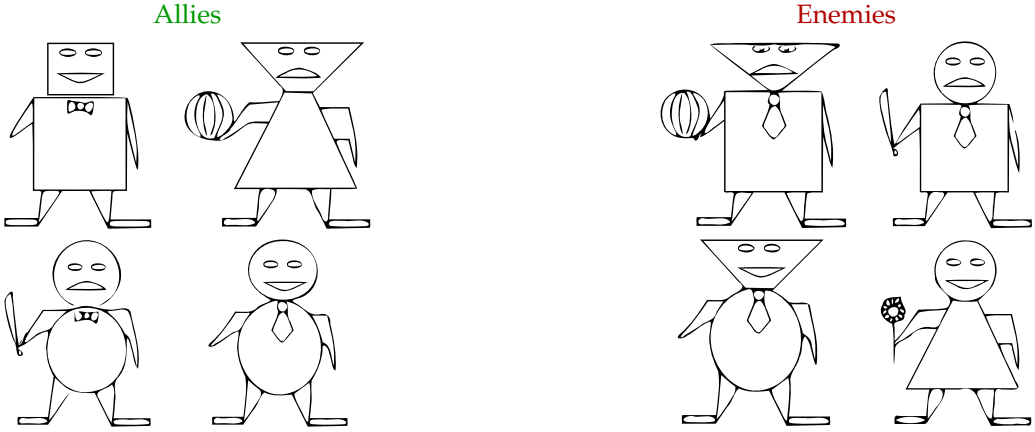
Any function in propositional logic can be expressed as a dec. tree.

- ✔ Trees are a suitable representation for some functions and unsuitable for others.
- ✔ What is the cardinality of the set of Boolean functions of n attributes?
  - ✘ It is equal to the number of truth tables that can be created with n attributes.
  - ✘ The truth table has 2^n rows, i.e. there is 2^{2^n} different functions.
  - ✘ The set of trees is even larger; several trees represent the same function.
- ✔ We need a clever algorithm to find good hypotheses (trees) in such a large space.

Learning a Decision Tree 6 / 29

A computer game

Example 1:
Can you distinguish between allies and enemies after seeing a few of them?

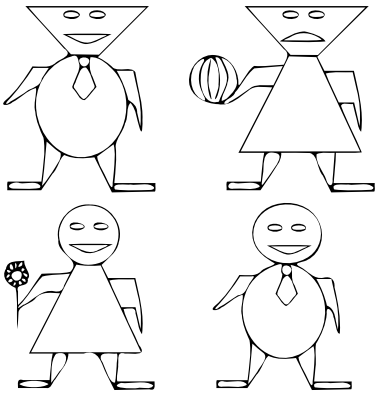


Hint: concentrate on the shapes of heads and bodies.
Answer: Seems like allies have the same shape of their head and body.
How would you represent this by a decision tree? (Relation among attributes.)
How do you know that you are right?

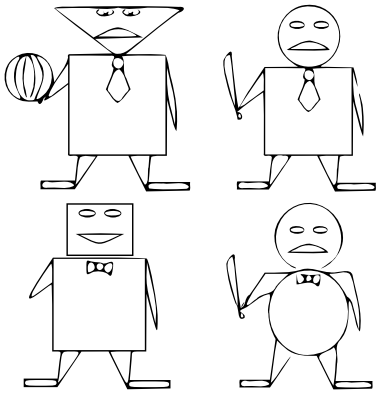
A computer game

Example 2:
Some robots changed their attitudes:

Allies



Enemies



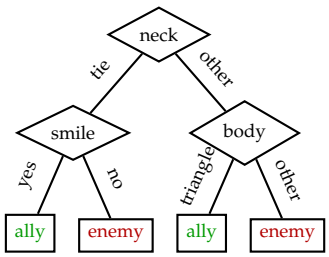
No obvious simple rule.
How to build a decision tree discriminating the 2 robot classes?

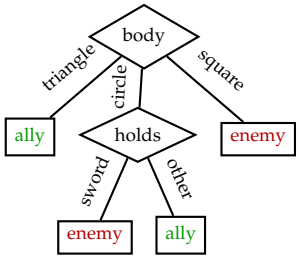
Alternative hypotheses

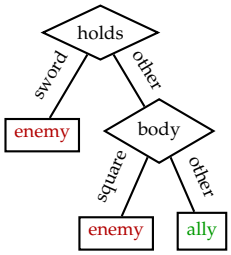
Example 2: Attribute description:

head	body	smile	neck	holds	class
triangle	circle	yes	tie	nothing	ally
triangle	triangle	no	nothing	ball	ally
circle	triangle	yes	nothing	flower	ally
circle	circle	yes	tie	nothing	ally
triangle	square	no	tie	ball	enemy
circle	square	no	tie	sword	enemy
square	square	yes	bow	nothing	enemy
circle	circle	no	bow	sword	enemy

Alternative hypotheses (suggested by an oracle for now): Which of the trees is the best (right) one?







How to choose the best tree?

We want a tree that is

- ✔ **consistent** with the data,
- ✔ is as **small** as possible, and
- ✔ which also **works for new data**.

Consistent with data?

- ✔ All 3 trees are consistent.

Small?

- ✔ The right-hand side one is the simplest one:

	left	middle	right
depth	2	2	2
leaves	4	4	3
conditions	3	2	2

Will it work for new data?

- ✔ We have no idea!
- ✔ We need a set of new testing data (different data from the same source).

Learning a Decision Tree

It is an intractable problem to find **the smallest consistent tree** among $> 2^{2^n}$ trees.
We can find approximate solution: **a small (but not the smallest) consistent tree**.

Top-Down Induction of Decision Trees (TDIDT):

- ✔ A greedy divide-and-conquer strategy.
- ✔ Progress:
  1. Test the most important attribute.
  2. Divide the data set using the attribute values.
  3. For each subset, build an independent tree (recursion).
- ✔ "Most important attribute": attribute that makes the most difference to the classification.
- ✔ All paths in the tree will be short, the tree will be shallow.

Attribute importance

head	body	smile	neck	holds	class
triangle	circle	yes	tie	nothing	ally
triangle	triangle	no	nothing	ball	ally
circle	triangle	yes	nothing	flower	ally
circle	circle	yes	tie	nothing	ally
triangle	square	no	tie	ball	enemy
circle	square	no	tie	sword	enemy
square	square	yes	bow	nothing	enemy
circle	circle	no	bow	sword	enemy

triangle:	2:1	triangle:	2:0	yes:	3:1	tie:	2:2	ball:	1:1
circle:	2:2	circle:	2:1	no:	1:3	bow:	0:2	sword:	0:2
square:	0:1	square:	0:3			nothing:	2:0	flower:	1:0
								nothing:	2:1

A perfect attribute divides the examples into sets each of which contain only a single class. (Do you remember the simply created perfect attribute from Example 1?)

A useless attribute divides the examples into sets each of which contains the same distribution of classes as the set before splitting.

None of the above attributes is perfect or useless. Some are more useful than others.

Choosing the test attribute

Information gain:

- ✔ Formalization of the terms "useless", "perfect", "more useful".
- ✔ Based on entropy, a measure of the uncertainty of a random variable V with possible values v_i:

$$H(V) = -\sum_i p(v_i) \log_2 p(v_i)$$

- ✔ Entropy of the target class C measured on a data set S (a finite-sample estimate of the true entropy):

$$H(C,S) = -\sum_i p(c_i) \log_2 p(c_i),$$

where $p(c_i) = \frac{N_S(c_i)}{|S|}$, and $N_S(c_i)$ is the number of examples in S that belong to class c_i.

- ✔ The entropy of the target class C **remaining in the data set** S **after splitting** into subsets S_k using values of attribute A (weighted average of the entropies in individual subsets):

$$H(C,S,A) = \sum_k p(S_k) H(C,S_k), \qquad \text{where } p(S_k) = \frac{|S_k|}{|S|}$$

- ✔ The information gain of attribute A for a data set S is

$$Gain(A,S) = H(C,S) - H(C,S,A).$$

Choose the attribute with the highest information gain, i.e. the attribute with the lowest $H(C,S,A)$.

Choosing the test attribute (special case: binary classification)

✔ For a Boolean random variable V which is true with probability q, we can define:

$$H_B(q) = -q \log_2 q - (1-q) \log_2 (1-q)$$

✔ Entropy of the target class C measured on a data set S with N_p positive and N_n negative examples:

$$H(C,S) = H_B\left(\frac{N_p}{N_p + N_n}\right) = H_B\left(\frac{N_p}{|S|}\right)$$

Choosing the test attribute (example)

head		body		smile		neck		holds	
triangle:	2:1	triangle:	2:0	yes:	3:1	tie:	2:2	ball:	1:1
circle:	2:2	circle:	2:1	no:	1:3	bow:	0:2	sword:	0:2
square:	0:1	square:	0:3			nothing:	2:0	flower:	1:0
								nothing:	2:1

head:
$p(S_{\text{head=tri}}) = \frac{3}{8}$; $H(C,S_{\text{head=tri}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$p(S_{\text{head=cir}}) = \frac{4}{8}$; $H(C,S_{\text{head=cir}}) = H_B\left(\frac{2}{2+2}\right) = 1$
$p(S_{\text{head=sq}}) = \frac{1}{8}$; $H(C,S_{\text{head=sq}}) = H_B\left(\frac{0}{0+1}\right) = 0$
$H(C,S,head) = \frac{3}{8}\cdot 0.92 + \frac{4}{8}\cdot 1 + \frac{1}{8}\cdot 0 = 0.84$
$Gain(head,S) = 1 - 0.84 = 0.16$

body:
$p(S_{\text{body=tri}}) = \frac{2}{8}$; $H(C,S_{\text{body=tri}}) = H_B\left(\frac{2}{2+0}\right) = 0$
$p(S_{\text{body=cir}}) = \frac{3}{8}$; $H(C,S_{\text{body=cir}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$p(S_{\text{body=sq}}) = \frac{3}{8}$; $H(C,S_{\text{body=sq}}) = H_B\left(\frac{0}{0+3}\right) = 0$
$H(C,S,body) = \frac{2}{8}\cdot 0 + \frac{3}{8}\cdot 0.92 + \frac{3}{8}\cdot 0 = 0.35$
$Gain(body,S) = 1 - 0.35 = 0.65$

smile:
$p(S_{\text{smile=yes}}) = \frac{4}{8}$; $H(C,S_{\text{yes}}) = H_B\left(\frac{3}{3+1}\right) = 0.81$
$p(S_{\text{smile=no}}) = \frac{4}{8}$; $H(C,S_{\text{no}}) = H_B\left(\frac{1}{1+3}\right) = 0.81$
$H(C,S,smile) = \frac{4}{8}\cdot 0.81 + \frac{4}{8}\cdot 0.81 + \frac{3}{8}\cdot 0 = 0.81$
$Gain(smile,S) = 1 - 0.81 = 0.19$

neck:
$p(S_{\text{neck=tie}}) = \frac{4}{8}$; $H(C,S_{\text{neck=tie}}) = H_B\left(\frac{2}{2+2}\right) = 1$
$p(S_{\text{neck=bow}}) = \frac{2}{8}$; $H(C,S_{\text{neck=bow}}) = H_B\left(\frac{0}{0+2}\right) = 0$
$p(S_{\text{neck=no}}) = \frac{2}{8}$; $H(C,S_{\text{neck=no}}) = H_B\left(\frac{2}{2+0}\right) = 0$
$H(C,S,neck) = \frac{4}{8}\cdot 1 + \frac{2}{8}\cdot 0 + \frac{2}{8}\cdot 0 = 0.5$
$Gain(neck,S) = 1 - 0.5 = 0.5$

holds:
$p(S_{\text{holds=ball}}) = \frac{2}{8}$; $H(C,S_{\text{holds=ball}}) = H_B\left(\frac{1}{1+1}\right) = 1$
$p(S_{\text{holds=swo}}) = \frac{2}{8}$; $H(C,S_{\text{holds=swo}}) = H_B\left(\frac{0}{0+2}\right) = 0$
$p(S_{\text{holds=flo}}) = \frac{1}{8}$; $H(C,S_{\text{holds=flo}}) = H_B\left(\frac{1}{1+0}\right) = 0$
$p(S_{\text{holds=no}}) = \frac{3}{8}$; $H(C,S_{\text{holds=no}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$H(C,S,holds) = \frac{2}{8}\cdot 1 + \frac{2}{8}\cdot 0 + \frac{1}{8}\cdot 0 + \frac{3}{8}\cdot 0.92 = 0.6$
$Gain(holds,S) = 1 - 0.6 = 0.4$

The **body** attribute

✔ brings us the largest information gain, thus
✔ it shall be chosen for the first test in the tree!

Choosing subsequent test attribute

No further tests are needed for robots with triangular and squared bodies.
Dataset for robots with circular bodies:

head	body	smile	neck	holds	class
triangle	circle	yes	tie	nothing	ally
circle	circle	yes	tie	nothing	ally
circle	circle	no	bow	sword	enemy

triangle:	1:0	yes:	2:0	tie:	2:0	nothing:	2:0	
circle:	1:1	no:	0:1	bow:	0:1	sword:	0:1	

All the attributes **smile**, **neck**, and **holds**

- ✔ take up the remaining entropy in the data set, and
- ✔ are equally good for the test in the group of robots with circular bodies.

Decision tree building procedure

Algorithm 1: BuildDT

Input : the set of examples S,
the set of attributes $\mathcal{A}$,
majority class of the parent node C_P
Output: a decision tree

```
1  begin
2      if S is empty then return leaf with C_P
3      C ← majority class in S
4      if all examples in S belong to the same class C then return leaf with C
5      if A is empty then return leaf with C
6      A ← arg max_{a∈A} Gain(a, S)
7      T ← a new decision tree with root test on attribute A
8      foreach value v_k of A do
9          S_k ← {x | x ∈ S ∧ x.A = v_k}
10         t_k ← BuildDT(S_k, A − A, C)
11         add branch to T with label A = v_k and attach a subtree t_k
12     return tree T
```

Algorithm characteristics

- ✔ There are many hypotheses (trees) consistent with the dataset S; the algorithm will return any of them, unless there is some bias in choosing the tests.
- ✔ The current set of considered hypotheses has always only 1 member (greedy selection of the successor). The algorithm cannot provide answer to the question how many hypotheses consistent with the data exist.
- ✔ The algorithm does not use backtracking; it can get stuck in a local optimum.
- ✔ The algorithm uses batch learning, not incremental.

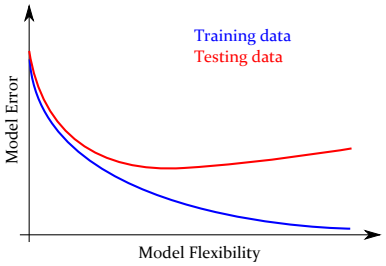
Generalization and Overfitting 19 / 29

Overfitting

Definition of overfitting:

- ✔ Let H be a hypothesis space.
- ✔ Let $h \in H$ and $h' \in H$ be 2 different hypotheses from this space.
- ✔ Let $\mathrm{Err}_{\mathrm{Tr}}(h)$ be an error of the hypothesis h measured on the training dataset (training error).
- ✔ Let $\mathrm{Err}_{\mathrm{Tst}}(h)$ be an error of the hypothesis h measured on the testing dataset (testing error).
- ✔ We say that h is overfitted if there is another h' for which

$$\mathrm{Err}_{\mathrm{Tr}}(h) < \mathrm{Err}_{\mathrm{Tr}}(h') \wedge \mathrm{Err}_{\mathrm{Tst}}(h) > \mathrm{Err}_{\mathrm{Tst}}(h')$$



- ✔ "Overfitting is a situation when the model works well for the training data, but fails for new (testing) data."
- ✔ Overfitting is a general phenomenon related to all kinds of inductive learning (i.e. it applies to all models, not only trees).

We want models and learning algorithms with a good **generalization ability**, i.e.

- ✔ we want models that encode **only the patterns valid in the whole domain**, not those that learned the specifics of the training data,
- ✔ we want algorithms able to find **only the patterns valid in the whole domain** and ignore specifics of the training data.

How to prevent overfitting for trees?

Tree pruning:

- ✔ Let's have a fully grown tree T.
- ✔ Choose a test node having only leaf nodes as descensdants.
- ✔ If the test appears to be irrelevant, remove the test and replace it with a leaf node with the majority class.
- ✔ Repeat, until all tests seem to be relevant.

How to check if the split is (ir)relevant?

1. Using statistical χ^2 test:
   - ✔ If the distribution of classes in the leaves does not differ much from the distribution of classes in their parent, the split is irrelevant.
2. Using an (independent) validation data set:
   - ✔ Create a temporary tree by replacing a subtree with a leaf.
   - ✔ If the error on validation set decreased, accept the pruned tree.

Early stopping:

- ✔ Hmm, if we grow the tree fully and then prune it, why cannot we just stop the tree building when there is no good attribute to split on?
- ✔ Prevents us from recognizing situations when
  - ✘ there is no single good attribute to split on, but
  - ✘ there are combinations of attributes that lead to a good tree!

Broadening the Applicability of Desicion Trees 22 / 29

Missing data

Decision trees are one of the rare model types able to handle missing attribute values.

1. Given a complete tree, how to classify an example with a missing attribute value needed for a test?
   - ✔ Pretend that the object has all possible values for this attribute.
   - ✔ Track all possible paths to the leaves.
   - ✔ The leaf decisions are weighted using the number of training examples in the leaves.
2. How to build a tree if the training set contains examples with missing attribute values?
   - ✔ Introduce a new attribute value: "Missing" (or N/A).
   - ✔ Build tree in a normal way.

Multivalued attributes

What if the training set contains e.g. name, social insurance number, or other id?

- ✔ When each example has a unique value of an attribute A, the information gain of A is equal to the entropy of the whole data set!
- ✔ Attribute A is chosen for the tree root; yet, such a tree is useless (overfitted).

Solutions:

1. Allow only Boolean test of the form $A = v_k$ and allow the remaining values to be tested later in the tree.
2. Use a different split importance measure instead of *Gain*, e.g. *GainRatio*:
   - ✔ Normalize the information gain by a maximal amount of information the split can have:

$$GainRatio(A,S) = \frac{Gain(A,S)}{H(A,S)},$$

   where $H(A,S)$ is the entropy of attribute A and represents the largest information gain we can get from splitting using A.

Attributes with different prices

What if the tests in the tree are also cost something?

- ✔ Then we would like to have the cheap test close to the root.
- ✔ If we have $Cost(A) \in \langle 0, 1\rangle$ then we can use e.g.

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

or

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

to bias the preference for cheaper tests.

Continuous input attributes

Continuous or integer-valued input attributes:

- ✔ have an infinite set of possible values. (Infinitely many branches?)
- ✔ Use a binary split with the highest information gain.
  - ✘ Sort the values of the attribute.
  - ✘ Consider only split points lying between 2 examples with different classification.

Temperature	-20	-9	-2	5	16	26	32	35
Go out?	No	No	Yes	Yes	Yes	Yes	No	No

- ✔ Previously used **attributes can be used again** in subsequent tests!

Continuous output variable

Regression tree:

- ✔ In each leaf, it can have
  - ✘ a **constant value** (usually an average of the output variable over the training set), or
  - ✘ a **linear function** of some subset of numerical input attributes
- ✔ The learning algorithm must decide when to stop splitting and begin applying linear regression.

Summary

- ✔ Decision trees are one of the simplest, most universal and most widely used prediction models.
- ✔ They are not suitable for all modeling problems (relations, etc.).
- ✔ TDIDT is the most widely used technique to build a tree from data.
- ✔ It uses greedy divide-and-conquer approach.
- ✔ Individual variants differ mainly
  - ✘ in what type of attributes they are able to handle,
  - ✘ in the attribute importance measure,
  - ✘ if they make enumerative or just binary splits,
  - ✘ if and how they can handle missing data,
  - ✘ etc.