

AI designs a forest

Decision trees

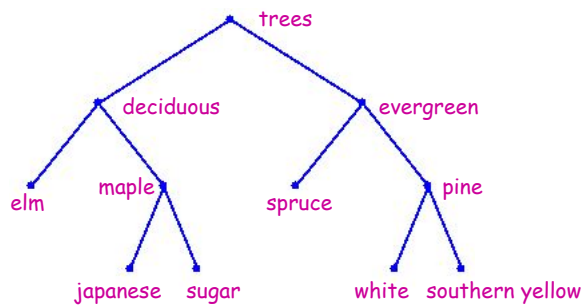


CS112 Scientific Computation

Department of Computer Science
Wellesley College

Phylogenetic tree of trees

```
philo = {'trees' {'deciduous' {'elm' {} {} } ...  
                {'maple' {'japanese' {} {} } ...  
                    {'sugar' {} {} } ... } ...  
        {'evergreen' {'spruce' {} {} } ...  
                    {'pine' {'white' {} {} } ...  
                        {'southern yellow' {} {} } ... } ...};
```



Think recursively!

If the tree is empty

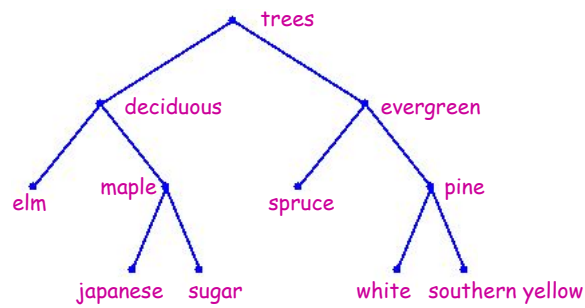
do nothing

Otherwise

print the string in the root node

print the contents of the left subtree

print the contents of the right subtree



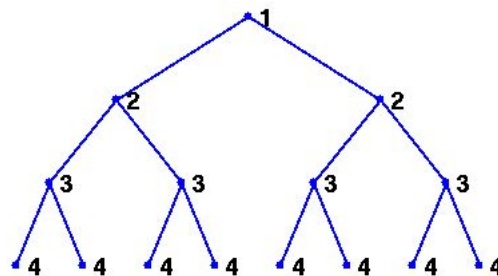
Decision trees 24-3

Growing trees recursively

Example: Create a binary tree that

- has n levels
- has all possible nodes on each level (a full binary tree)
- stores the level of the nodes (as a string) in each node

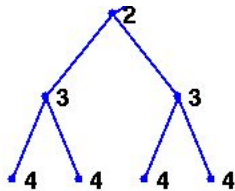
>> tree4 = levelTree(4);



Decision trees 24-4

levelTree's helper function

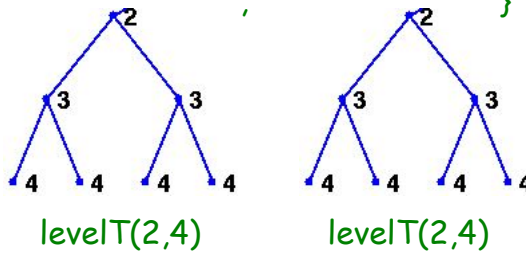
- We use a recursive helper function:
`levelT(root-level, leaf-level)`
- For example,
`>> smallTree = levelT(2, 4)`



Decision trees 24-5

Creating levelT(1,4) from levelT(2,4)

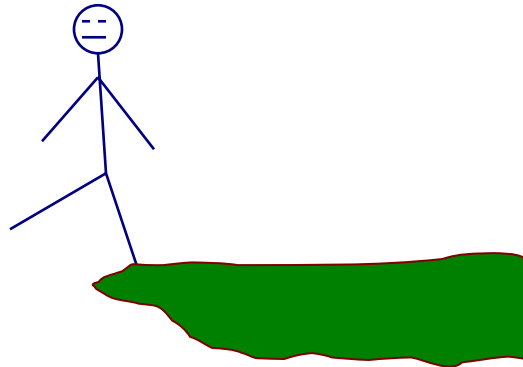
`tree = {'1',`



Decision trees 24-6

Recursive leap of faith

Assume we have a working function called `levelT` and then use it to write a function called `levelT` ...



Decision trees 24-7

Recursive tree growth

```
function tree = levelT (rootLevel, leafLevel)
% recursive function that creates a full subtree containing
% (leafLevel-rootLevel+1) levels, with rootLevel in the root
% node and leafLevel in all of the leaf nodes
if                                     % base case
else                                   % recursive step

end
```

Decision trees 24-8

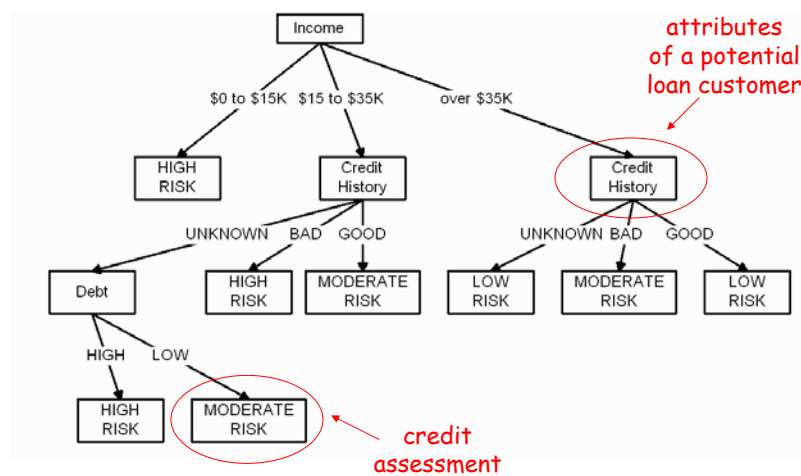
Recursive tree growth

```
function tree = levelT (rootLevel, leafLevel)
% recursive function that creates a full subtree containing
% (leafLevel-rootLevel+1) levels, with rootLevel in the root
% node and leafLevel in all of the leaf nodes

if (rootLevel == leafLevel)           % base case
    tree = {num2str(leafLevel) {} {}};
else                                   % recursive step
    tree = {num2str(rootLevel) ...
            levelT(rootLevel+1, leafLevel) ...
            levelT(rootLevel+1, leafLevel)};
end
```

Decision trees 24-9

Decisions, decisions



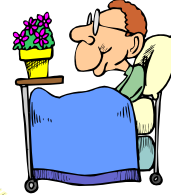
Decision trees 24-10

Data Mining with decision trees

Medicine: What health factors result in increased risk for certain diseases or other health problems?


Economics: What social, political, or financial factors contribute most directly to certain economic indicators?

Geology: What physical factors best predict catastrophic events such as earthquakes and volcanoes?



Decision trees 24-11

Medical Decision Making



ELSEVIER Artificial Intelligence in Medicine 19 (2000) 189–202
www.elsevier.com/locate/artmed

Artificial Intelligence in Medicine

Multiple signal integration by decision tree induction to detect artifacts in the neonatal intensive care unit

Christine L. Tsien ^{a,b,*}, Isaac S. Kohane ^{b,c}, Neil McIntosh ^d

^a MIT Laboratory for Computer Science, 545 Technology Square, NE43-420, Cambridge, MA 02139, USA
^b Harvard Medical School, 260 Longwood Avenue, Boston, MA 02115, USA
^c Children's Hospital Informatics Program, Children's Hospital, 300 Longwood Avenue, Boston, MA 02115, USA
^d Neonatal Unit, Simpson Memorial Maternity Pavilion, Royal Infirmary, Lauriston Place, Edinburgh EH1, Scotland, UK

Decision trees 24-12

How do we create a decision tree?

- The MATLAB statistics toolbox contains software for working with decision trees

... but we'll build our own!



It is vain to do with more what can be done with less... Entities should not be multiplied beyond necessity

-- William of Occam, 1324

Decision trees 24-13

Building a decision tree

- Suppose we're given the health data shown in the table, where all attributes are either true (1) or false (0)
- We can use this data to create a decision tree that classifies new individuals as being at risk for developing this health condition or not

sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1

Decision trees 24-14

First we ask ...

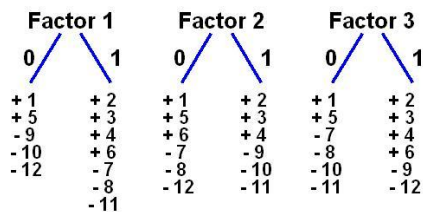
... which of the attributes
best separates the samples
 into individuals **with** the
 condition and those **without**?

sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1

Decision trees 24-15

1. Use each attribute ...

... to divide samples into two groups
 that have the value 0 or 1 for this
 attribute and label the samples as
 having the condition (+) or not (-)



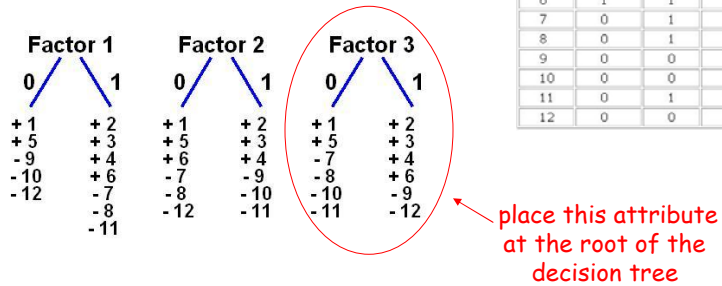
sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1

Decision trees 24-16

2. Select the attribute ...

... that creates the **most homogeneous subgroups** and place this attribute at the root of the decision tree

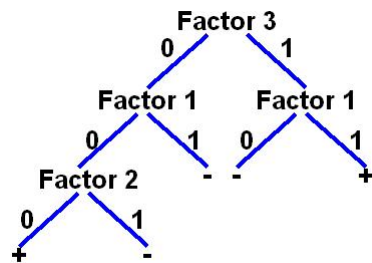
sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1



Decision trees 24-17

3. Use remaining factors ...

... to further segregate the samples*



sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1

* Think recursively!

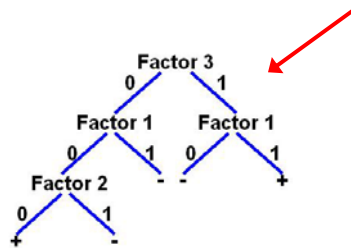
Decision trees 24-18

The programming goal

Input: data table with attributes and their values, and a target attribute for classifying each data sample

sample number	condition present?	health factor 1	health factor 2	health factor 3
1	1	0	0	0
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	0	0
6	1	1	0	1
7	0	1	0	0
8	0	1	0	0
9	0	0	1	1
10	0	0	1	0
11	0	1	1	0
12	0	0	0	1

Output: decision tree



Decision trees 24-19

Our strategy

If there are no remaining samples to classify
then return an empty tree

If no attributes left to classify samples
then return a tree with the most common value for the
target attribute

Otherwise

1. Determine which attribute separates the samples into the most homogeneous subgroups
2. Separate the samples into two subgroups using the best attribute

Decision trees 24-20

Attaching subtrees

If the group of samples with attribute value 0 is homogeneous,
then assign the left subtree to the value of the target attribute

Otherwise

assign left subtree to the decision tree constructed from this
subgroup of samples and remaining attributes

If the group of samples with attribute value 1 is homogeneous,
then assign the right subtree to the value of the target attribute

Otherwise

assign the right subtree to the decision tree constructed
from this subgroup of samples and remaining attributes

Return a final tree composed of the best attribute, left subtree
and right subtree

Decision trees 24-21

One missing piece

- How do we measure the homogeneity of the two groups of data samples created by each attribute?
- Borrow a formula from information theory for measuring average entropy of groups of samples:

average disorder =

$$\sum_b [(n_b / n_t) \times (\sum_c -(n_{bc} / n_b) \log_2(n_{bc} / n_b))]$$

where

- n_b is the number of samples in branch b
- n_t is the total number of samples in all branches
- n_{bc} is the total number of samples in branch b of class c



Decision trees 24-22