Decision Support

Dr. Johan Hagelbäck

johan.hagelback@lnu.se

http://aiguy.org

Decision Support

- One of the earliest AI problems was decision support
- The first solution to this problem was expert systems
- They used an often very large number of hand-crafted *if-then* rules
- These problems are suitable for a type of algorithms called Decision Trees
- The dataset typically mostly contains categorical features, but can have numerical features as well.

Decision Trees

- Decision Trees has one big advantage: the trained model is easy to visualize and interpret
- We can understand what the algorithm has learned
- This can be important in some applications where we want to investigate why the system took a decision
- This is commonly referred to as a completely transparent method

Example: Weather dataset

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
overcast	hot	high	false	YES
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
overcast	cool	normal	true	YES
sunny	mild	high	false	NO
sunny	cool	normal	false	YES
rainy	mild	normal	false	YES
sunny	mild	normal	true	YES
overcast	mild	high	true	YES
overcast	hot	normal	false	YES
rainy	mild	high	true	NO

Building the tree

- At each node, we need to find the attribute that best divides the data into Yes and No.
- To do this we calculate the information gain for each parameter and value.
- The attribute with the highest information gain is selected at each node.

$$Gain(A) = 1 - \left(\sum_{i=1}^{v} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)\right)$$

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = \frac{p+n}{p_{tot} + n_{tot}} \left(-\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}\right)$$

Find the root node

Outlook	Sunny	Overcast	Rainy
Yes	2	4	3
No	3	0	2

$$I(sunny) = \frac{5}{14}\left(-\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5}\right) \approx 0.347$$

$$I(overcast) = \frac{4}{14}\left(-\frac{4}{4}log_2\frac{4}{4} - 0\right) = 0$$

$$I(rainy) = \frac{5}{14}\left(-\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}\right) \approx 0.347$$

$$Gain(outlook) \approx 1 - 0.347 - 0 - 0.347 = \boxed{0.306}$$

Find the root node

Temperature	Hot	Mild	Cool
Yes	2	4	3
No	2	2	1

$$I(hot) = \frac{4}{14}\left(-\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4}\right) \approx 0.286$$

$$I(mild) = \frac{6}{14}\left(-\frac{4}{6}log_2\frac{4}{6} - \frac{2}{6}log_2\frac{2}{6}\right) \approx 0.394$$

$$I(cool) = \frac{4}{14}\left(-\frac{3}{4}log_2\frac{3}{4} - \frac{1}{4}log_2\frac{1}{4}\right) \approx 0.232$$

$$Gain(temperature) \approx 1 - 0.286 - 0.394 - 0.232 = \boxed{0.088}$$

Find the root node

Humidity	High	Normal
Yes	3	6
No	4	1

$$I(high) = \frac{7}{14}\left(-\frac{3}{7}log_2\frac{3}{7} - \frac{4}{7}log_2\frac{4}{7}\right) \approx 0.493$$

$$I(normal) = \frac{7}{14}\left(-\frac{6}{7}log_2\frac{6}{7} - \frac{1}{7}log_2\frac{1}{7}\right) \approx 0.296$$

$$Gain(humidity) \approx 1 - 0.493 - 0.296 = \boxed{0.211}$$

Find the root node

Windy	True	False
Yes	3	6
No	3	2

$$I(true) = \frac{6}{14}\left(-\frac{3}{6}log_2\frac{3}{6} - \frac{3}{6}log_2\frac{3}{6}\right) \approx 0.429$$

$$I(false) = \frac{8}{14}\left(-\frac{6}{8}log_2\frac{6}{8} - \frac{2}{8}log_2\frac{2}{8}\right) \approx 0.464$$

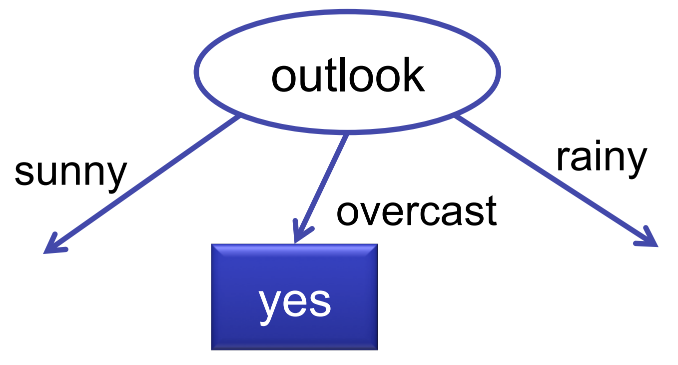
$$Gain(windy) \approx 1 - 0.429 - 0.464 = \boxed{0.107}$$

Find the root node

Attribute	Gain
Outlook	0.306
Temperature	0.088
Humidity	0.211
Windy	0.107

Outlook has the highest gain and is selected as root node

Find the root node



Overcast has perfect gain = all examples belongs to the same category: Yes

Let’s find the sunny node!

All examples with sunny

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
sunny	mild	high	false	NO
sunny	cool	normal	false	YES
sunny	mild	normal	true	YES

- Now we use a subset of the data
- It contains all examples with Outlook = sunny
- 5 examples

Find the sunny node

Temperature	Hot	Mild	Cool
Yes	0	1	1
No	2	1	0

$$I(hot) = 0$$

$$I(mild) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(cool) = 0$$

$$Gain(temperature) = 1 - 0 - 0.4 - 0 = \boxed{0.6}$$

Find the sunny node

Humidity	High	Normal
Yes	0	2
No	3	0

$$I(high) = 0$$

$$I(normal) = 0$$

$$Gain(humidity) = 1 - 0 - 0 = \boxed{1}$$

Find the sunny node

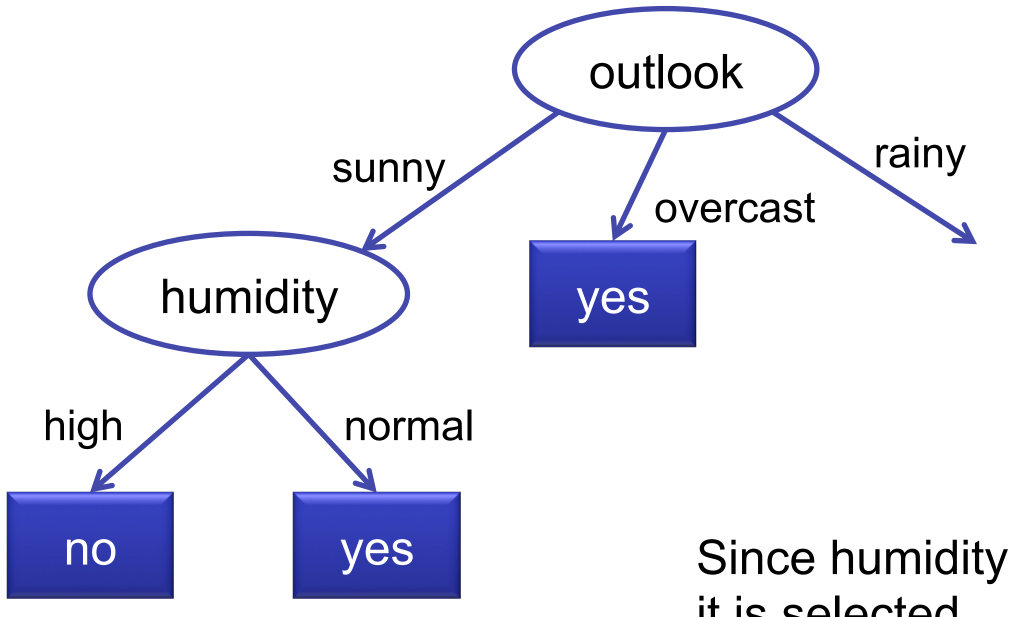
Windy	True	False
Yes	1	1
No	1	2

$$I(true) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(false) = \frac{3}{5}\left(-\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}\right) \approx 0.551$$

$$Gain(windy) \approx 1 - 0.4 - 0.551 = \boxed{0.049}$$

Find the sunny node



Since humidity has perfect gain
it is selected

Let's find the rainy node!

All examples with rainy

Outlook	Temperature	Humidity	Windy	Play
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
rainy	mild	normal	false	YES
rainy	mild	high	true	NO

- Again, we use a subset of the data
- It contains all examples with Outlook = rainy
- 5 examples

Find the rainy node

Temperature	Hot	Mild	Cool
Yes	0	2	1
No	0	1	1

$$I(hot) = 0$$

$$I(mild) = \frac{3}{5}\left(-\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3}\right) \approx 0.551$$

$$I(cool) = \frac{2}{5} \cdot 1 = 0.4$$

$$Gain(temperature) \approx 1 - 0 - 0.551 - 0.4 = \boxed{0.049}$$

Find the rainy node

Humidity	High	Normal
Yes	1	2
No	1	1

$$I(high) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(normal) = \frac{3}{5}\left(-\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3}\right) \approx 0.551$$

$$Gain(humidity) \approx 1 - 0.4 - 0.551 = \boxed{0.049}$$

Find the rainy node

Windy	True	False
Yes	0	3
No	2	0

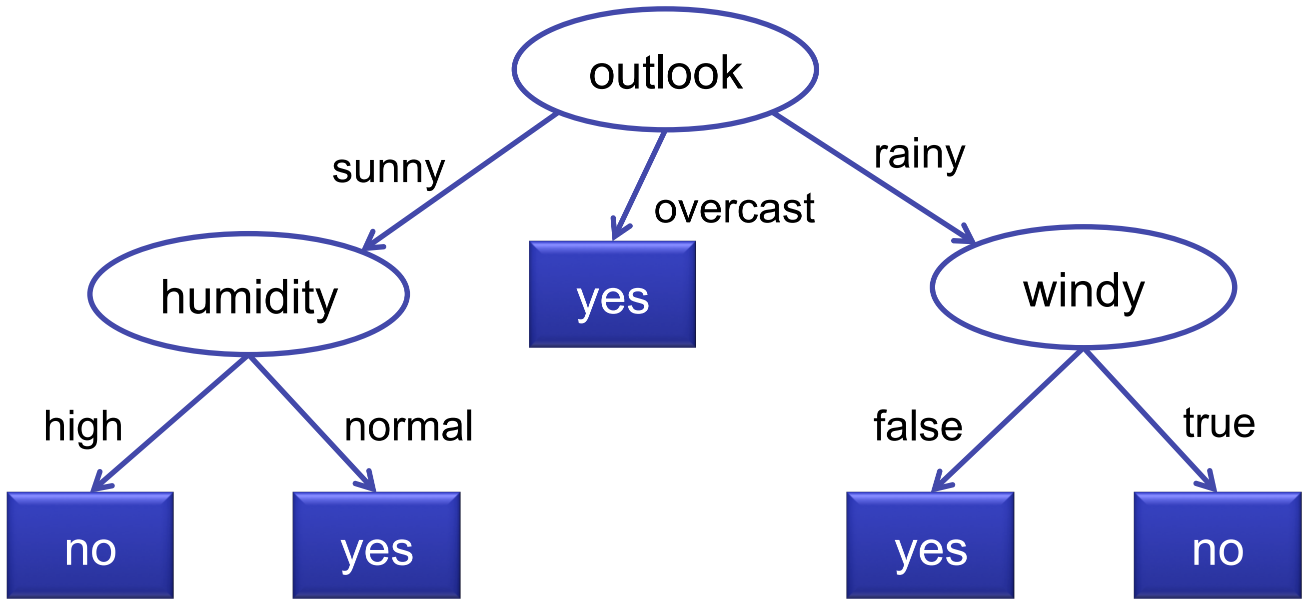
$$I(true) = 0$$

$$I(false) = 0$$

$$Gain(windy) = 1 - 0 - 0 = \boxed{1}$$

Since windy has perfect gain, it is selected

Final tree


outlook
sunny
overcast
rainy
humidity
yes
windy
high
normal
false
true
no
yes
yes
no

The problem

- In most cases, there are several possible trees that can be generated
- The aim is to:
  1. Generate a tree that as accurately as possible can classify the training data
  2. Generate the smallest possible tree
- It can be tricky to satisfy both
- The first is of highest priority

Generating a good tree

- There is a wide range of different algorithms for generating decision trees
- Each tries to fulfill both criteria as much as possible
- Weka uses an algorithm called J48

Classification

- To classify an example, we need to traverse the tree by following the nodes that matches the attribute values in the example
- When we reach a leaf node, the result (category) is returned

Overfitting

- Decision Trees can suffer from *overfitting*
- It means that the model learned is very specific to the training data, but can be bad at classifying unknown examples
- To get around this problem, learning is usually stopped before there is a risk of *overfitting*

Overfitting

- A common approach to reduce *overfitting* in Decision Trees is to stop creating more branches if there is only a very small increase in gain
- We can set a minimum threshold of how large the gain must be to allow a new branch to be created
- There is no universal answer to which limit to use
- You have to experiment on the dataset you use

When to use Decision Trees

- As mentioned, one big advantage of DTs is that we can interpret the trained model
- There are some other benefits of DTs
- They work on both numerical and nominal attributes without pre-processing the data, which many other algorithms don’t
- They also support probabilistic reasoning of assignments, which we did when we returned the most probable category

When to use Decision Trees

- The major drawback is that DTs are not very good for complex learning problems
- If we have lots of categories, the decision tree tends to be very complicated and will most likely make poor predictions
- Another disadvantage is that they can only do simple greater-than/less-than decisions for numerical attributes
- They work best if we have combinations of numerical and nominal data, and few categories (which many real-world problems satisfy)

Weka

- Weka's standard Decision Tree classifier is called J48.
- When using J48 on the Weather dataset we get the following result:

Classifier output

```
Correctly Classified Instances          9               64.2857 %
Incorrectly Classified Instances        5               35.7143 %
Kappa statistic                         0.186
Mean absolute error                     0.2857
Root mean squared error                 0.4818
Relative absolute error                60      %
Root relative squared error            97.6586 %
Total Number of Instances              14
```

R

- In R, we can use an algorithm called CART
- The dataset needs to be in csv format
- The R script looks like this:

R script

```r
#Load the ML library
library(caret)

#Read the dataset
dataset <- read.csv("FIFA_skill.csv")

#setup 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"

#Train model using CART
set.seed(7)
cart <- train(PlayerSkill~., data=dataset, method="rpart",
                metric=metric, trControl=control)

#Print result
print(cart)
```

R result

```
Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  :
  There were missing values in resampled performance measures.
>
> #Print result
> print(cart)
CART

19 samples
 3 predictor
 2 classes: 'bad', 'good'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 17, 17, 17, 17, 17, 17, ...
Resampling results:

  Accuracy  Kappa
  0.55      0
```

R result

- The warning message from R means that the 10-fold CV split the dataset so one class was missing in some iteration
- This has large impact on the result
- R needs more data to accurately predict the dataset
- If we make a copy of each example in the dataset (twice as much data), the result is:

R result

```
CART

38 samples
 3 predictor
 2 classes: 'bad', 'good'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 35, 34, 34, 34, 34, 34, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.0000000  0.8416667  0.69
  0.3333333  0.8416667  0.69
  0.6666667  0.6083333  0.19

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.3333333.
```

Decision Support

Dr. Johan Hagelbäck

johan.hagelback@lnu.se

http://aiguy.org