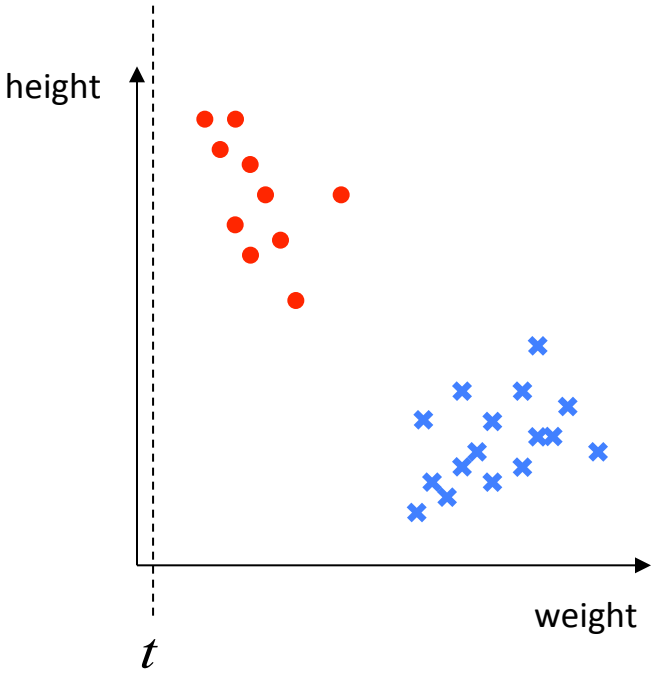
COM24111: Machine Learning

Decision Trees

Gavin Brown

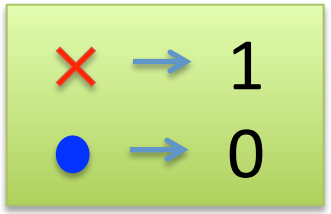
www.cs.man.ac.uk/~gbrown

Recap: threshold classifiers



$$\text{if } (weight > t) \text{ then "player" else "dancer"}$$

Also known as “decision stump”



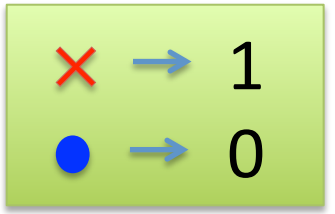
`if` $x > t$ `then` $\hat{y} = 1$ `else` $\hat{y} = 0$

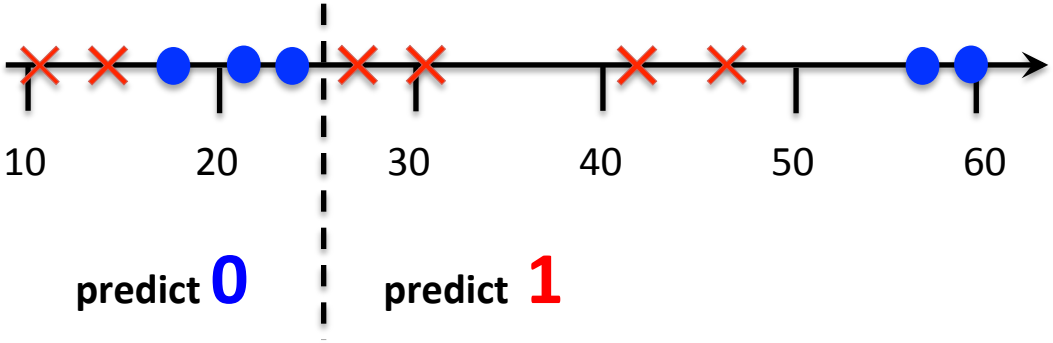
Q. Where is a good threshold?

From Decision Stumps, to Decision Trees

- New type of **non-linear model**
- Copes naturally with continuous and **categorical data**
- **Fast** to both train and test (highly parallelizable)
- Generates a set of **interpretable** rules

Recap: Decision Stumps

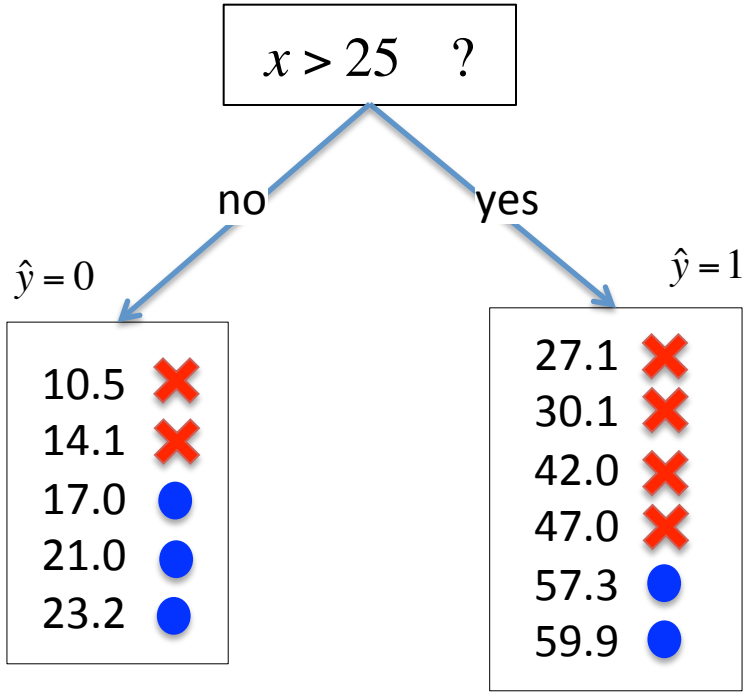




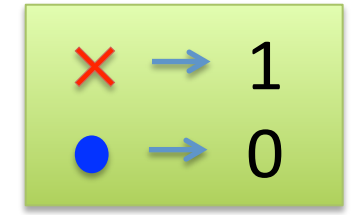
$$\texttt{if } x > t \texttt{ then } \hat{y} = 1 \texttt{ else } \hat{y} = 0$$

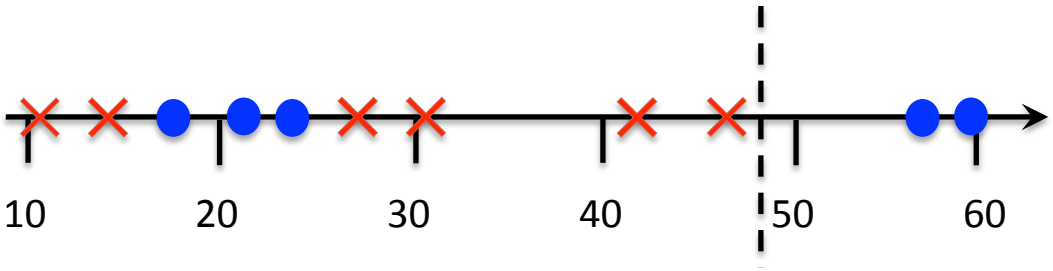
The stump "splits" the dataset.

Here we have 4 classification errors.



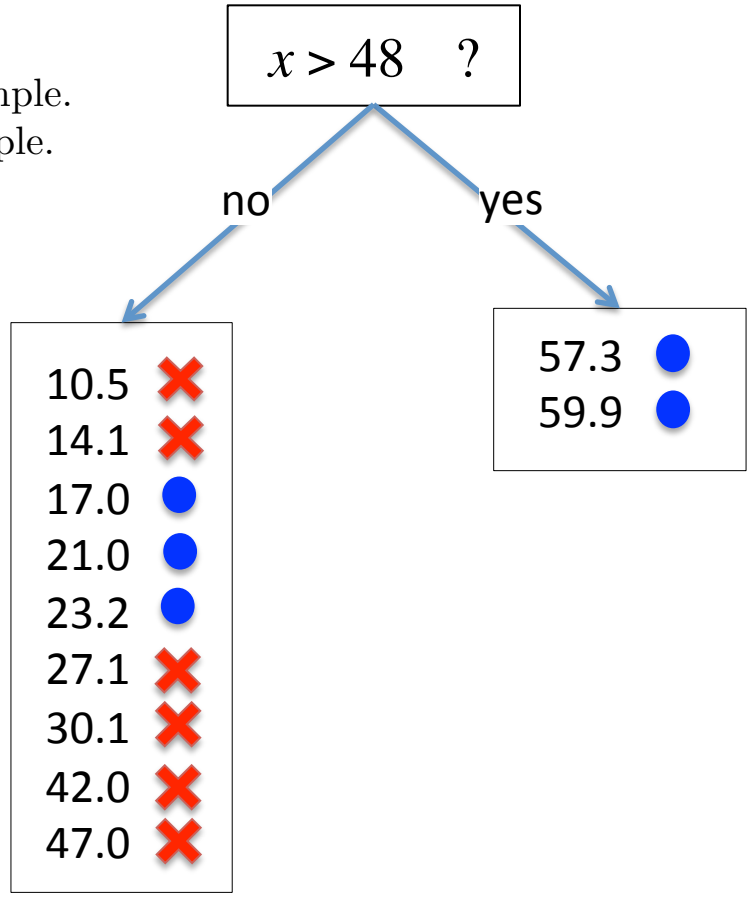
A modified stump





Set y_{right} to the most common label in the $(> t)$ subsample.
Set y_{left} to the most common label in the $(< t)$ subsample.

```
if x > t  then
    predict ŷ = y_right
else
    predict ŷ = y_left
endif
```

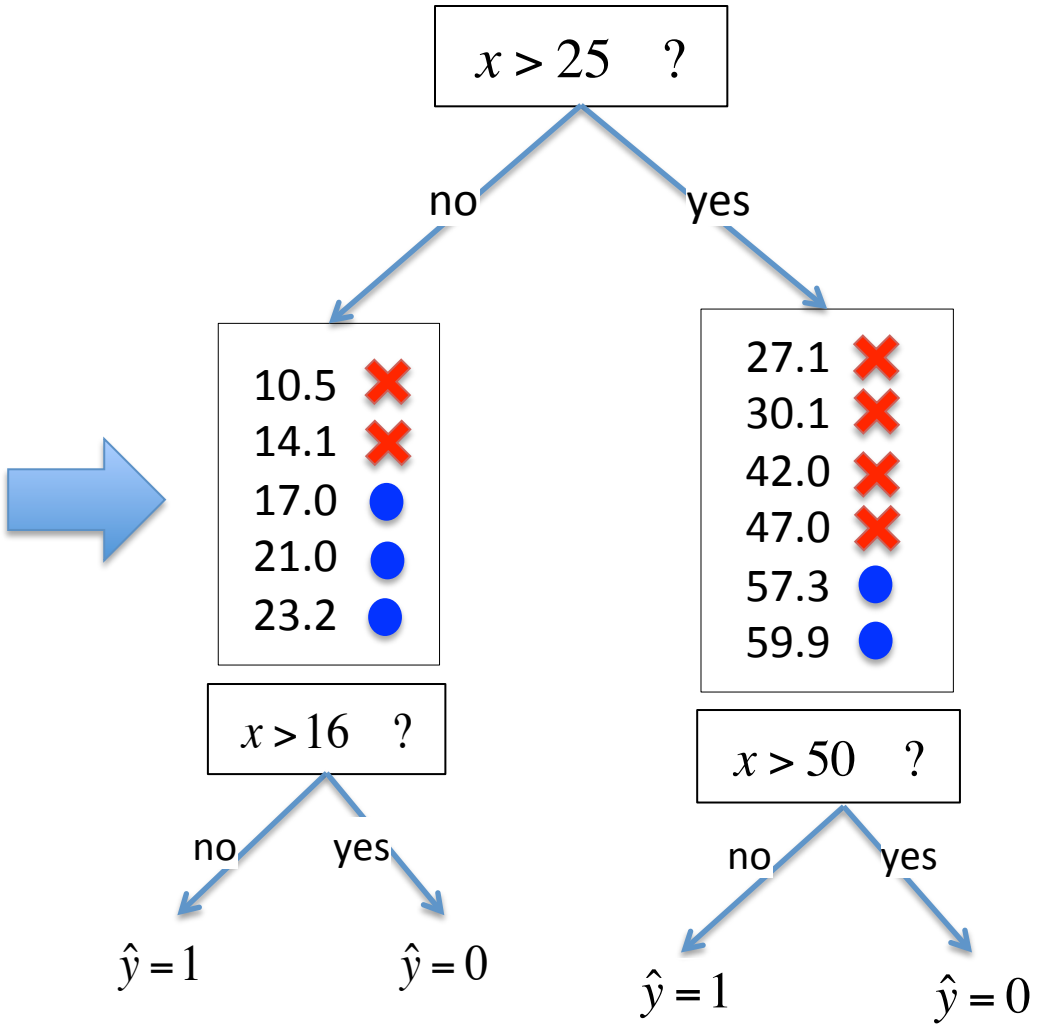


Here we have 3 classification errors.

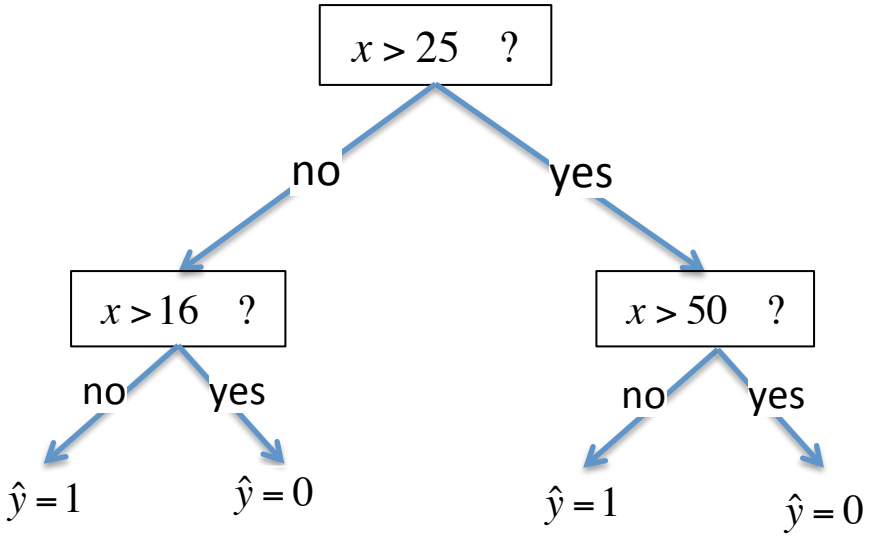
Recursion…

Just another dataset!

Build a stump!



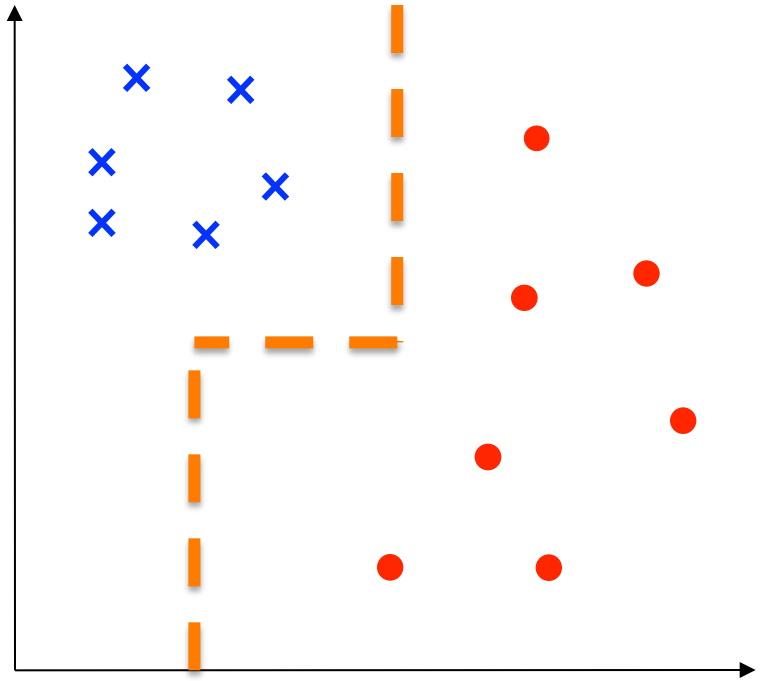
Decision Trees = nested rules



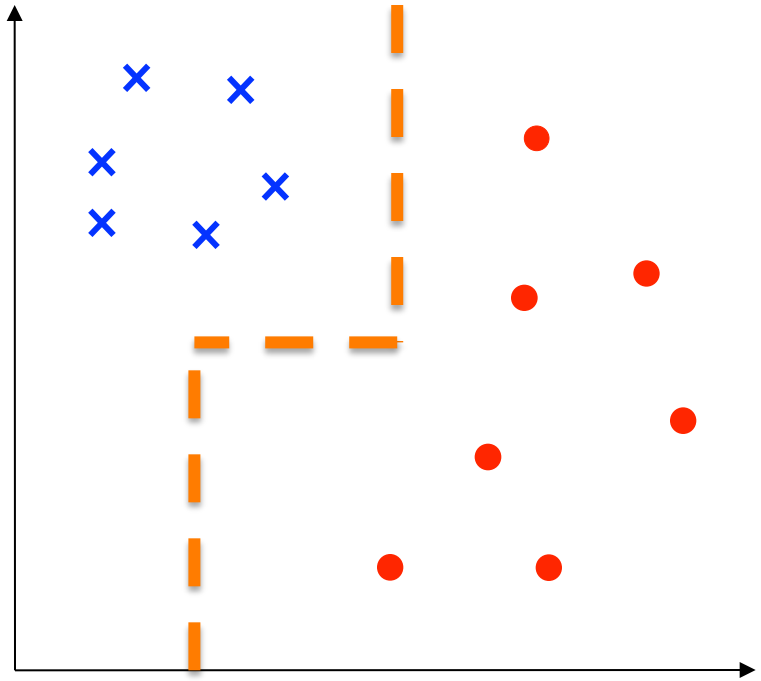

```
if x>25 then
    if x>50 then y=0 else y=1; endif
else
    if x>16 then y=0 else y=1; endif
endif
```

Trees build “orthogonal” decision boundaries.

Boundary is piecewise, and at 90 degrees to feature axes.

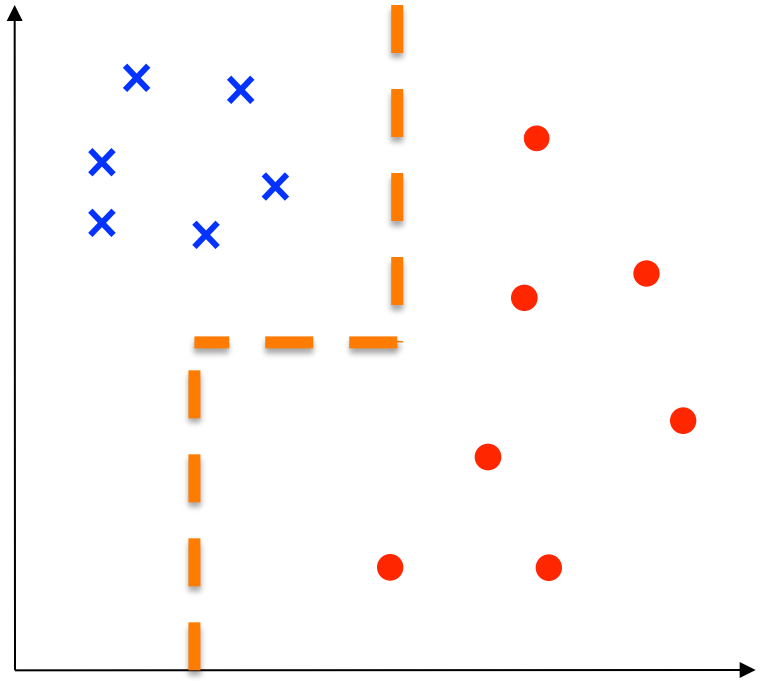


The *most important* concept in Machine Learning

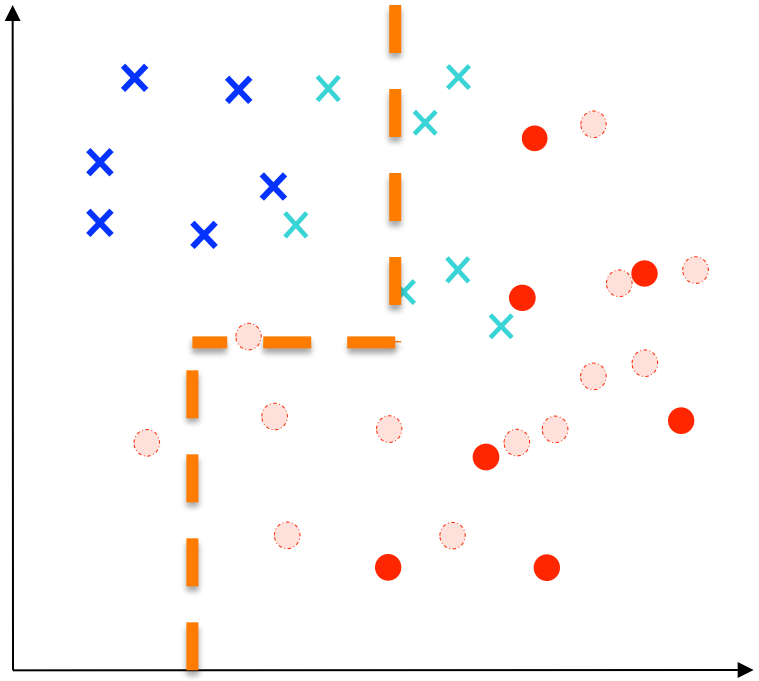


The *most important* concept in Machine Learning

Looks good so far…



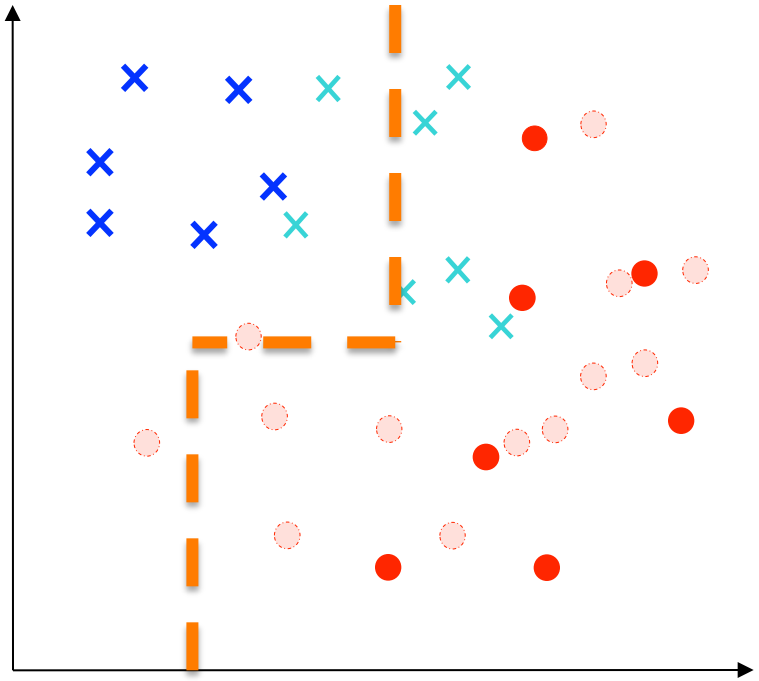
The *most important* concept in Machine Learning



Looks good so far…

Oh no! Mistakes!
What happened?

The *most important* concept in Machine Learning



Looks good so far…

Oh no! Mistakes!
What happened?

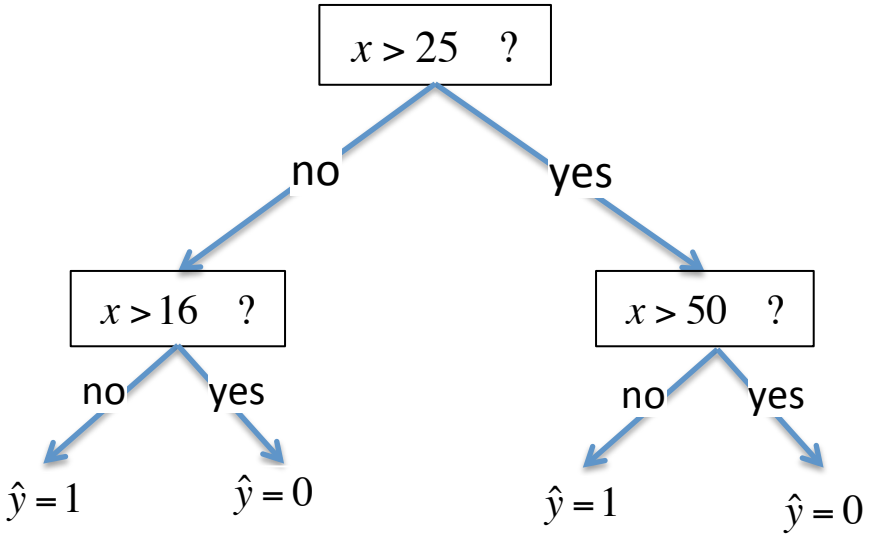
We didn't have all the data.

We can never assume that we do.

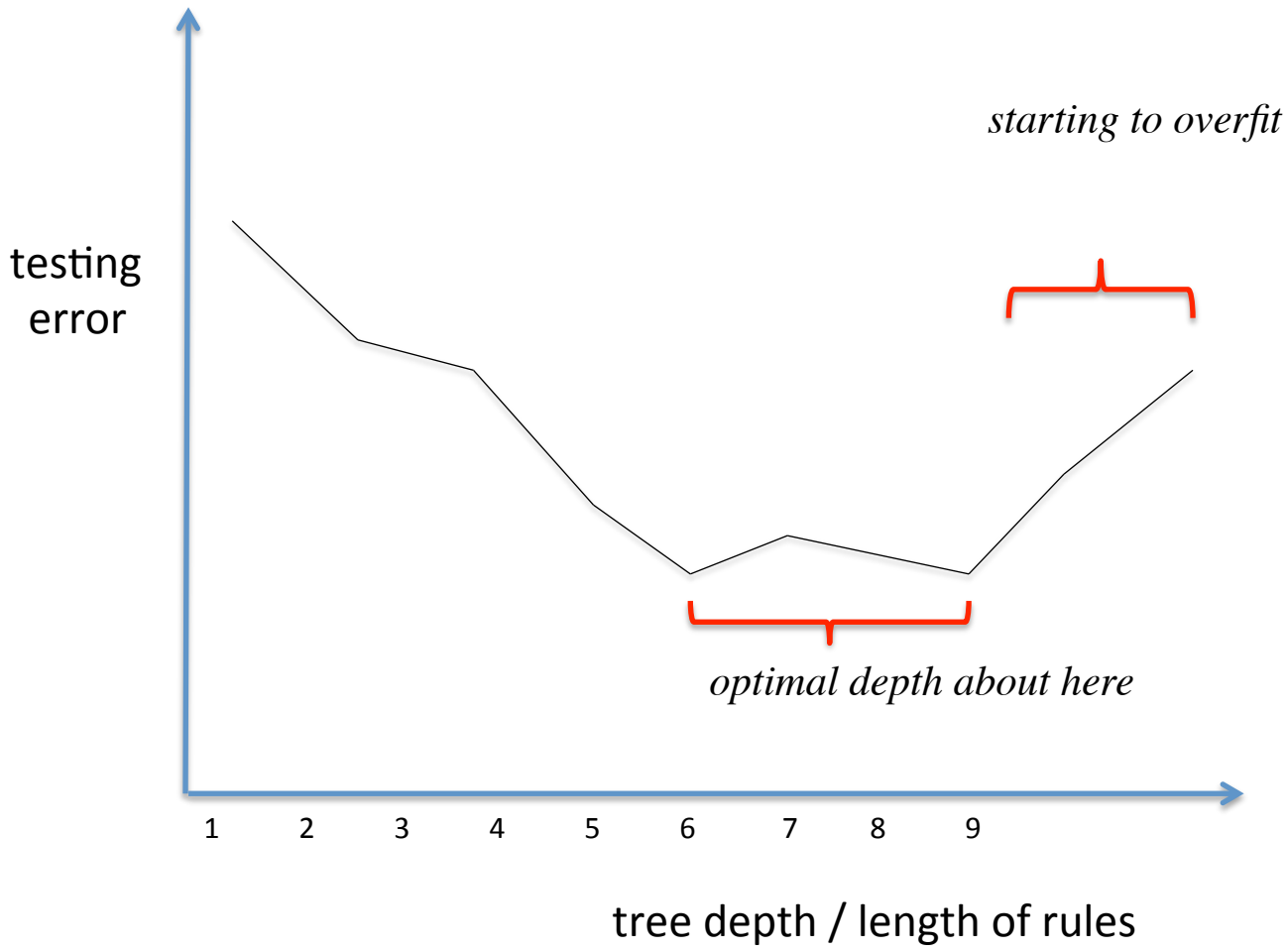
This is called "OVER-FITTING" to the small dataset.

Number of possible paths down tree tells you the number of rules.
More rules = more complicated.

Could have N rules, where N is the num of examples in training data.
… the more rules… the more chance of **OVERFITTING.**



Overfitting….



Decision Tree Learning Algorithm (sometimes called "ID3")

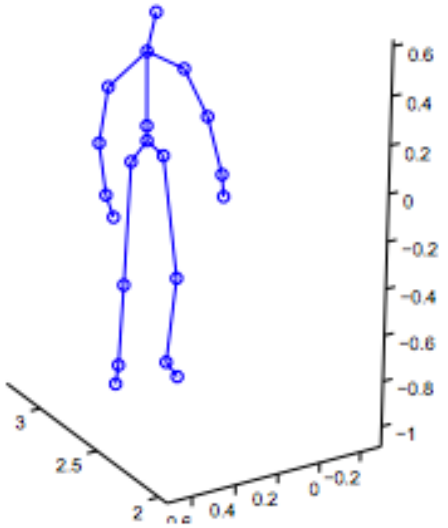
```
1: function BUILDTREE( subsample, depth )
2:
3:     //BASE CASE:
4:     if (depth == 0) OR (all examples have same label) then
5:         return most common label in the subsample
6:     end if
7:
8:     //RECURSIVE CASE:
9:     for each feature do
10:        Try splitting the data (i.e. build a decision stump)
11:        Calculate the cost for this stump
12:    end for
13:    Pick feature with minimum cost
14:
15:    Find left/right subsamples
16:    Add left branch ← BUILDTREE( leftSubSample, depth − 1 )
17:    Add right branch ← BUILDTREE( rightSubSample, depth − 1 )
18:
19:    return tree
20:
21: end function
```

Take a short break.

Talk to your neighbours.

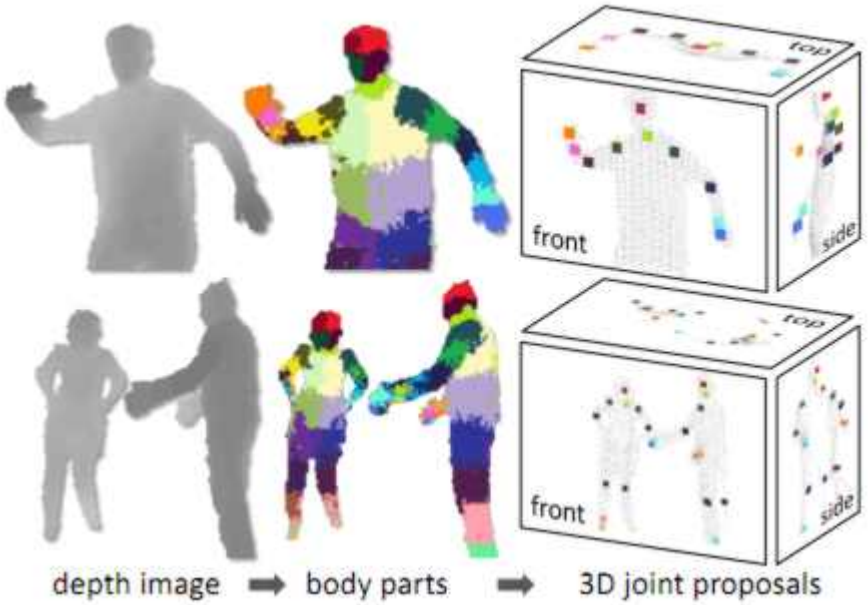
Make sure you understand the recursive algorithm.

Decision trees can be seen as nested rules.
Nested rules are FAST, and highly parallelizable.



x,y,z-coordinates per joint, ~60 total
x,y,z-velocities per joint, ~60 total
joint angles (~35 total)
joint angular velocities (~35 total)

```
if x>25 then
     if x>50 then y=0 else y=1; endif
else
     if x>16 then y=0 else y=1; endif
endif
```



Real-time human pose recognition in parts from single depth images

Computer Vision and Pattern Recognition 2011

Shotton et al, Microsoft Research

- *Trees are the basis of Kinect controller*
- *Features are simple image properties.*
- *Test phase: 200 frames per sec on GPU*
- *Train phase more complex but still parallel*

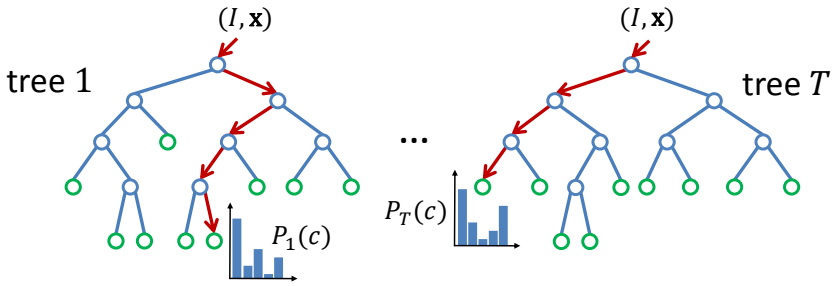


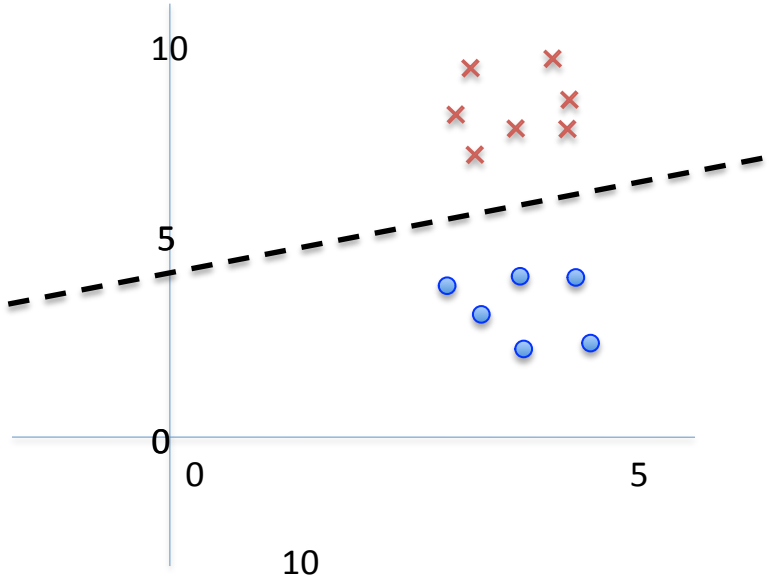
Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

3.3. Randomized decision forests

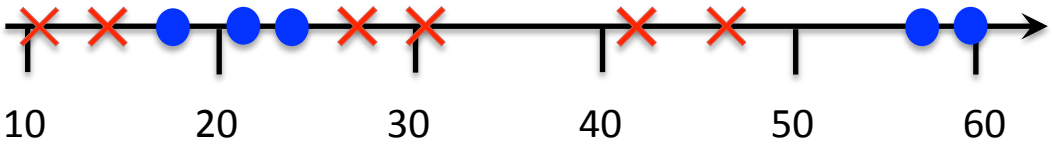
Randomized decision trees and forests [35, 30, 2, 8] have proven fast and effective multi-class classifiers for many tasks [20, 23, 36], and can be implemented efficiently on the GPU [34]. As illustrated in Fig. 4, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes.

To keep the training times down we employ a distributed implementation. Training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster.

We've been assuming **continuous** variables!

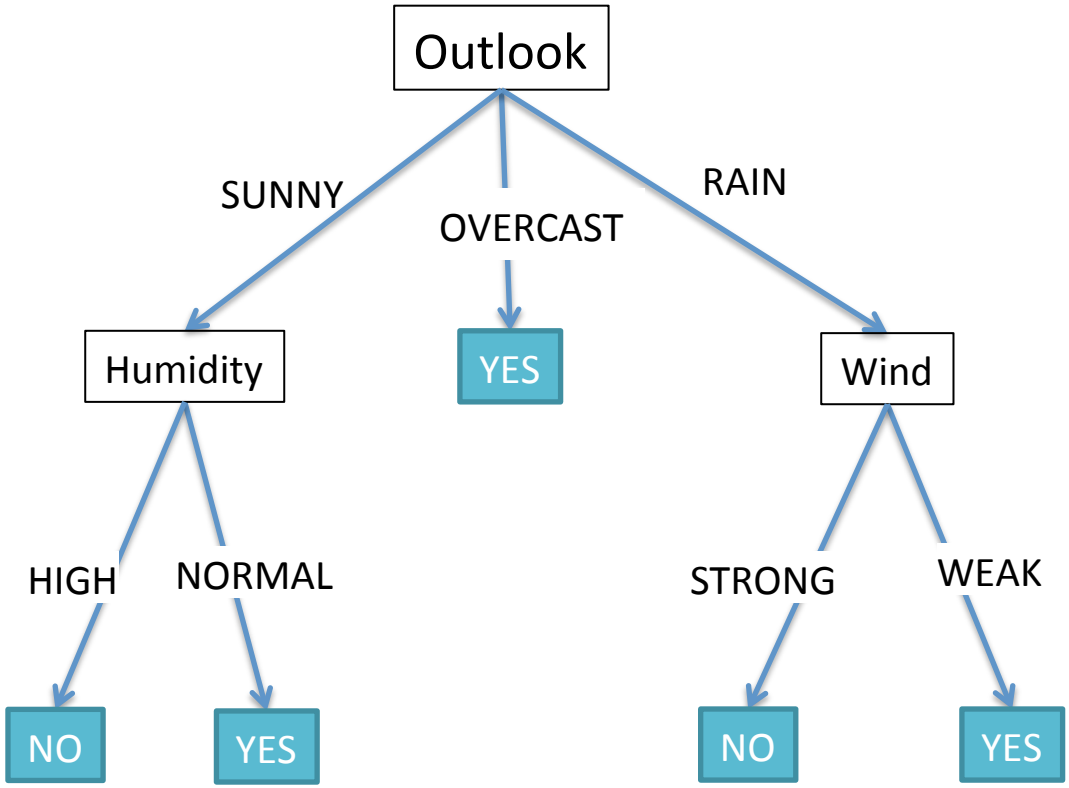


x_1,	x_2,	y (label)
98.79,	157.59,	1
93.64,	138.79,	1
42.89,	171.89,	0
...		
...		
87.91,	142.65,	1
97.92,	162.12,	1
47.63,	182.26,	0
92.72,	154.50,	1



The Tennis Problem

	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	**No**
2	Sunny	Hot	High	Strong	**No**
3	Overcast	Hot	High	Weak	**Yes**
4	Rain	Mild	High	Weak	**Yes**
5	Rain	Cool	Normal	Weak	**Yes**
6	Rain	Cool	Normal	Strong	**No**
7	Overcast	Cool	Normal	Strong	**Yes**
8	Sunny	Mild	High	Weak	**No**
9	Sunny	Cool	Normal	Weak	**Yes**
10	Rain	Mild	Normal	Weak	**Yes**
11	Sunny	Mild	Normal	Strong	**Yes**
12	Overcast	Mild	High	Strong	**Yes**
13	Overcast	Hot	Normal	Weak	**Yes**
14	Rain	Mild	High	Strong	**No**




```
if ( Outlook==sunny AND Humidity==high )      then NO
if ( Outlook==sunny AND Humidity==normal )    then YES
if ( Outlook==overcast )                      then YES
if ( Outlook==rain AND Wind==strong )         then NO
if ( Outlook==rain AND Wind==weak )           then YES
```

Decision Tree Learning Algorithm (sometimes called "ID3")

```
1: function BUILDTREE( subsample, depth )
2:
3:     //BASE CASE:
4:     if (depth == 0) OR (all examples have same label) then
5:         return most common label in the subsample
6:     end if
7:
8:     //RECURSIVE CASE:
9:     for each feature do
10:        Try splitting the data (i.e. build a decision stump)
11:        Calculate the cost for this stump
12:    end for
13:    Pick feature with minimum cost
14:
15:    Find left/right subsamples
16:    Add left branch ← BUILDTREE( leftSubSample, depth − 1 )
17:    Add right branch ← BUILDTREE( rightSubSample, depth − 1 )
18:
19:    return tree
20:
21: end function
```

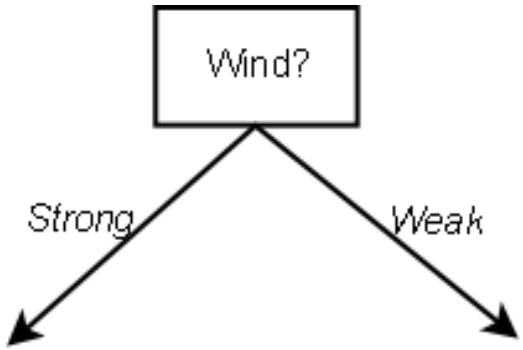
The Tennis Problem

	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Note: 9 examples say “YES”, while 5 say “NO”.

Partitioning the data…

	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



	Outlook	Temp	Humid	Wind	Play?
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

3 examples say yes, 3 say no.

	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

6 examples say yes, 2 examples say no.

Thinking in Probabilities...

Before the split : 9 'yes', 5 'no', ... $p('yes') = \frac{9}{14} \approx 0.64$

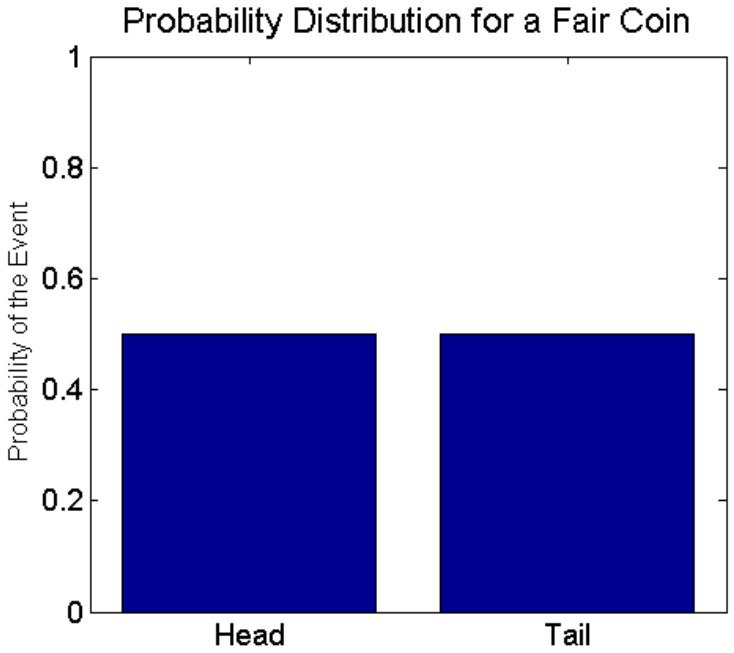
On the left branch : 3 'yes', 3 'no', ... $p('yes') = \frac{3}{6} = 0.5$

On the right branch : 6 'yes', 2 'no', ... $p('yes') = \frac{6}{8} = 0.75$

Remember... $p('no') = 1 - p('yes')$

The “Information” in a feature

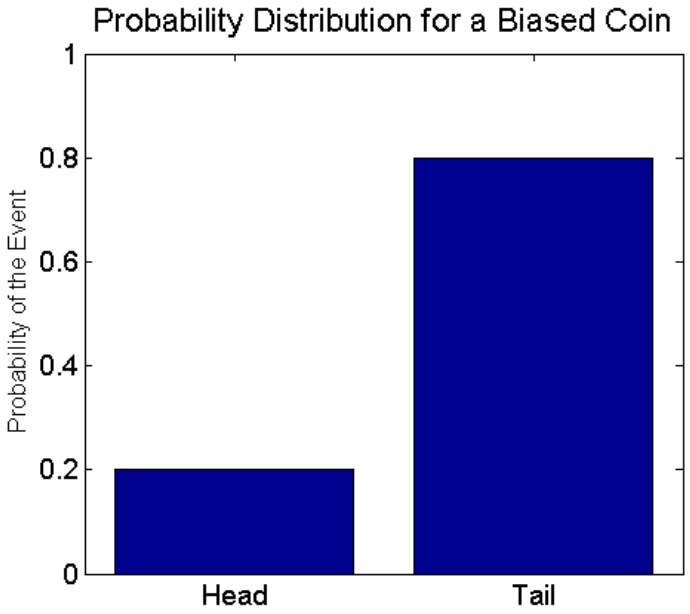
More uncertainty = less information



H(X) = 1

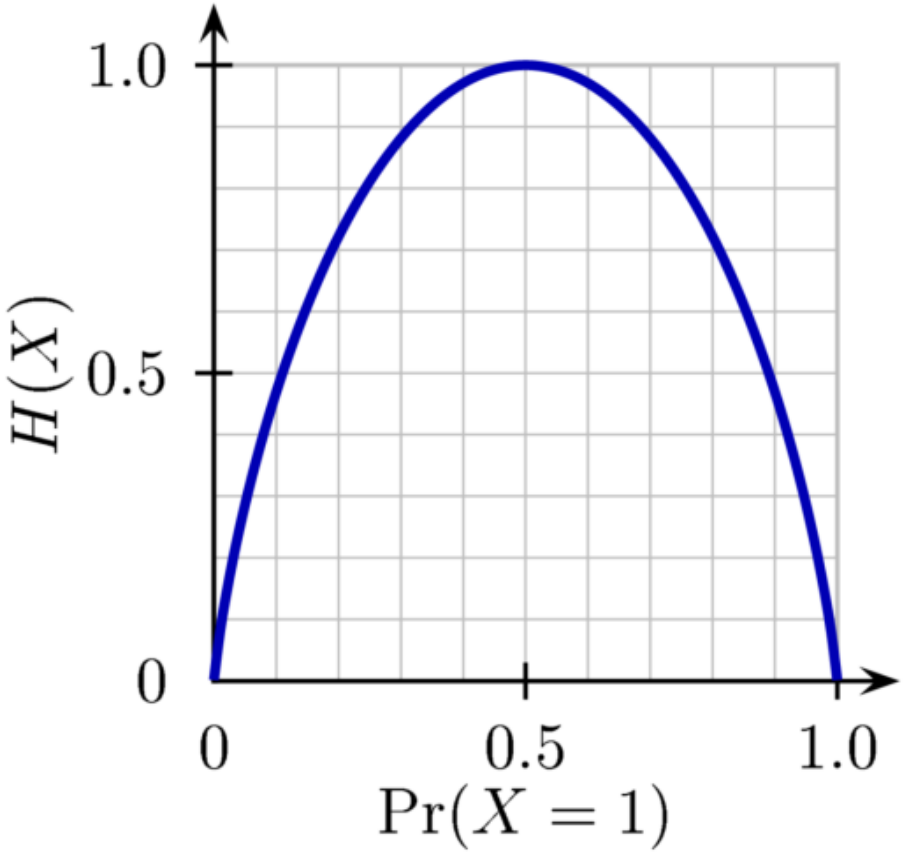
The “Information” in a feature

Less uncertainty = more information



H(X) = 0.72193

Entropy



$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

$$\begin{aligned} H(X) &= -\Big(p(head) \log p(head) + p(tail) \log p(tail)\Big) \\ &= -\Big(0.5 \log 0.5 + 0.5 \log 0.5\Big) \\ &= -\Big((-0.5) + (-0.5)\Big) = 1 \end{aligned}$$

Calculating Entropy

The variable of interest is "T" (for tennis), taking on 'yes' or 'no' values. Before the split : 9 'yes', 5 'no', ...
$p('yes') = \frac{9}{14} \approx 0.64$

In the whole dataset, the entropy is:

$$\begin{aligned} H(T) &= -\sum_i p(x_i) \log p(x_i) \\ &= -\left\{ \frac{5}{14} log \frac{5}{14} + \frac{9}{14} log \frac{9}{14} \right\} = 0.94029 \end{aligned}$$

$H(T)$ is the entropy **before** we split.

Information Gain, also known as "Mutual Information"

$H(T)$ is the entropy before we split.

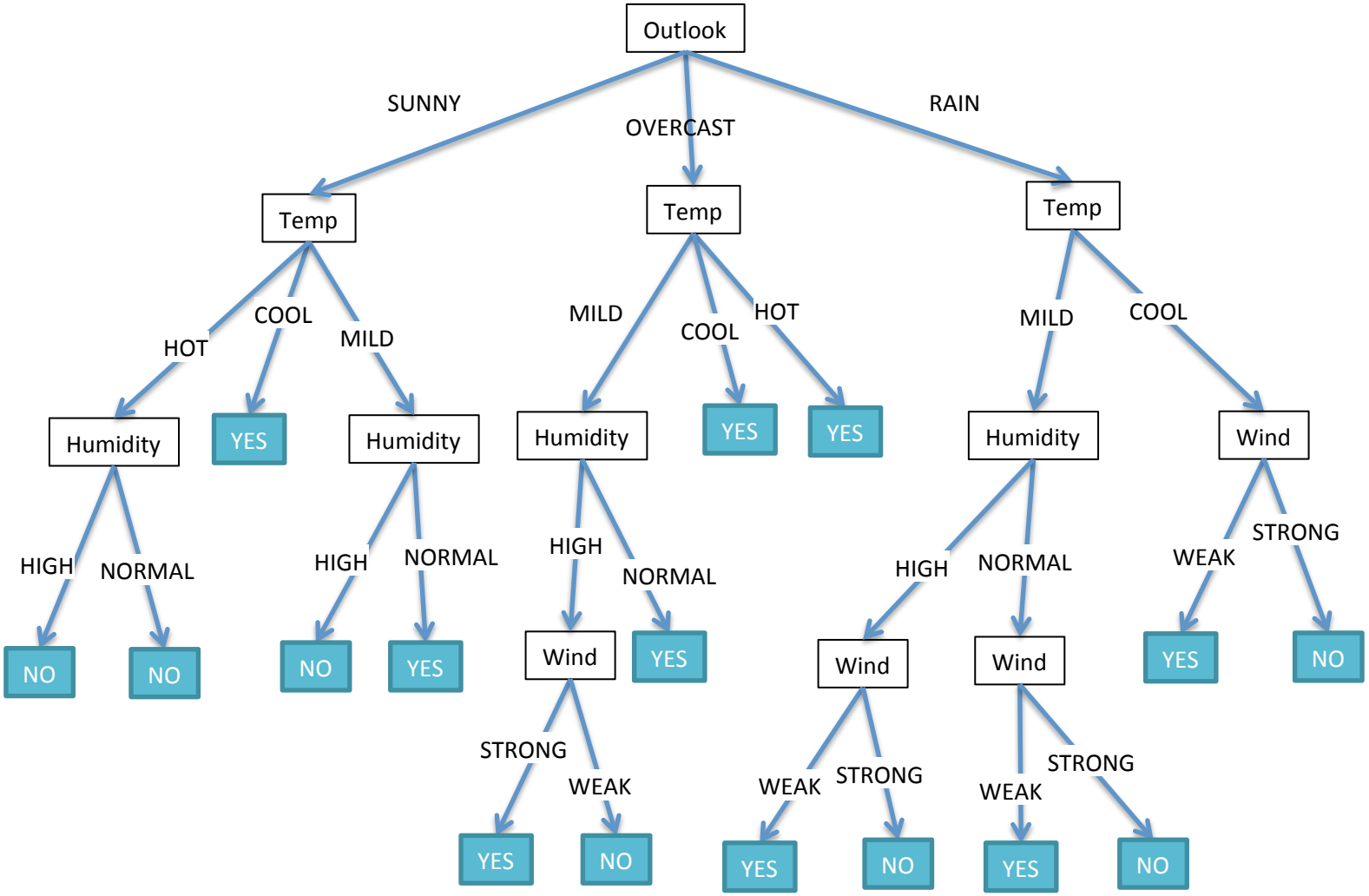
$H(T|W = \textit{strong})$ is the entropy of the data on the left branch.
$H(T|W = \textit{weak})$ is the entropy of the data on the right branch.

$H(T|W)$ is the weighted average of the two.

Choose the feature with maximum value of $H(T) - H(T|W)$.

Decision Tree Learning Algorithm (sometimes called "ID3")

```
1:  function BUILDTREE( subsample, depth )
2:
3:      //BASE CASE:
4:      if (depth == 0) OR (all examples have same label) then
5:          return most common label in the subsample
6:      end if
7:
8:      //RECURSIVE CASE:
9:      for each feature do
10:         Try splitting the data (i.e. build a decision stump)
11:         Calculate gain for this stump
12:     end for
13:     Pick feature with ~~minimum cost~~
14:                          maximum information gain
15:     Find left/right subsamples
16:     Add left branch ← BUILDTREE( leftSubSample, depth − 1 )
17:     Add right branch ← BUILDTREE( rightSubSample, depth − 1 )
18:
19:     return tree
20:
21: end function
```


Outlook
SUNNY
OVERCAST
RAIN
Temp
Temp
Temp
HOT
COOL
MILD
MILD
COOL
HOT
MILD
COOL
Humidity
YES
Humidity
Humidity
YES
YES
Humidity
Wind
HIGH
NORMAL
HIGH
NORMAL
HIGH
NORMAL
HIGH
NORMAL
WEAK
STRONG
NO
NO
NO
YES
Wind
YES
Wind
Wind
YES
NO
STRONG
WEAK
WEAK
STRONG
WEAK
STRONG
YES
NO
YES
NO
YES
NO

Decision trees

Trees learnt by recursive algorithm, ID3 (but there are many variants of this)

Trees draw a particular type of decision boundary (look it up in these slides!)

Highly efficient, highly parallelizable, lots of possible “splitting” criteria.

Powerful methodology, used in lots of industry applications.

SELF-TEST

Calculate the entropy of a two-valued feature, i.e. $x \in \{0, 1\}$, with $p(x = 1) = 0.6$.