Decision Trees

Tennis with a flaky partner

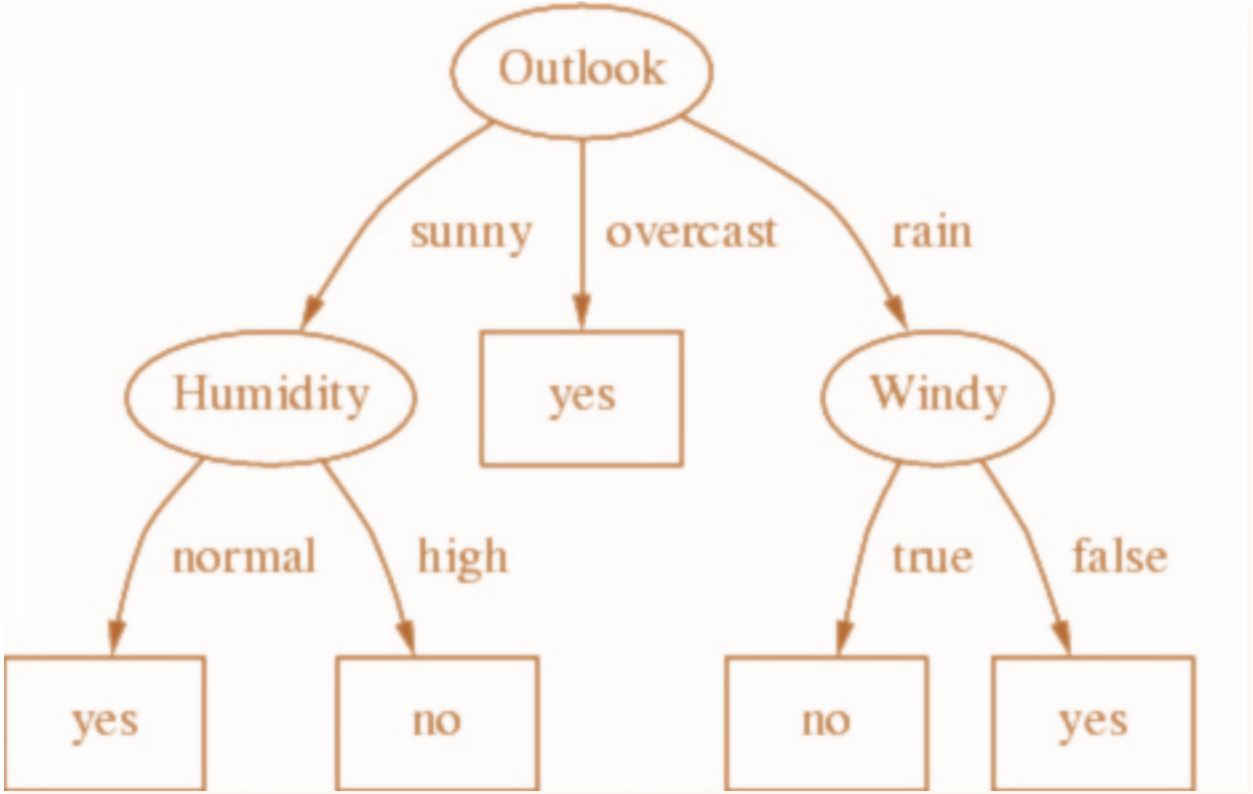
Temp	Outlook	Humidity	Windy	Played
Hot	Sunny	High	False	No
Hot	Sunny	High	True	No
Hot	Overcast	High	False	**Yes**
Cool	Rain	Normal	False	**Yes**
Cool	Overcast	Normal	True	**Yes**
Mild	Sunny	High	False	No
Cool	Sunny	Normal	False	**Yes**
Mild	Rain	Normal	False	**Yes**
Mild	Sunny	Normal	True	**Yes**
Mild	Overcast	High	True	**Yes**
Hot	Overcast	Normal	False	**Yes**
Mild	Rain	High	True	No
Cool	Rain	Normal	True	No
Mild	Rain	High	False	**Yes**

Modeling these whims

That change as often as the weather does...

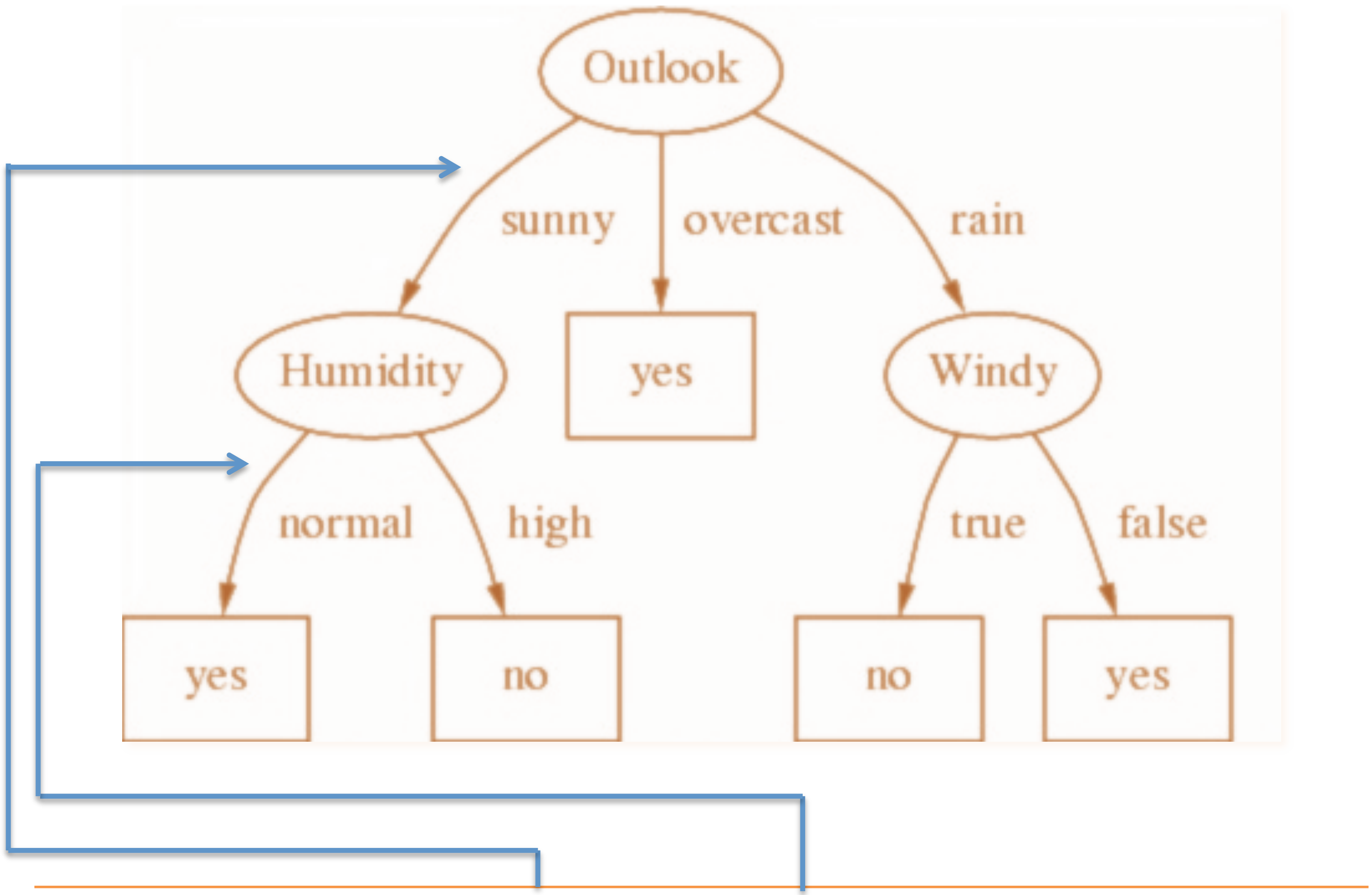
Day	Temp	Outlook	Humidity	Windy	Played
Today	Cool	Sunny	Normal	False	?
Tomorrow	Mild	Sunny	Normal	False	?

Decision Trees!



Day	Temp	Outlook	Humidity	Windy	Played
Today	Cool	Sunny	Normal	False	?
Tomorrow	Mild	Sunny	Normal	False	?

Decision Tree



Day	Temp	Outlook	Humidity	Windy	Played
Today	Cool	Sunny	Normal	False	?
Tomorrow	Mild	Sunny	Normal	False	?

Decision Tree Terminology

A decision tree consists of

- **Nodes:**
  Test for the value of a certain attribute
- **Edges:**
  Correspond to the outcome of a test
  Connect a node to the next node or leaf
- **Root:**
  The node that performs the first split
- **Leaves:**
  Terminal nodes that predict the outcome

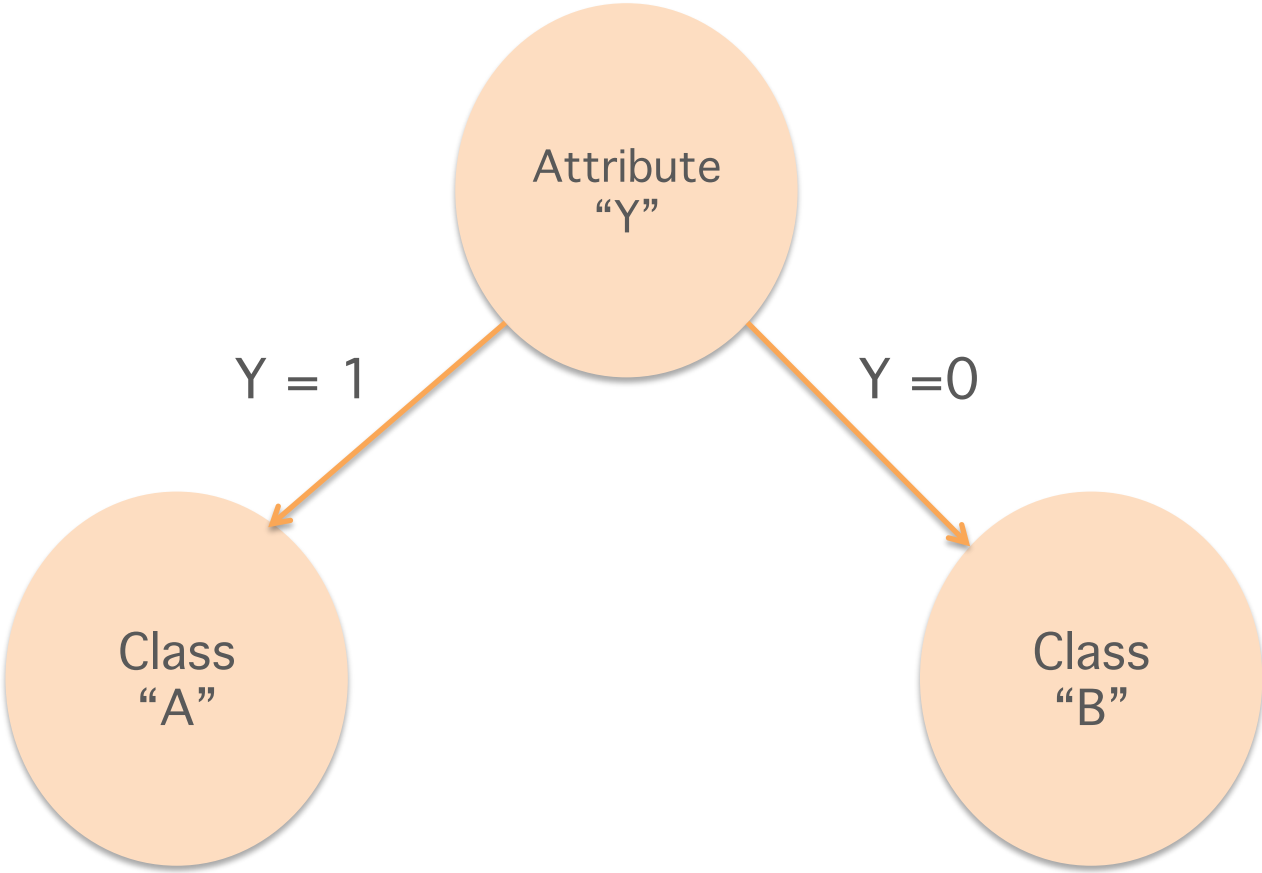
A toy example

Training set: 3 features and 2 classes

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

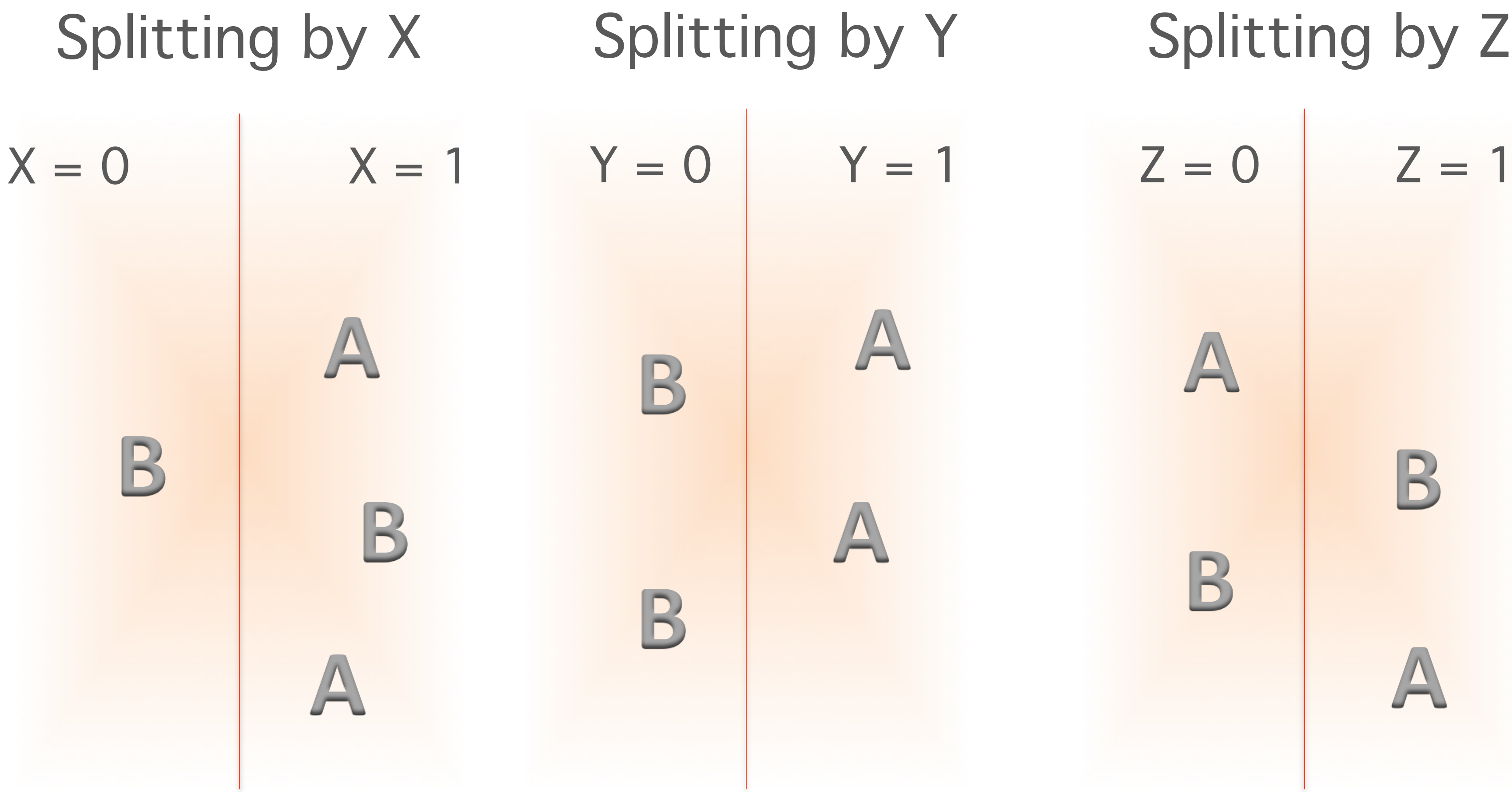
How can Class A be distinguished from Class B?

Exploring our intuition



Splitting on Y gives us a clear separation between classes
We could also first split on X and then split on Z. But this is less optimal than splitting on Y
We cannot root our tree with Z

Formalizing this intuition



For automatic tree construction:
Consider maximizing a measure of information gain OR
Minimizing a measure of impurity

Shannon's Entropy & Information Gain

Entropy, H(S) = $$\sum_{i=1}^{c} -p_i \log_2 p_i$$

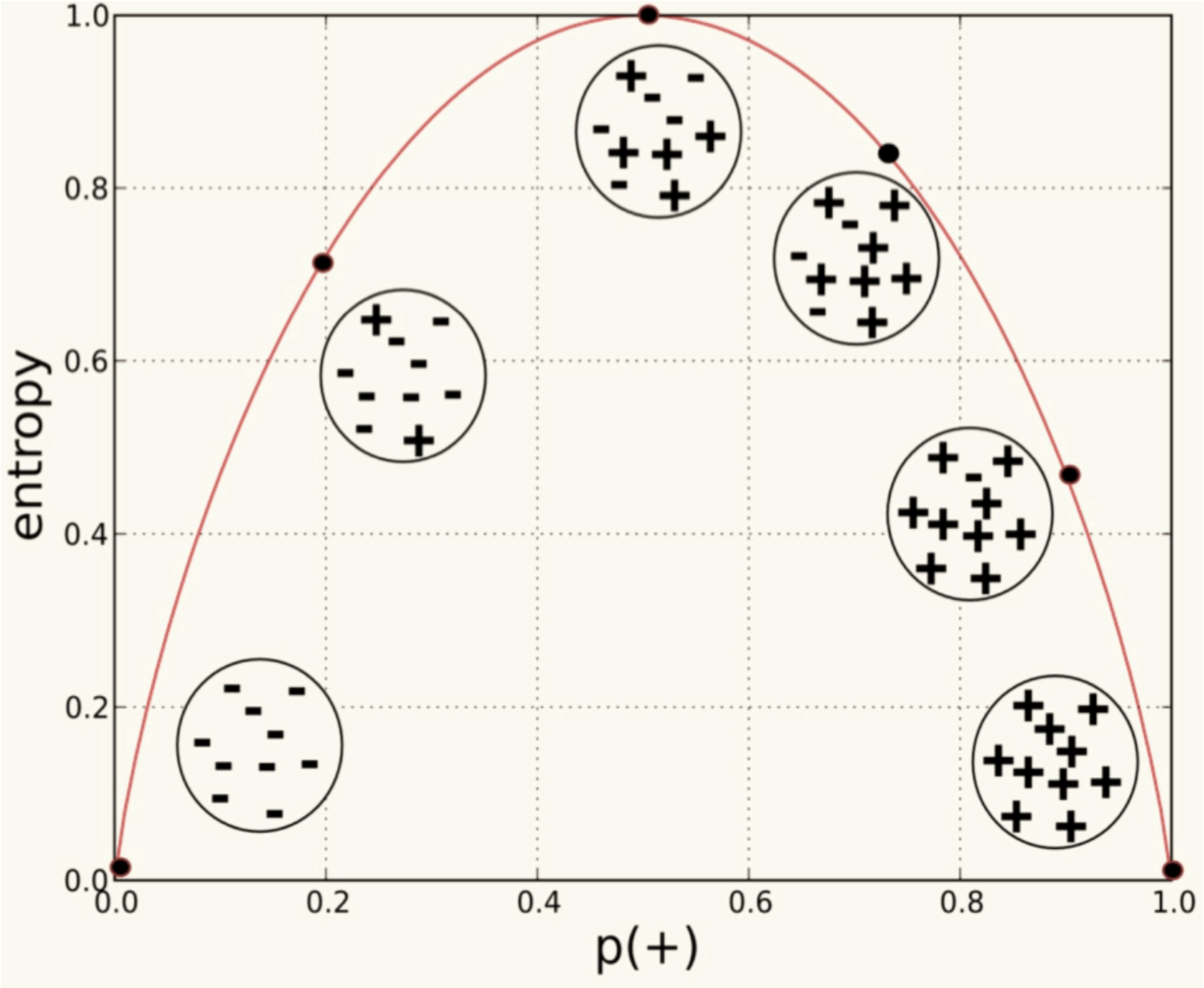
Information Gain (S,A) = H(S) $$- \sum_{v \in Values(A)} \frac{|S_v|}{|S|}$$ H(S_v)

H(S): Entropy for a set S. Each element S belongs to a certain class.
p_i : Probability of the class 'i' in set S
A: Denotes an attribute your set S can be split on (eg. Outlook, Humidity, Windy etc)
v ε Values(A): Denotes the set of values a given attribute can take on (As an eg. For attribute A = Outlook, v takes elements from {Sunny, Overcast}
| | operator: Simply tells you how many elements in the set. For example for attribute "Outlook" $|S_{sunny}|$ gives how many days in your dataset were sunny.

Entropy is a measures of randomness. The less probable an event the more random it is…AND therefore the more information it contains. Consider "The sky is falling" vs "The sun just rose"

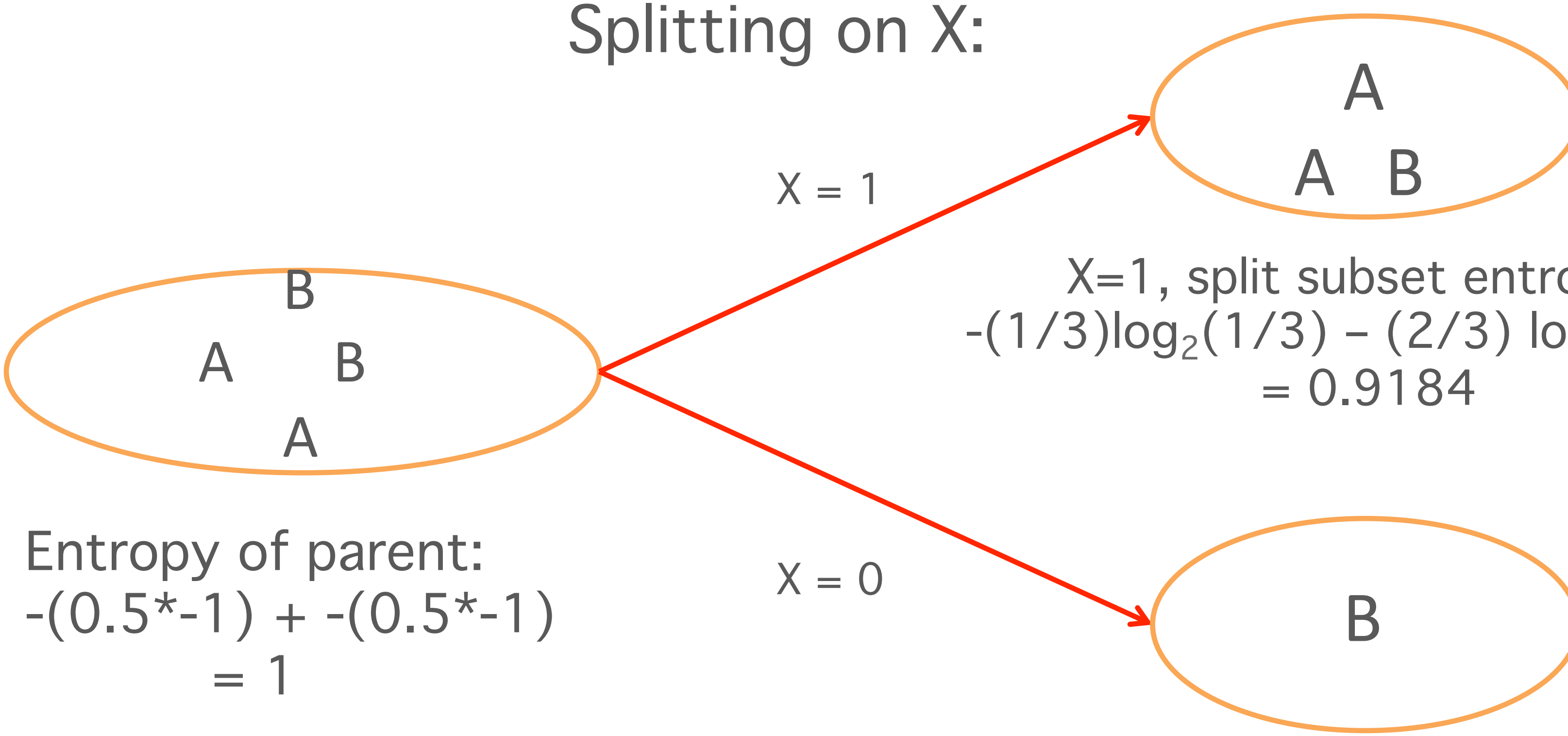
More on Entropy

$$\text{Entropy, H}(S) = \sum^{c} -p_i \log_2 p_i$$



Our toy example

Splitting on X:



X=1, split subset entropy:
$-(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= 0.9184$

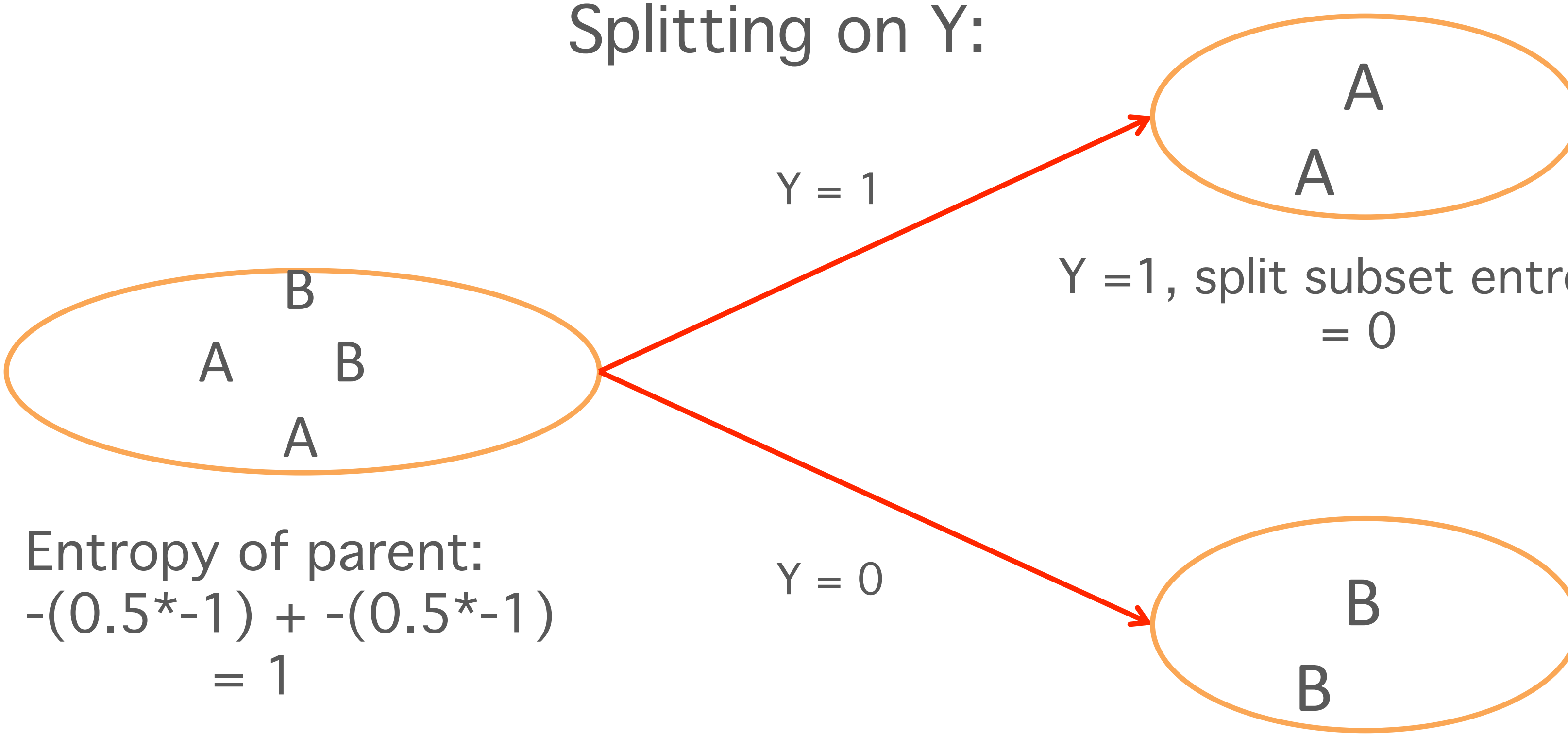
Entropy of parent:
$-(0.5*-1) + -(0.5*-1)$
$= 1$

X = 0, split subset entropy :
$= 0$

Information Gain from splitting on X:
$1 - \{0.75 * 0.9184 + 0.25*0\} = .3112$

Our toy example

Splitting on Y:



Y =1, split subset entropy:
= 0

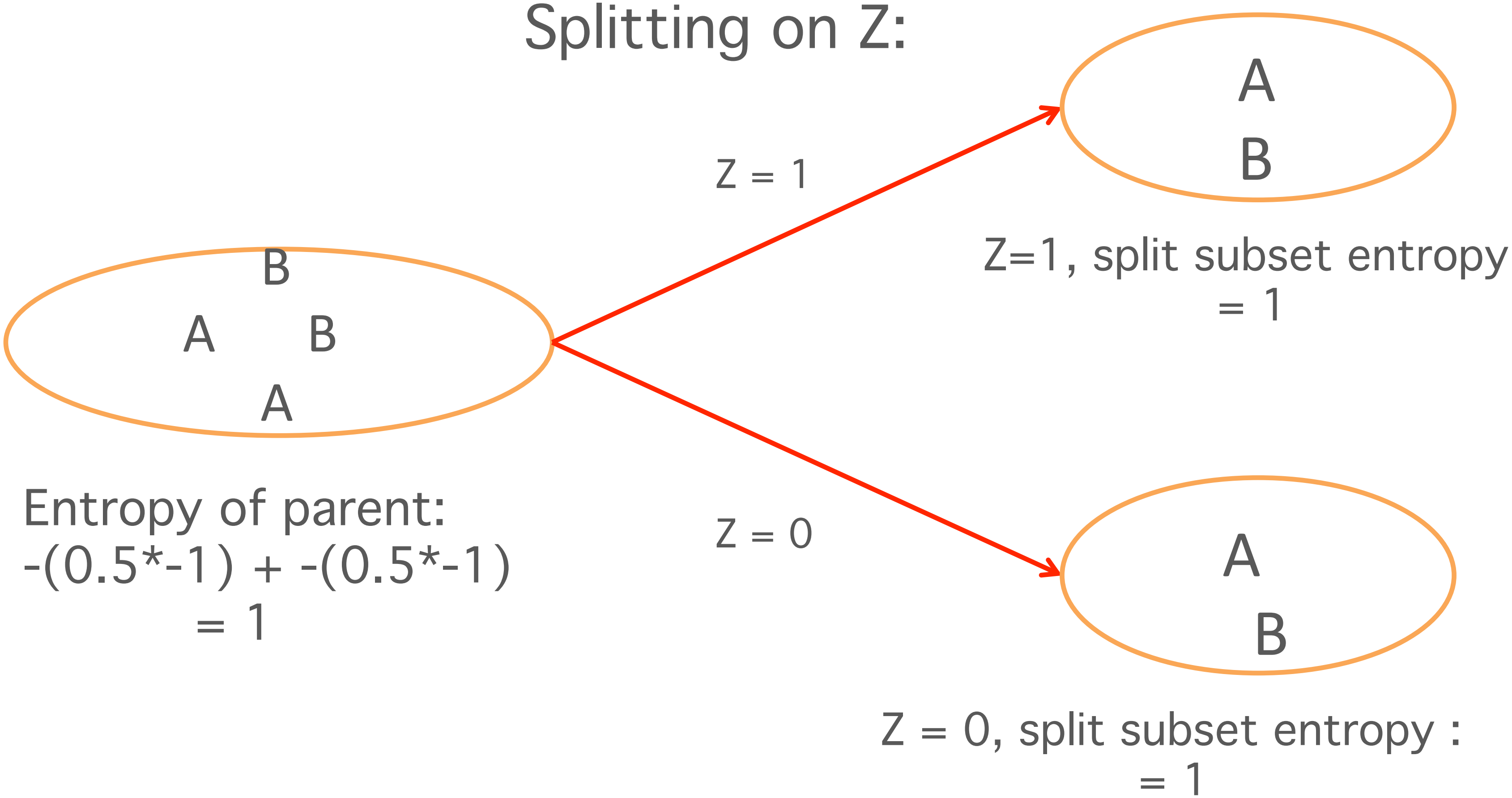
Entropy of parent:
-(0.5*-1) + -(0.5*-1)
= 1

Y = 0, split subset entropy :
= 0

Information Gain from splitting on Y:
1 – {0.5 *0 + 0.5*0} = 1, BEST!

Our toy example

Splitting on Z:



Information Gain from splitting on Z:

1 – {0.5 *1+ 0.5*1} = 0, WORST!

Gini Index

Many alternatives to Information Gain, the most popular being the Gini Index

- The Gini Index is a measure of impurity (not entropy).
- It gives the probability that any element when randomly classified according to the distribution of the classes is labeled incorrectly

$$Gini(S) = 1 - \sum_i p_i^2$$

Minimize the average Gini Index {impurity}, to make splits in the decision tree

$$Gini(S, A) = \sum_i \frac{|S_i|}{|S|} \cdot Gini(S_i)$$

Building a Decision Tree

Psuedo code

```
function BuildTree:
If every item in the dataset is in the same class or
there is no feature left to split the data:
     return a leaf node with the class label
Else:
     find the best feature and value to split the data
     split the dataset
     create a node
     for each split
     call BuildTree & add the result as a child of the node
return node
```

Other Caveats

- Handling numerical data
- Handling missing data
- Binary vs Multi-way splits
- Using Decision Trees for regression:

Can't use information gain or gini impurity

- Choose best splits using residual sum of squares (calculate against mean value of each leaf)
- Can also use a combination of decision trees and linear regression on the leaf nodes (model trees)

What could go possibly wrong?!?

Your tree will correctly fit EVERY SINGLE sample in the training set!

In other words...
OVERFITTING is always a problem.
We "prune" our tree to address overfitting.

Pre pruning/Early Stopping

- Leaf size
  Stop when the number of data points for a leaf gets below a threshold

- Depth
  Stop when the depth of the tree (distance from root to leaf) reaches a threshold

- Mostly the same
  Stop when some percent of the data points are the same (rather than all the same)

- Error threshold
  Stop when the error reduction (information gain) is not improved significantly

Post pruning - CV

Involves building the tree first & then choosing to cut off some of the leaves.

Psuedo code

```
function Prune:
    if either left or right is not a leaf:
        call Prune on that split
    if both left and right are leaf nodes:
        calculate error associated with merging two nodes
        calculate error associated without merging two nodes
    if merging results in lower error:
        merge the leaf nodes
```

In pursuit of pruning

What if we could find ways to automatically prune?

- No having to set parameters like depth etc
- No calculating errors

How about we build more than one tree and find ways to automatically combine them that reduce overfitting?

COMING AHEAD IN FURTHER LECTURES...

Decision Trees, Summary

Why Decision Trees

- Easily interpretable
- Handles missing values and outliers
- Non-parametric/non-linear/discontinuity/ model complex phenomenon
- Computationally cheap to predict
- Can handle irrelevant features
- Mixed data (nominal and continuous)

Why Not Decision Trees

- Computationally expensive to train
- Very easy to overfit
- Greedy algorithm (local optima)
- Algorithm makes best splits

Decision Trees in sklearn

- Pruning with max_depth, min_samples_split, min_samples_leaf or max_leaf_nodes
- Gini is default, but you can also choose entropy
- Does binary splits (you would need to binarize categorical features)