DECISION TREE

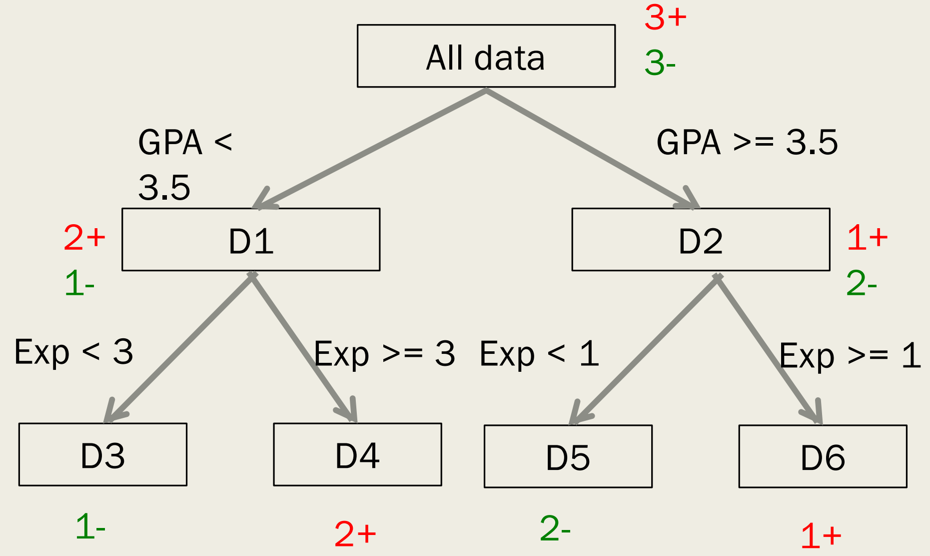
Anurag Nagar

What is a DT?

Decision Tree

- 2-D case

GPA	Years Exp	Internship
4.0	0	0
3.0	1	0
3.4	4	1
3.6	0	0
3.8	4	1
2.5	3	1



Of course, you could start with Exp as the first sorting or splitting criteria and get a different tree.

Decision Tree - Representation

- In this class, DT is a way to represent concepts & hypothesis about a target concept
- Can be written in form of rules

IF exp > 3 and GPA > 3.5 THEN internship = 1

- Leaf nodes decide values of output variable
- Internal nodes & edges represent splitting (sorting) criteria.
- We will consider Boolean attributes and output.
- Given instances with n Boolean attributes i.e. each X^i is of the form:

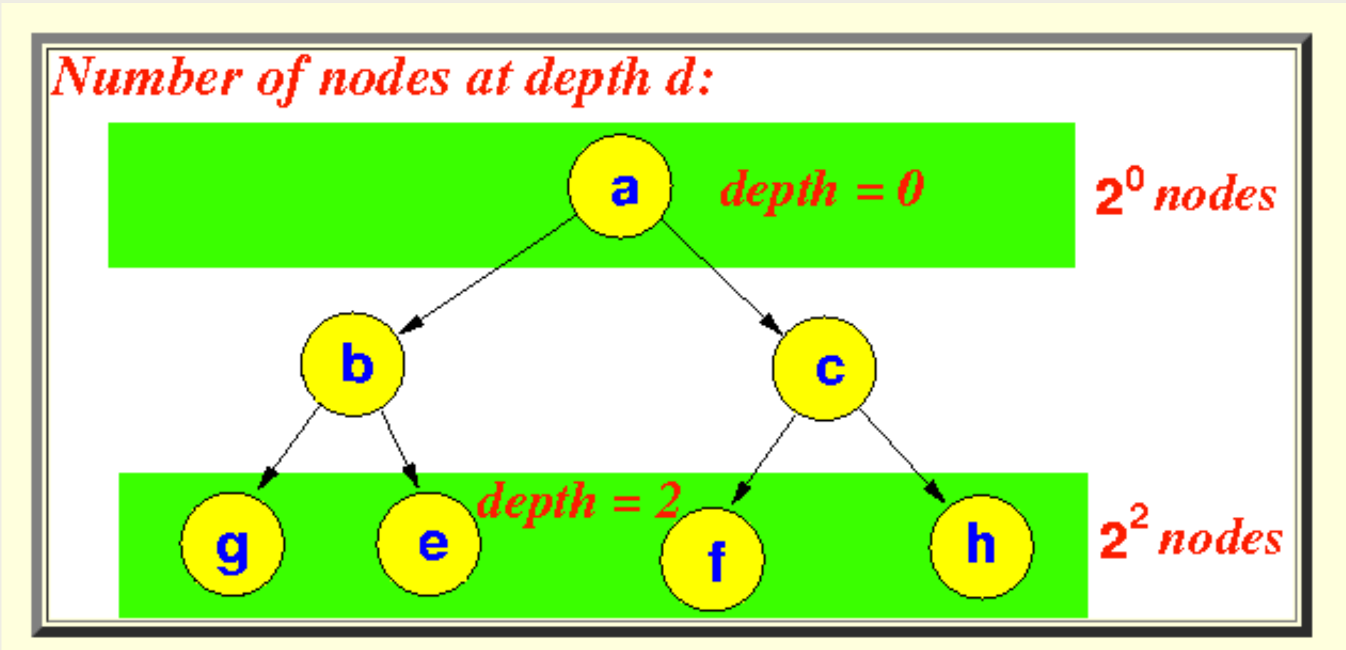
X^i = (x1, x2, …, xn) e.g. X^i = (0, 1, …, 0)

How will you represent one hypothesis

=> A binary tree

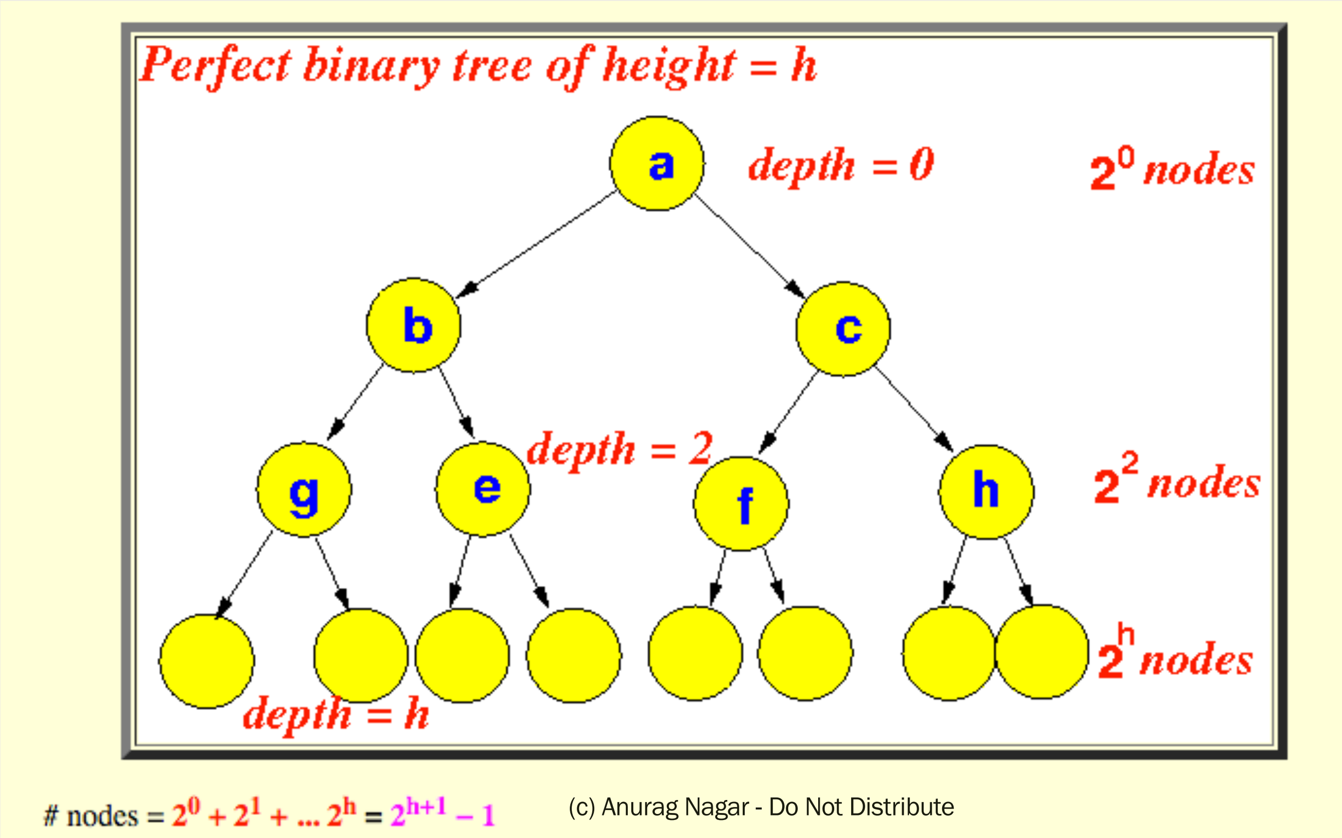
Properties of Binary Trees

- A complete binary tree of height h has $2^{h+1} - 1$ nodes
- Number of nodes at depth d is 2^d



Properties of Binary Trees

- A complete binary tree of height h has 2^{h+1} -1 nodes
- Number of nodes at depth d is 2^d



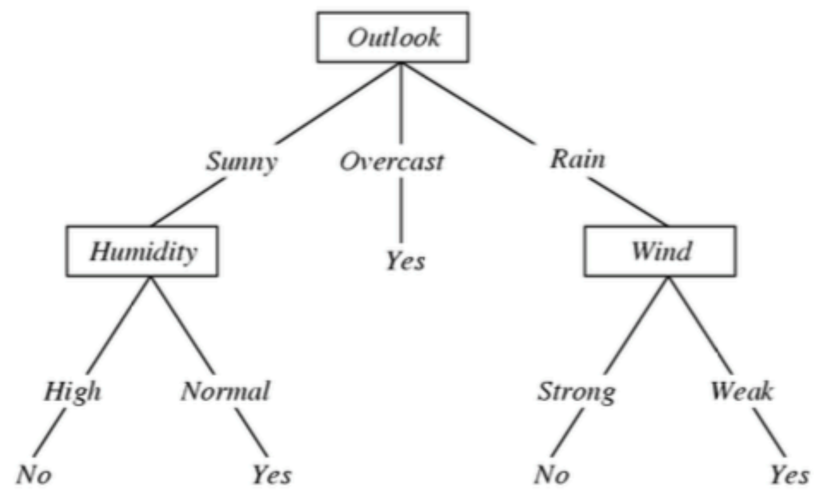
Learning a DT

Use of DT in learning

- Use the training examples and their labels to construct decision tree
- For example, (X^1, y^1) could be ((0, 0, 0), 1)
- You can use DT to model knowledge from training data.

A Decision tree for

F: <Outlook, Humidity, Wind, Temp> → PlayTennis?



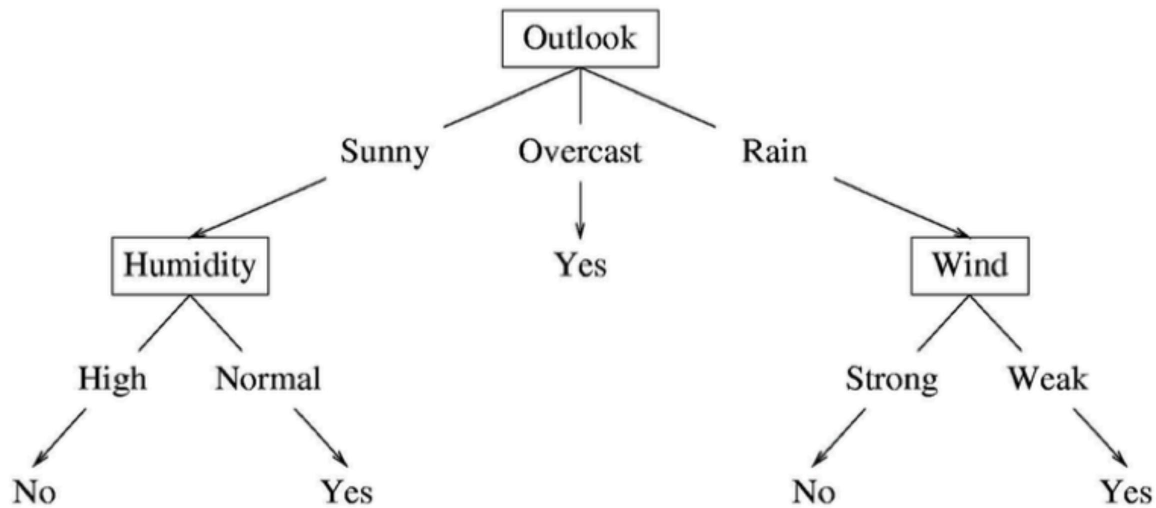
Each internal node: test one discrete-valued attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

Using DT to represent hypotheses

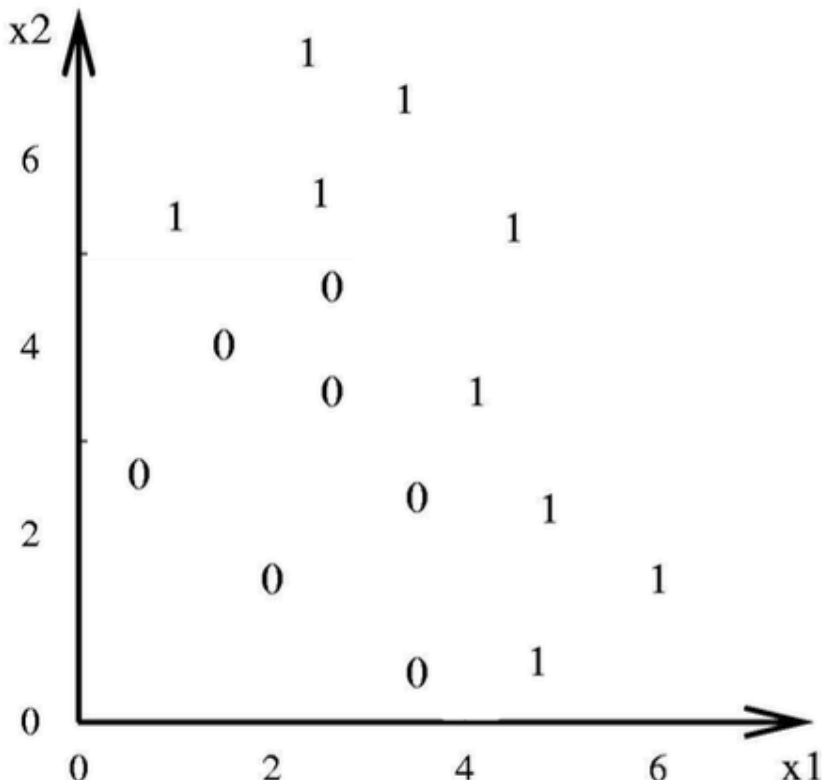
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.

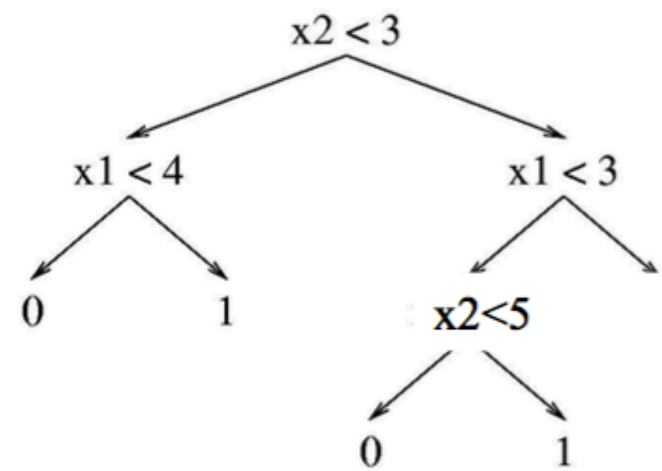


Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

Classification Boundary of a DT

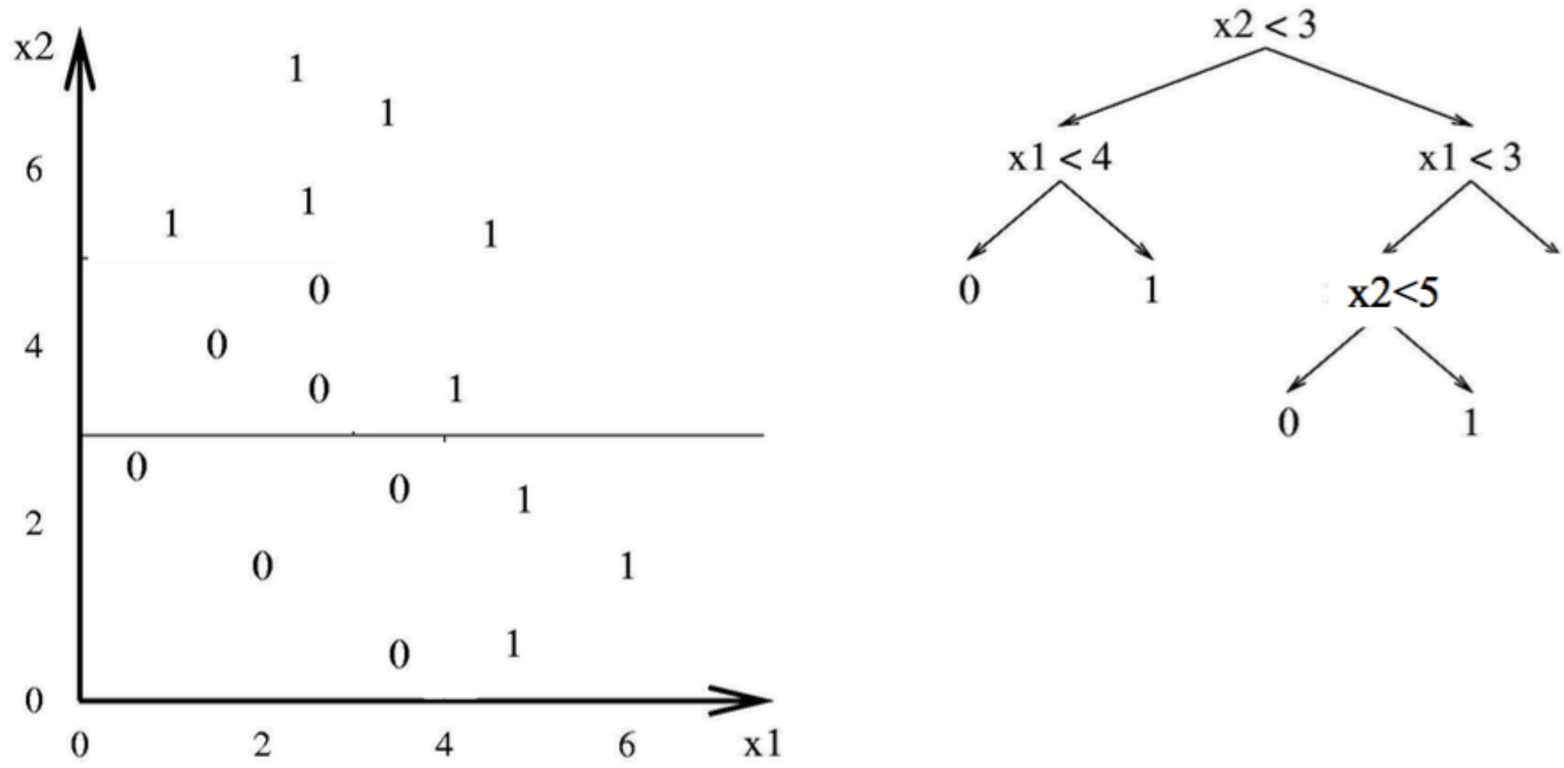
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





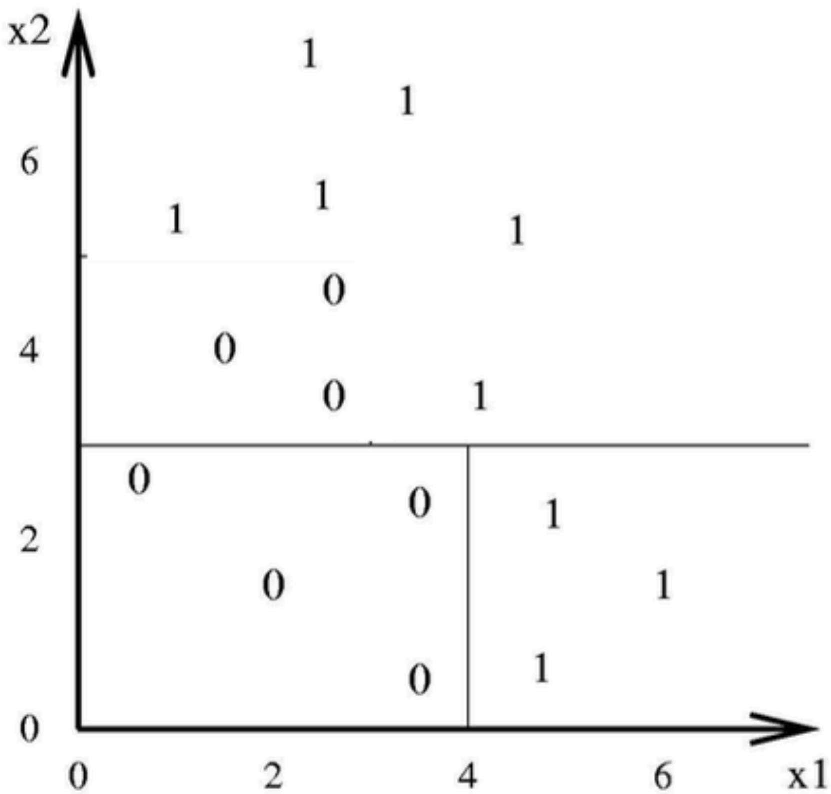
Classification Boundary of a DT

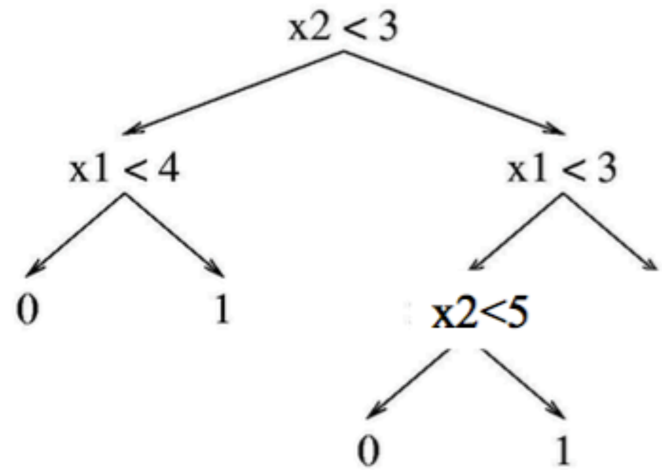
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.



Classification Boundary of a DT

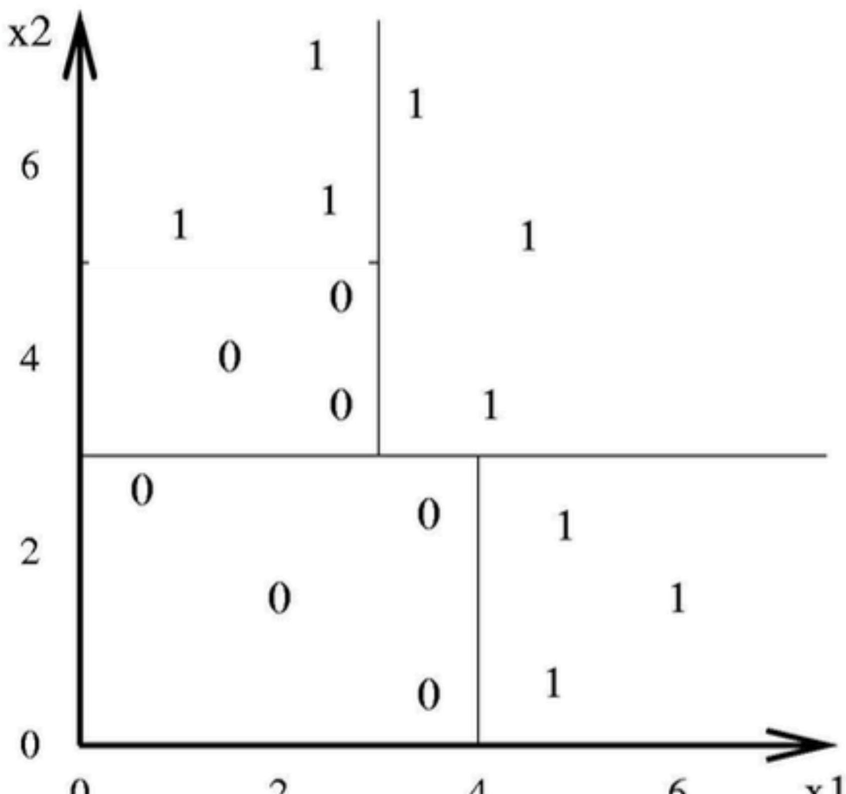
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

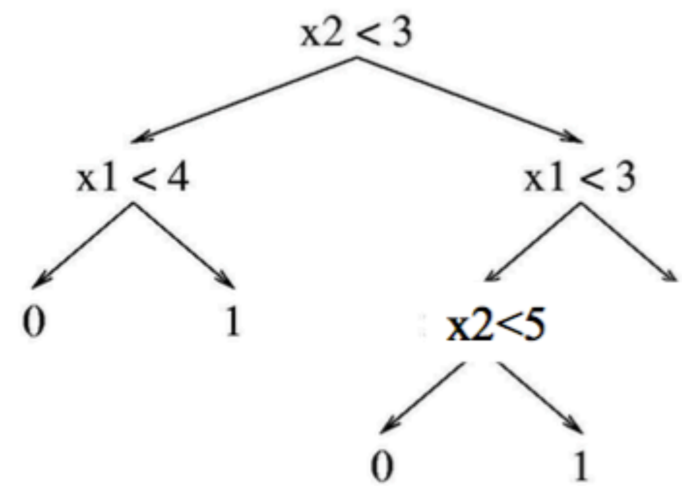




Classification Boundary of a DT

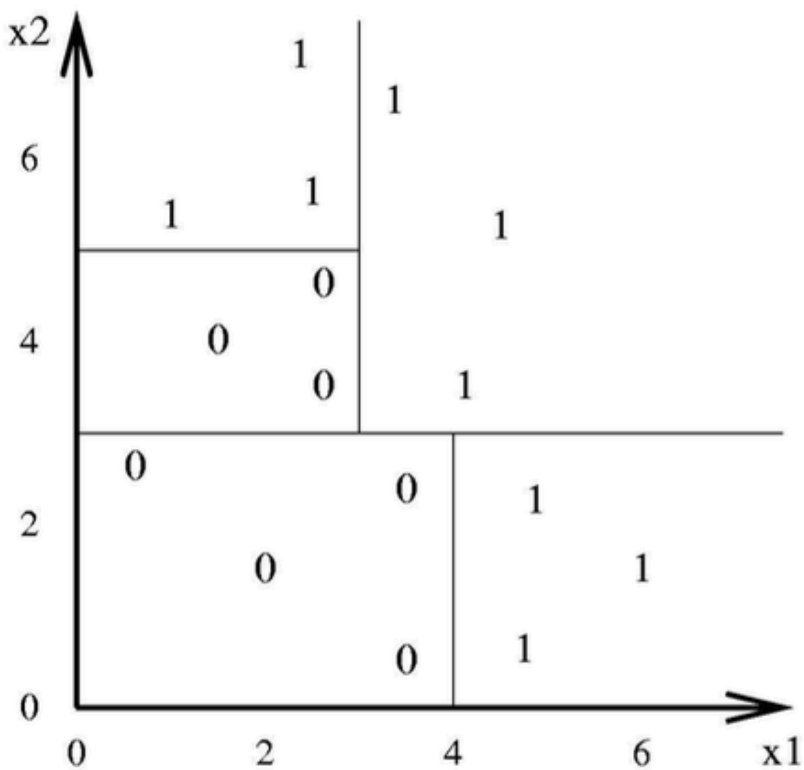
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

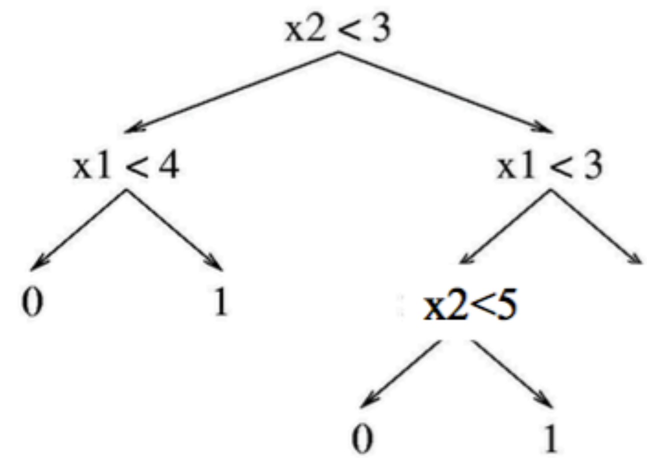




Classification Boundary of a DT

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Hypothesis Space of DT

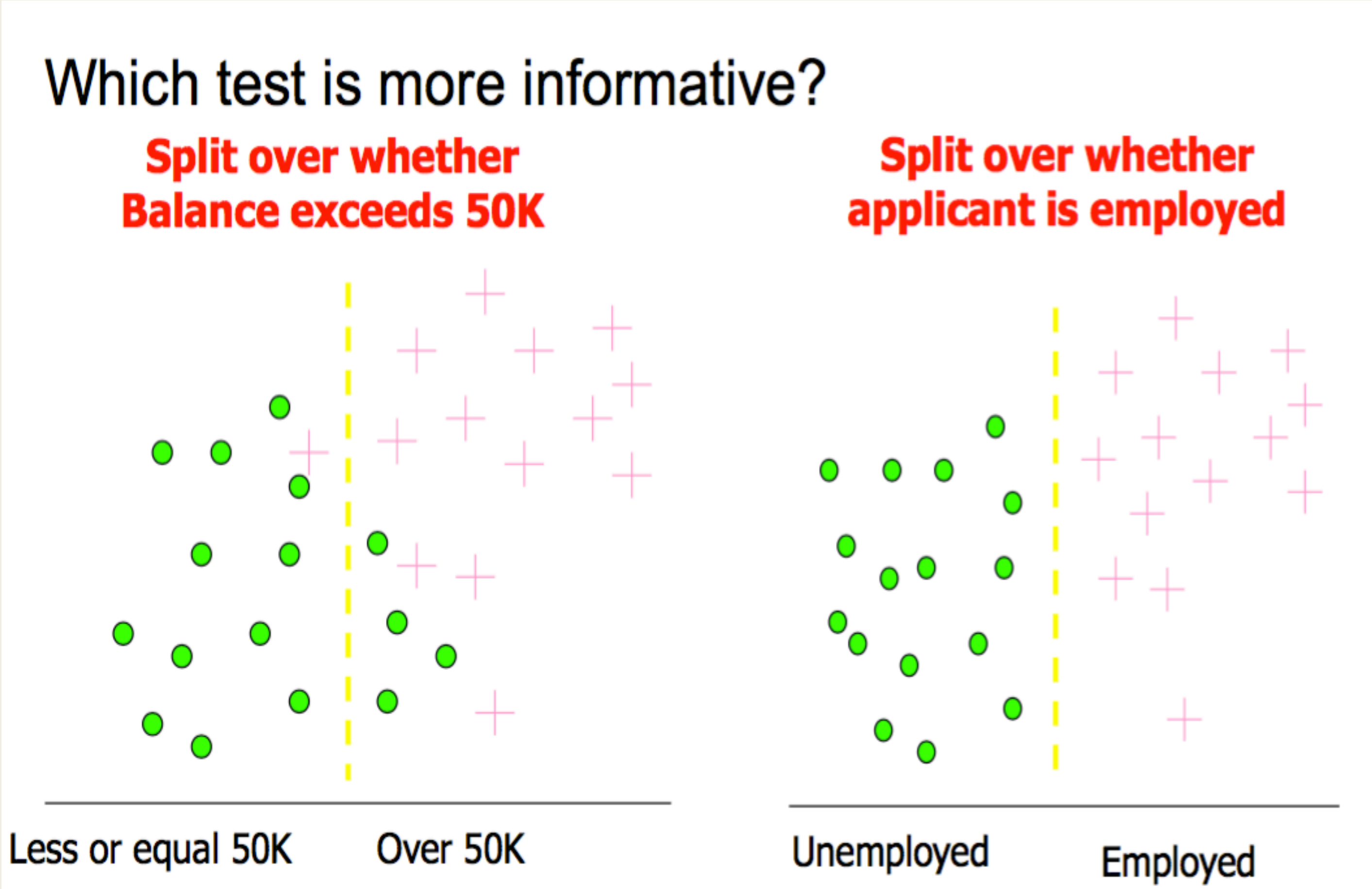
As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)$
- **etc.**

Finding the best split (also called sort)

DT – How to find best sorting?

- Which attribute should I sort (split) on first?
  - It DOES make a difference.
- Informally, we want that split that gives maximum purity at each node i.e. split such that all instances are of a single class (or close to it).



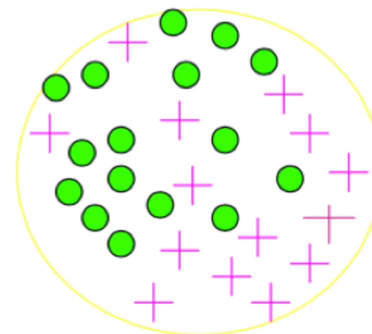
Entropy

Entropy is a measure of Information Content (IC).

$$H(X) = \sum -p_i log_2\, p_i$$

where pi is the probability of the i^{th} class.

If you think deeply, it is the expected value of $-\log_2 p_i$ or $\log(1/p_i)$.
This quantity is also known as information of an attribute.
Also, H(x) can be thought of as the number of bits needed to encode a dataset



- Entropy = $\sum_i - p_i \log_2 p_i$

  p_i is the probability of class i
  Compute it as the proportion of class i in the set.

  16/30 are green circles; 14/30 are pink crosses
  $\log_2(16/30)$ = -.9; $\log_2(14/30)$ = -1.1
  Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

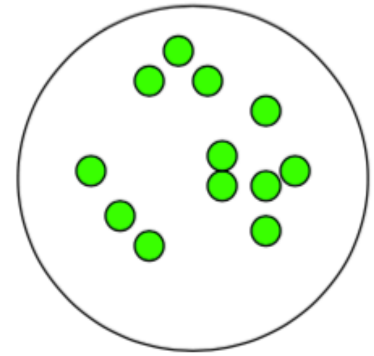
- Entropy comes from information theory. The higher the entropy the more the information content.

  What does that mean for learning from examples?

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = $-1 \log_2 1 = 0$

not a good training set for learning

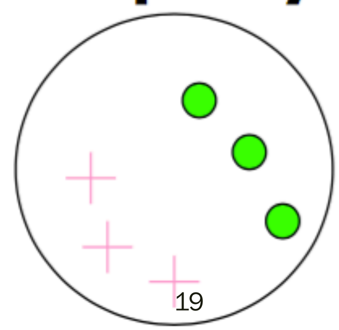
Minimum impurity



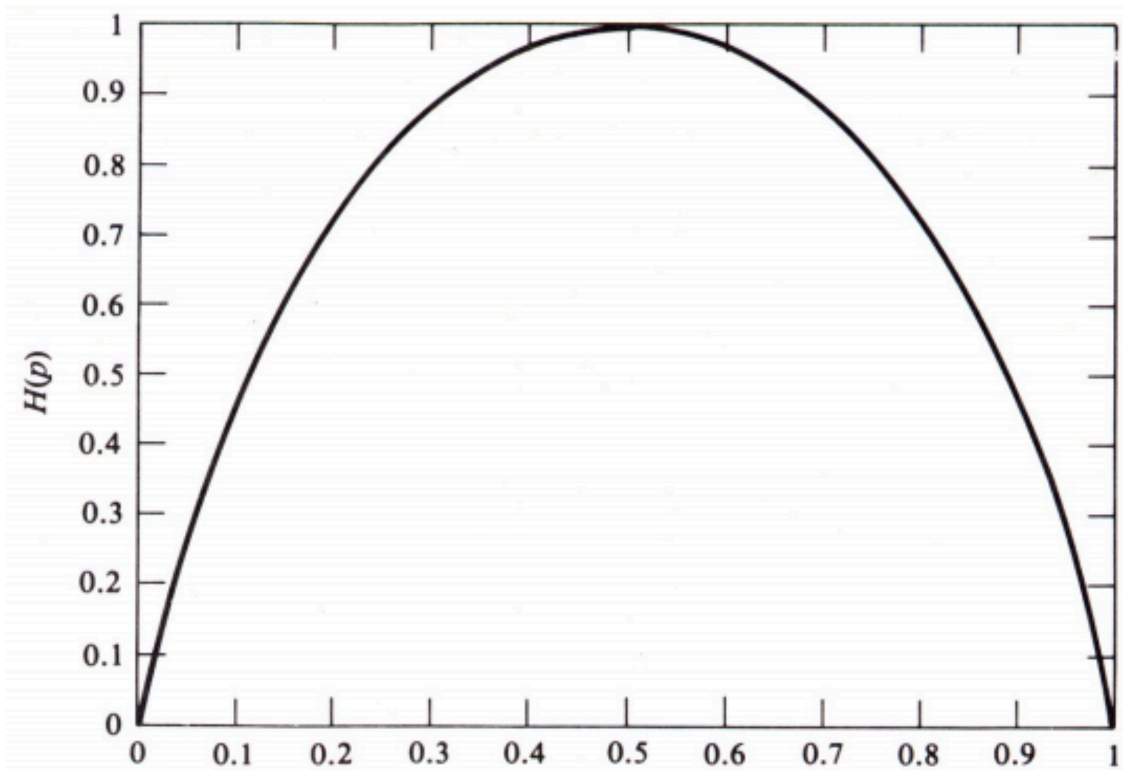
- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

good training set for learning

Maximum impurity



Entropy of a binary random variable



- Entropy is maximum at p=0.5
- Entropy is zero and p=0 or p=1.

Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

- Information gain tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Information Gain

- Suppose you just know the class labels initially.
- Then you know one of the attributes.

=> Does it really help you?
=> Do you get any information gain or reduction in entropy ?
=> Do you get any increase in purity of the classes by knowing an attribute?

Mean the same thing

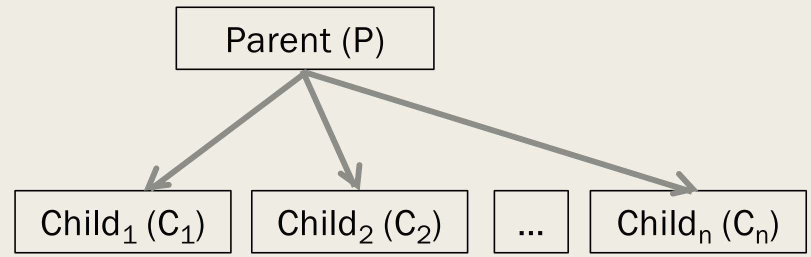
Information Gain

- Information Gain (IG) is defined as:

  IG = Parent Entropy – Average Entropy of children

- Average entropy of children:
  Probability weighted entropy of all the child nodes.
  $= \frac{C_1}{P} \times H_1 + \frac{C_2}{P} \times H_2 + \cdots + \frac{C_n}{P} \times H_n$

- Information Gain (IG) =
  $= \mathrm{H} - (\frac{C_1}{P} \times H_1 + \frac{C_2}{P} \times H_2 + \cdots + \frac{C_n}{P} \times H_n)$



Parent has P data instances and entropy of H
After splitting into n child nodes:
Child$_1$ has C_1 data instances and entropy of H_1
Child$_2$ has C_2 data instances and entropy of H_2
....
Child$_n$ has C_n data instances and entropy of H_n

Example of IG

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Class attribute is defaulted?
Independent Attributes - the first two

How many bits does it take to specify the attribute of 'defaulted?'

- **P(defaulted = Y) = 3/10**
- **P(defaulted = N) = 7/10**

$$\begin{aligned} H(Y) &= -\sum_{i=\mathrm{Y,N}} P(Y=y_i)\log_2 P(Y=y_i) \\ &= -0.3\log_2 0.3 - 0.7\log_2 0.7 \\ &= 0.8813 \end{aligned}$$

Can you do better than this by knowing another attribute?

IG

Back to the credit risk example

$$
\begin{aligned}
H(Y|X) &\equiv -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x) \\
&= -\sum_x P(x) \sum_y P(Y=y|X=x) \log_2 P(Y=y|X=x) \\
&= -\sum_x P(x) \sum_y H(Y|X=x)
\end{aligned}
$$

Predicting credit risk

<2 yrs	missed	def?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

$$
\begin{aligned}
H(\text{defaulted}|<2\text{years}=\text{N}) &= -\frac{4}{4+2}\log_2\frac{4}{4+2} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183 \\
H(\text{defaulted}|<2\text{years}=\text{Y}) &= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8133 \\
H(\text{defaulted}|\text{<2years}) &= \frac{6}{10}0.9183 + \frac{4}{10}0.8133 = 0.8763
\end{aligned}
$$

Average entropy given value of "<2years" attribute

$$
\begin{aligned}
H(\text{defaulted}|\text{missed}=\text{N}) &= -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5917 \\
H(\text{defaulted}|\text{missed}=\text{Y}) &= -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.9183 \\
H(\text{defaulted}|\text{missed}) &= \frac{7}{10}0.5917 + \frac{3}{10}0.9183 = 0.6897
\end{aligned}
$$

Average entropy given value of "missed" attribute

IG

- We now have the entropy - the minimal number of bits required to specify the target attribute:

$$H(Y) \quad = \quad \sum_y P(y) \log_2 P(y)$$

- The conditional entropy - the remaining entropy of Y knowing X

$$H(Y|X) \quad = \quad -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x)$$

Attribute X has different values represented by x
y represents the possible class labels

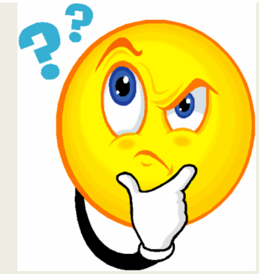
- So we can now define the reduction of the entropy after learning Y.
- This is known as the *mutual information* between Y and X

$$I(Y;X) = H(Y) - H(Y|X)$$

Original Entropy

Average entropy after sorting on attribute X

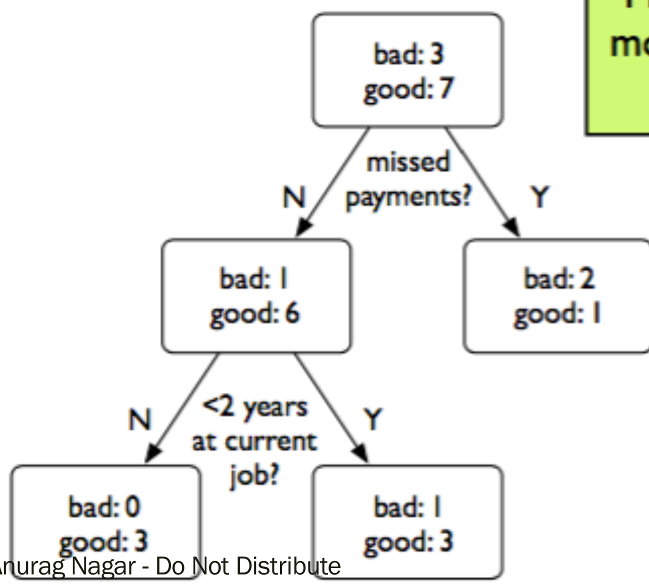
So... which attribute should I split on?



$$\begin{array}{rcl} H(\text{defaulted}) & - & H(\text{defaulted}|< 2 \text{ years}) \\ 0.8813 & - & 0.8763 = 0.0050 \end{array}$$

$$\boxed{\begin{array}{rcl} H(\text{defaulted}) & - & H(\text{defaulted}|\text{missed}) \\ 0.8813 & - & 0.6897 = 0.1916 \end{array}}$$

Missed payments are the most informative attribute about defaulting.

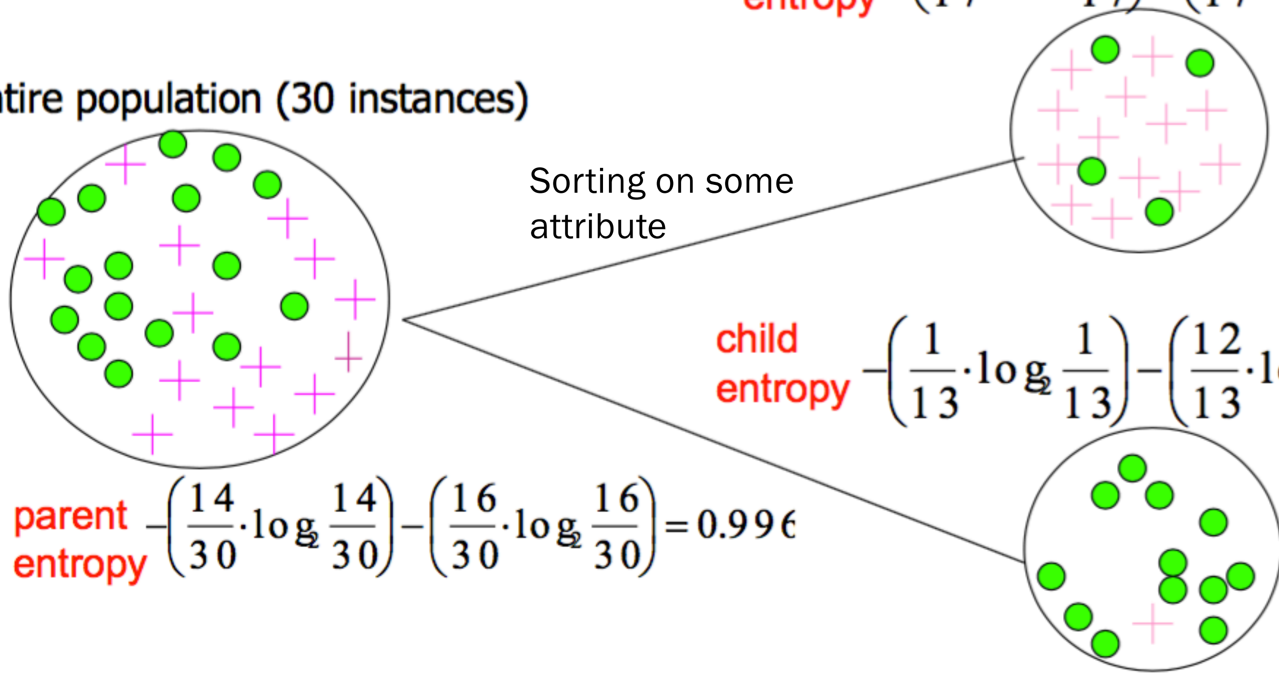


Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain = 0.996 – 0.615 = 0.38 for this split

Calculating IG

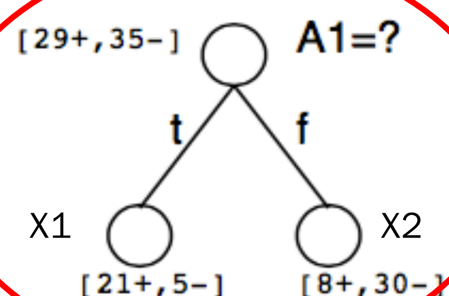
E(S) = E(29, 35)
E(X1) = E(21, 5)
E(X2) = E(30, 8)

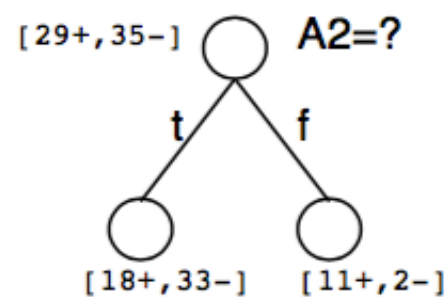
IG = E(S) –
    [26/64 * E(X1) + 38/64 * E(X2)]

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

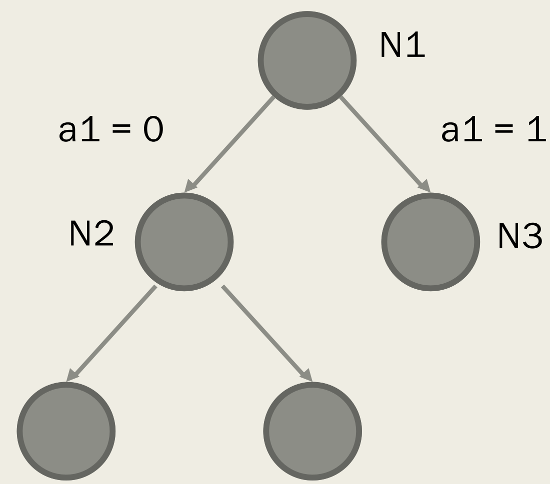
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$





Use IG to build a tree

Available attributes= {a1, a2, …, an}



- Start at the root of the tree (N1) and choose the best attribute according to IG. Let's say it is a1
- Create its children according to possible values of a1. If a1 is Boolean, it will have 2 values 0 and 1.
- These two nodes will be the leaf nodes for now -> since we haven't constructed whole tree yet.
- Also, these two nodes will have only less attribute to choose from. Their available choices will be {a2, …, an}
- Similarly create the rest of the tree, choosing the best attribute at each step.
- Terminating Condition:
  - The node is pure in terms of class label.
  - There are no more attributes remaining.

ID3 algorithm

Top-Down Induction of Decision Trees

[ID3, C4.5, Quinlan]

node = Root

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

one that gives the best IG

Note: These newly created nodes will be leaf nodes for now

ID3 algorithm

- Simpler version

ID3 (node, {attributes})

1. Let A = best attribute among {attributes} according to IG
2. Sort according to A and create child nodes (which will be leaf nodes for now)
3. For all the child nodes

    If it is pure or {attributes} is empty,

        STOP

    else

        call ID3(childNode, {attributes} – A)   /* Recursive Call */

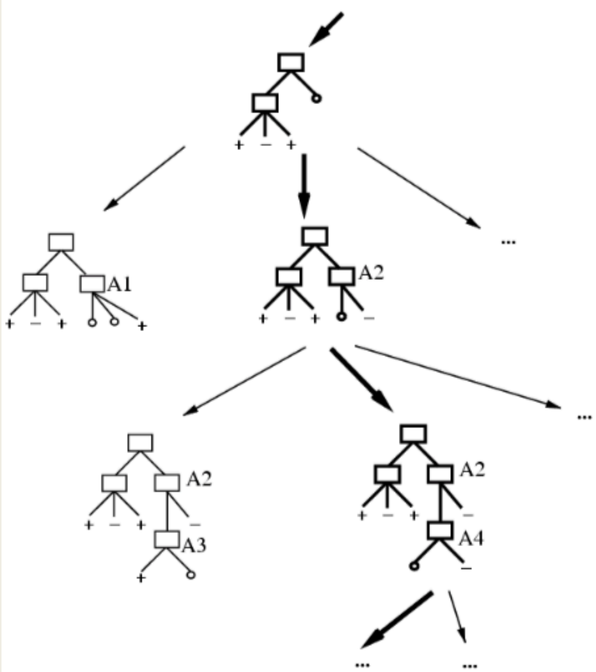
Worked out example

- Please see the handout/notes for a worked out example using ID3
- Remember:

- For each node, you have to find the best attribute
- You can only use an attribute once along a path. So, a node needs to inherit a list of attributes from its parent class
  -> You have to program this. ☺
- At the leaf node, find the majority class (by count). Use that for the prediction rule.

Does it really matter which attribute comes first?

- ID3 helps us in selecting the shortest i.e. most compact tree



- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

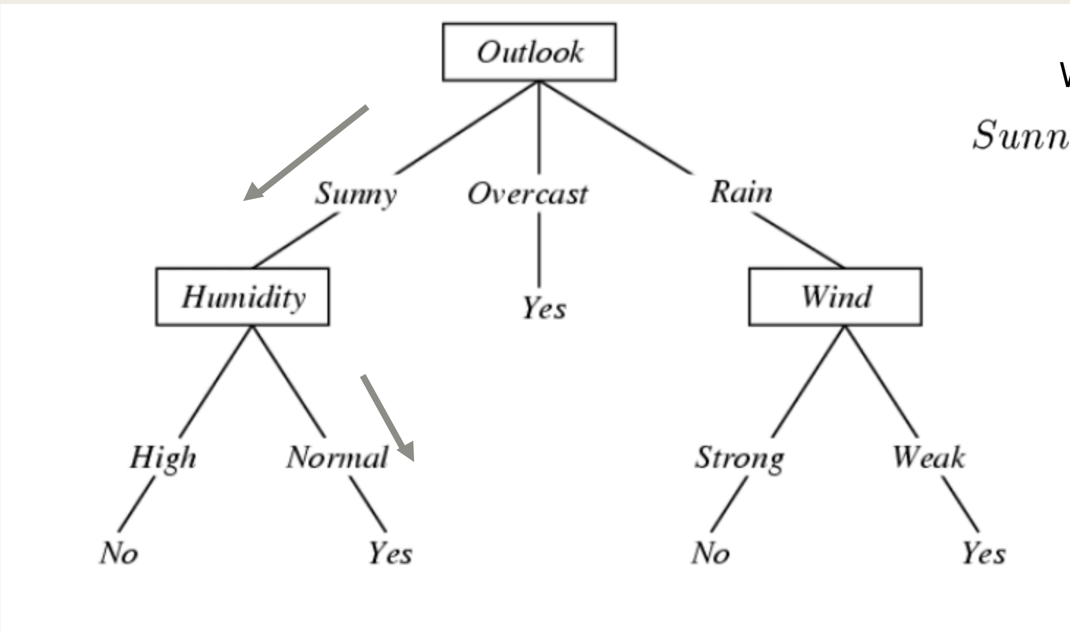
Occam's razor: prefer the simplest hypothesis that fits the data

ID3 is a greedy algorithm
Top-down induction of trees

Problem with DT

Problem with DT?

- Over-zealous learner -> learns all features
- What if there is noise?
- DT will try to change everything??
- Consider tree below:



What would happen if you get noisy data point

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

Overfitting

- You train on the training dataset

The data that the learner trains with.
=> It is possible to design a DT that gives 100% accuracy on training data. Think how??
e.g. each instance gets its own leaf node

- But is that a good thing?

=> NO! Because you are in fact memorizing (rote learning) the training data
=> No room for generalization, it's a case of Overfitting

- So, you have to find a balance between underfitting (learning very little) and overfitting.

- Notation:
  Training error of hypothesis h = $e_{train}(h)$
  True error (on unseen data) of hypothesis h = $e_{true}(h)$

Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

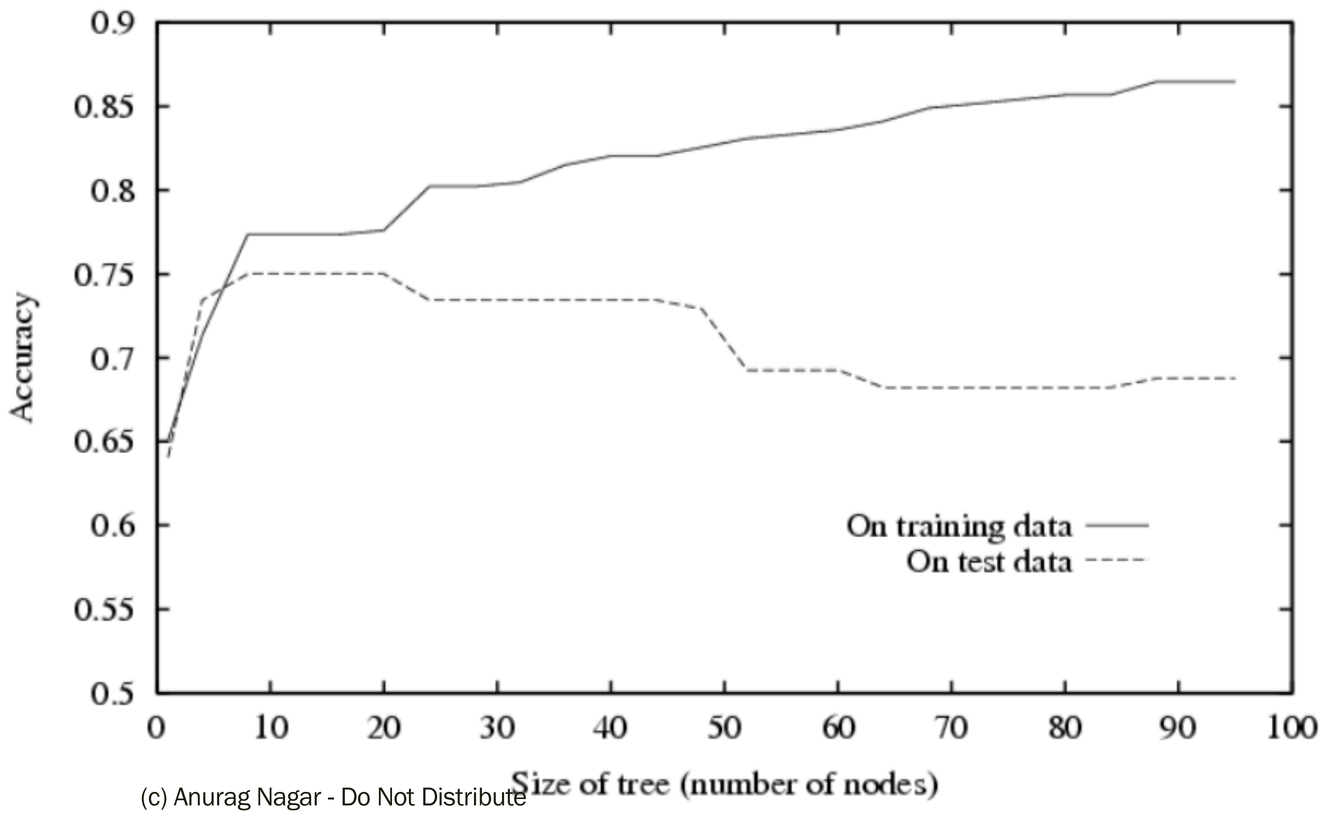
We say h <u>overfits</u> the training data if

$$error_{true}(h) > error_{train}(h)$$

Amount of overfitting =

$$error_{true}(h) - error_{train}(h)$$

Overfitting in Decision Tree Learning



How to avoid overfitting?

1. Stop growing when splits are not statistically significant (TOUGHER PROBLEM)

OR

2. Grow full tree then post-prune i.e. remove nodes and see if true error decreases (EASIER PROBLEM)

How to avoid overfitting?

- Keep another dataset -> validation dataset
- Build model on training, test accuracy on validation
- Learn model from training dataset.
- Randomly remove nodes and see if validation accuracy improves

Reduced-Error Pruning

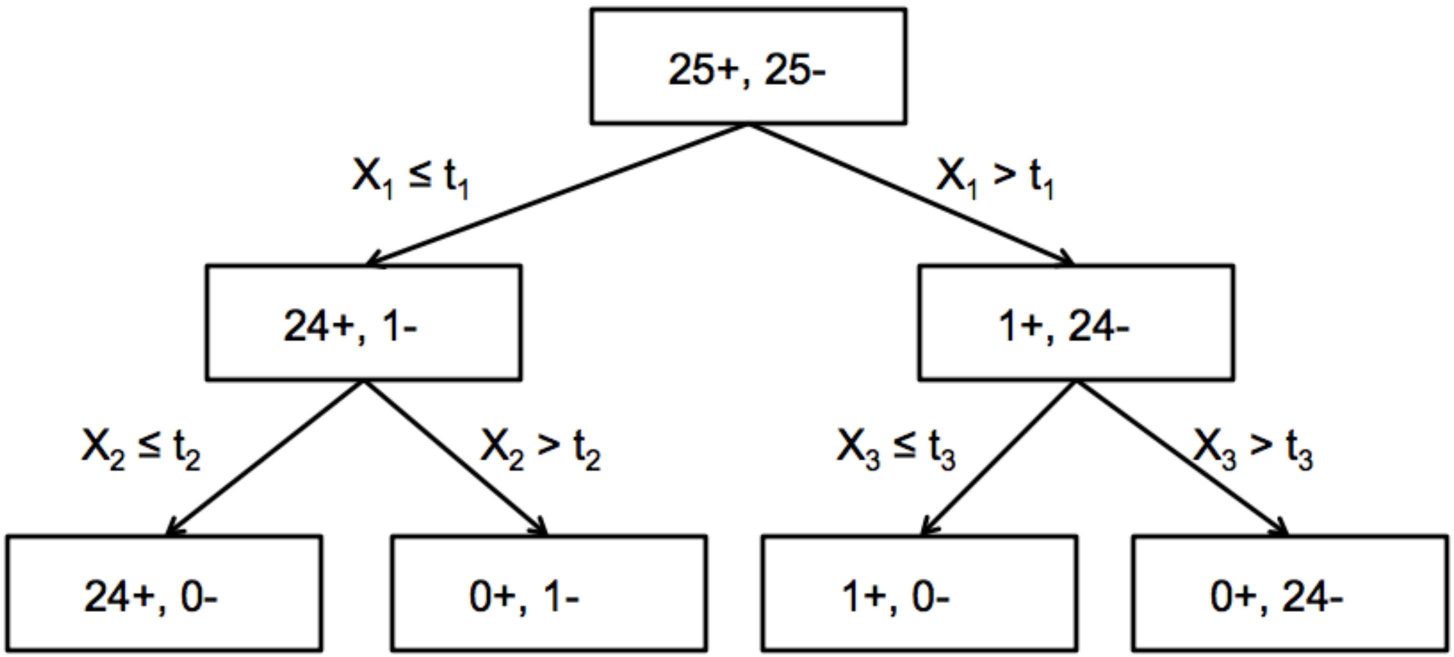
Split data into *training* and *validation* set

Create tree that classifies *training* set correctly

Do until further pruning is harmful:

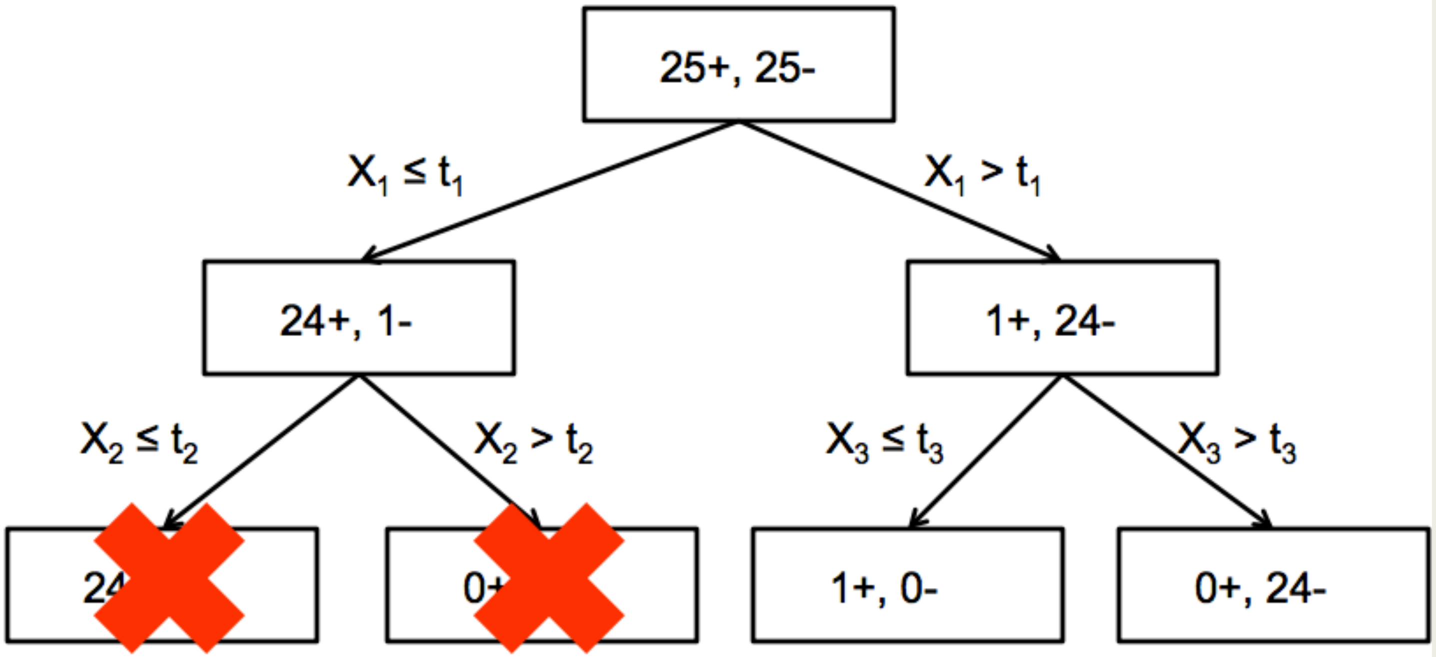
1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

Post-pruning



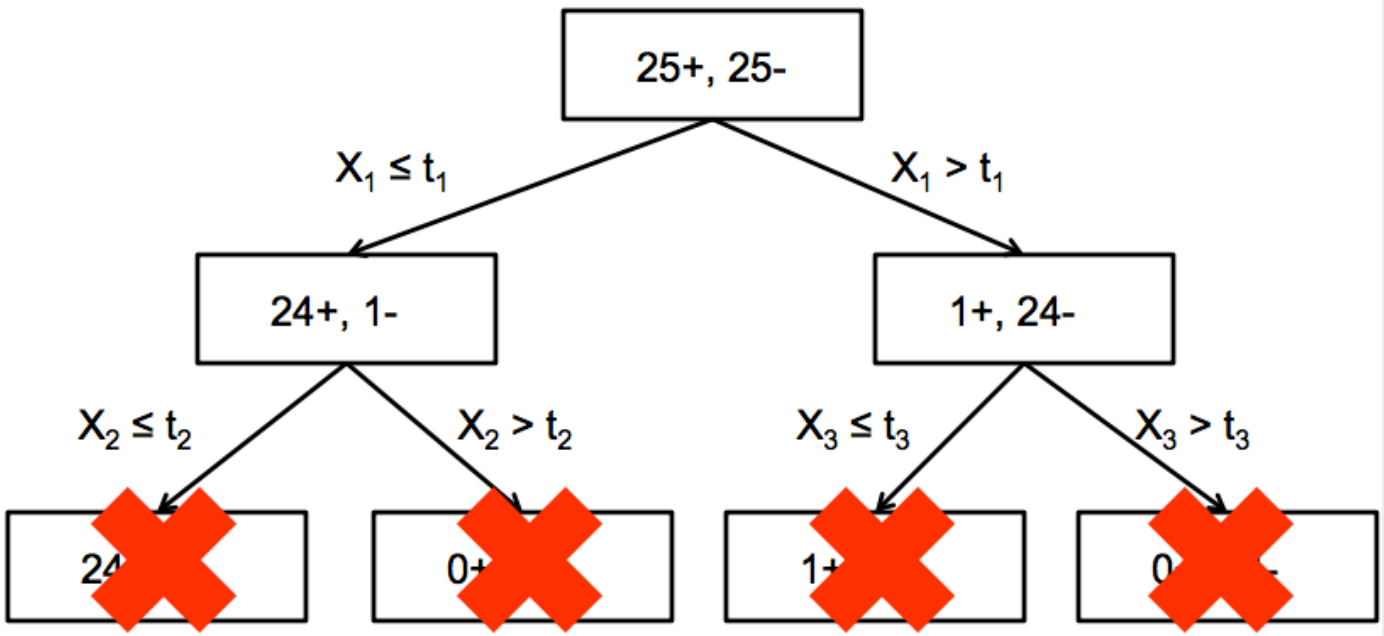
Validation Set Accuracy: 80%

Post-pruning



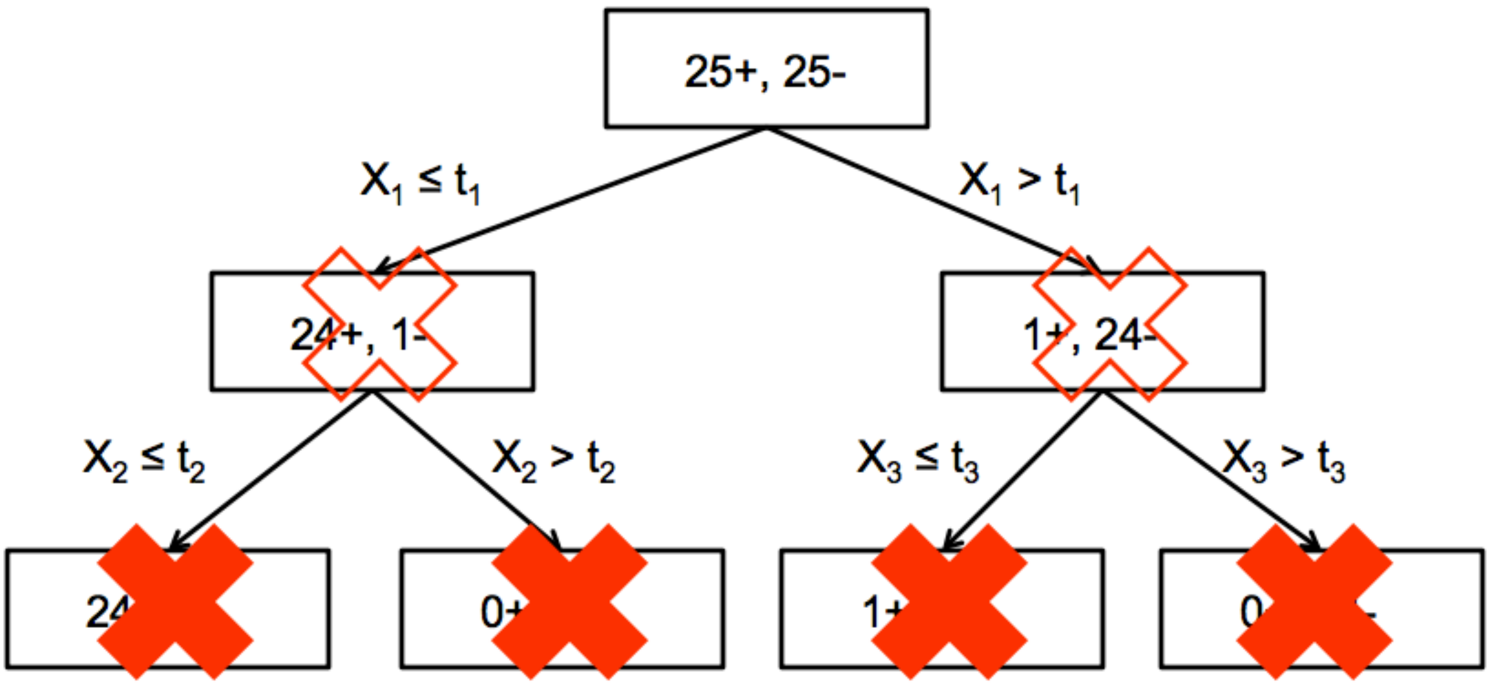
Validation Set Accuracy: 85%

Post-pruning



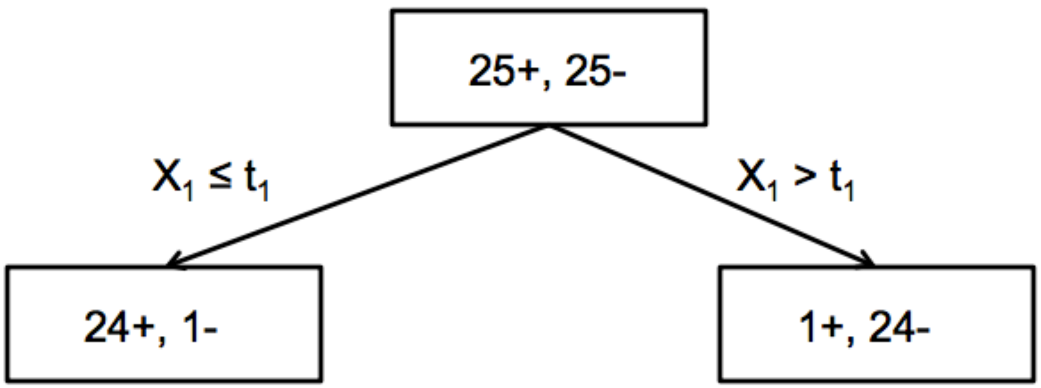
Validation Set Accuracy: 90%

Post-pruning



Validation Set Accuracy: 50%

Post-pruning



Final Decision Tree

What have we learnt?

- Idea of DT
- Number of instances (leaf nodes) and hypotheses
- How to choose best sorting attribute for each node
- How to induce top-down tree using ID3
- What is overfitting
- Avoiding overfitting