THEORY

OF

DECISION TREE

This page intentionally left blank

Decision Tree
Function Approximation

credits : Tom Mitchell

I. Problem Setting :

- Set of possible instances X [feature vector
- Unknown target function f. [discrete-valued for decision tree]
- Set of function hypotheses : $H = \{ h \mid h : X \to Y \}$ [set of all possible decision trees]

II. Input :

- Training examples $\{ \langle \overset{\nearrow\, i^{th}\text{ example}}{x^{(i)}}, y^{(i)} \rangle \}$ of unknown target function f

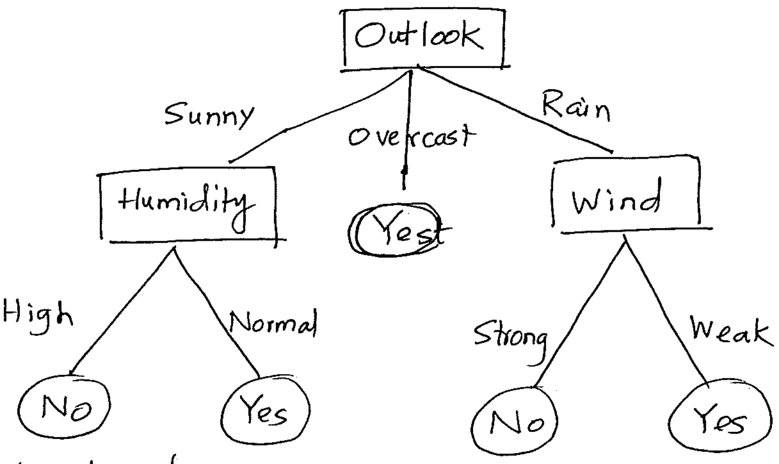
III. Output :

- Hypothesis $h \in H$ that best approximates f.

Example :

	X				Y
Day	Outlook	Temperature	Humidity	Wind	PlayTennis ?
	Sunny – 0	Hot – 0	High – 0	Strong – 0	Yes – 1
	Overcast – 1	Mild – 1	Normal – 1	Weak – 1	No – 0
	Rain – 2	Cold – 2			
Sample :(feature-label combination)					
1	0	1	0	1	0
2	1	0	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮

A Decision Tree for $f : \langle$ Outlook, Temperature, Humidity, Wind $\to$ PlayTennis $\rangle$



O = Sunny

T = Hot

H = Normal

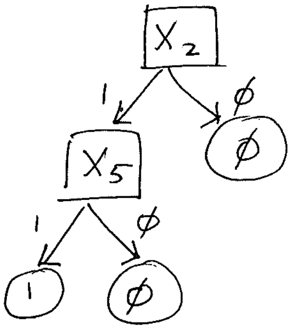
Wind = Weak / Strong

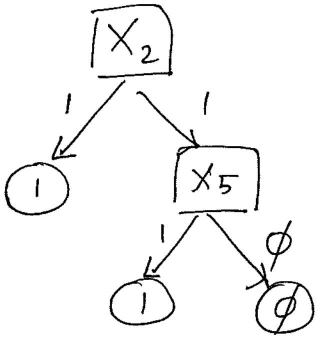
- Internal node : test one attribute X_i
- Branch from node : select one value for X_i
- Leaf node : predicts Y

Example Decision Trees

Suppose $X = \langle X_1, \cdots X_n \rangle$ $X_i \in \{0, 1\}$ boolean

Q.1. How would you represent $Y = X_2 \cdot X_5$ and $Y = X_2 \vee X_5$



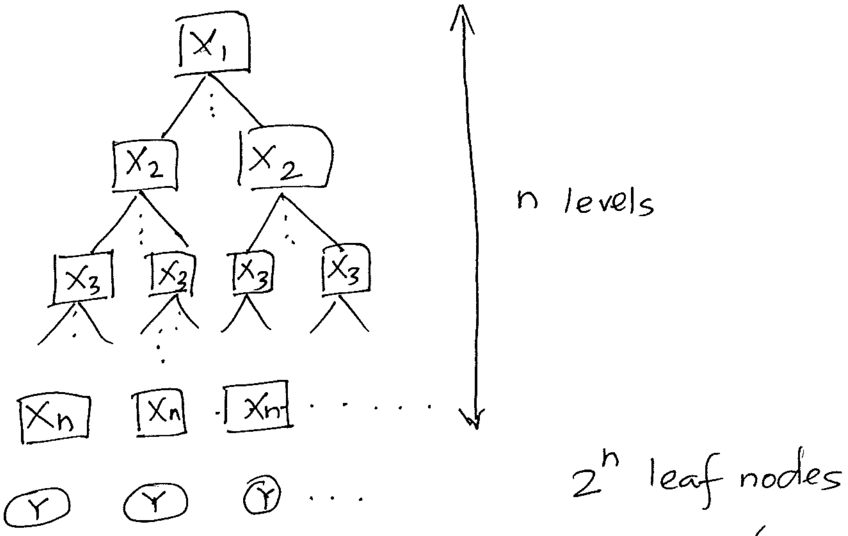


$\cdot \equiv$ AND
$\vee \equiv$ OR
$\neg \equiv$ NOT

or a more complicated one $Y = X_2 X_5 \vee X_3 X_4 (\neg X_1)$

Q.2. Can we represent arbitrary boolean (or discrete-valued ?) functions using decision trees?

Yes!



- So decision trees are expressive (can represent any boolean (discrete-valued) function)
- This makes decision trees universal function approximator.

Top-Down Decision Tree Learning : ID3

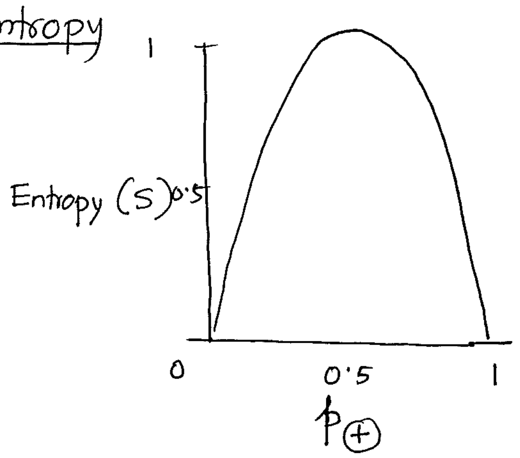
node = Root [ID3, C4.5, Quinlan]

Main Loop:

1. A ← 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 traing examples to leaf nodes
5. If traing examples are perfectly classified, STOP.
   else iterate over new leaf nodes

- Top-down greedy growth of decision tree using "best" attribute until all examples are perfectly classified
- Which attribute is best?

Sample Entropy



- S is a sample of training examples
- $p_\oplus$ is proportion of +ve examples
- $p_\ominus$ is proportion of -ve examples
- Entropy measures impurity of S

$$H(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

all +ve examples → entropy is 1
all -ve examples → entropy is 1
$p_\oplus = p_\ominus$ → entropy is 0.5

Entropy

Entropy $H(X)$ of r.v. X

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

$H(X)$ is based on information theory:

- Most efficient possible code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is

$$\sum_{i=1}^{n} P(X=i) - \log_2 P(X=i)$$

- How much H is reduced if we sort data by attributes
- choose the attribute which reduces entropy the most

Some Notations

Specific conditional entropy $H(X \mid Y=v)$ of X given $Y=v$:

$$H(X \mid Y=v) = -\sum_{i=1}^{n} P(X=i \mid Y=v)\log_2 P(X=i \mid Y=v)$$

Conditional entropy $H(X \mid Y)$ of X given Y:

$$H(X \mid Y) = \sum_{v \in \text{values}(Y)} P(Y=v)\, H(X \mid Y=v)$$

Mutual Information (a.k.a. information gain) of X and Y:

$$I(X, Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$

Information gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

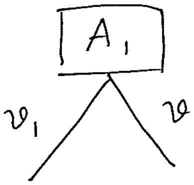
$$\text{Gain}(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y \mid A)$$

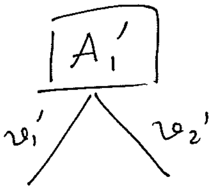
choose the attribute with highest Gain

Selecting the "best" attribute

$S : [x_1+, x_2-]$

Sample Entropy : $E_\phi = ?$





$S_1:[x_3+, x_4-]$ $S_2:[x_5+, x_6-]$

Conditional Entropy : $E_1 = ?$ $E_2 = ?$

$S_1' : [x_3'+, x_4'-]$ $S_2' : [x_5'+, x_6'-]$

$E_1' = ?$ $E_2' = ?$

$Gain(S, A') = G'$

$$= E_\phi - \frac{|S_1'|}{|S|} E_1' - \frac{|S_2'|}{|S|} E_2'$$

$Gain(S, A_1) = G$

$$= E_\phi - \frac{|S_1|}{|S|} \cdot E_1 - \frac{|S_2|}{|S|} \cdot E_2$$

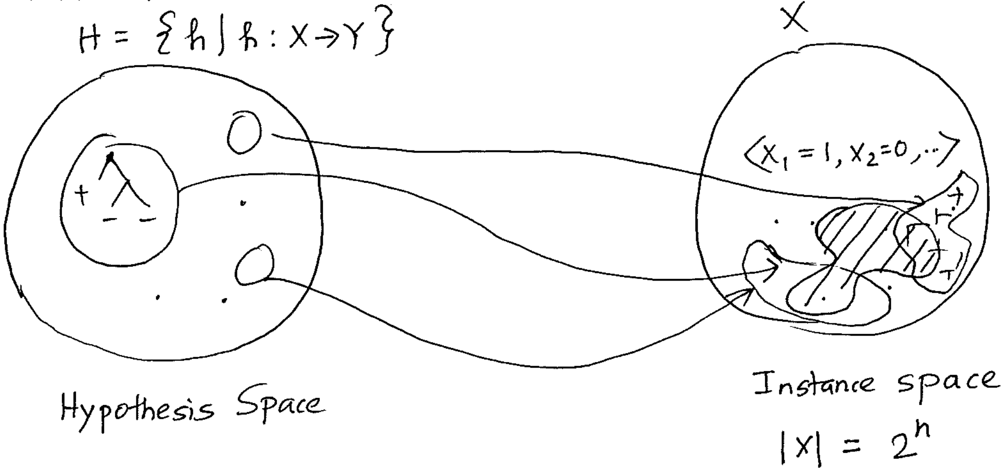
- Find whether G is greater than G'
- If so, choose A_1 over A_1', else : choose A_1'

Q. Is there more than one DT that will perfectly sort the data? If yes, which one do you choose?

A. Function Approximation : Big Picture

$f : X \to Y$ $X = \langle x_1, x_2, \cdots x_n \rangle$ $x_i \in \{0,1\}$ $Y \in \{0,1\}$

$H = \{h \mid h : X \to Y\}$



$|X| = 2^n$

$\#$ of DT's that can represent all possible functions $= 2^{2^n}$

$\#$ possible functions $= 2^{|X|} = 2^{2^n}$

sad fact about inductive inference

No Free Lunch !

PAC learning

$\#$ training examples we need to label so that there is just one DT (unique) in the hypothesis space $= \boxed{\text{All}}$

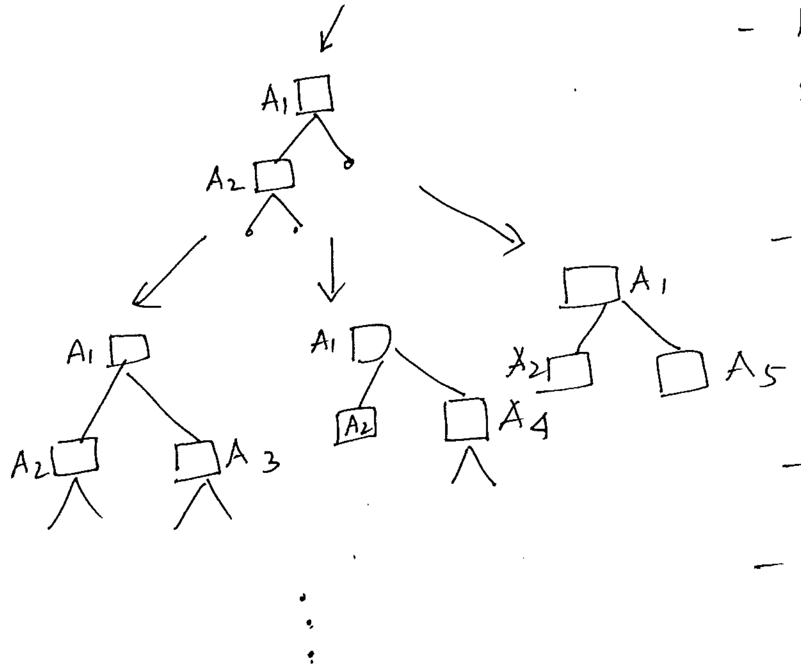
"All models are wrong" – George Box 1976. J. American Stat. Ass

In practical ML applications, we don't have all possible labeled traing data. So, we make some assumptions.during learning.

Q. In decision tree learning algorithm (ID3), what's the assumption?

A. It stops at the smallest acceptable tree.

Function Approximation as Search for the best hypothesiss:



- ID3 performs heuristic search through the space of decision trees
- It stops at smallest acceptable tree. Why?
- William of Ockham ~1300
- Occam's razor: prefer the simplest hypothesis that fits the data

All models are wrong; some are useful

Measuring Accuray of a trained model using a test data

- Accuracy/Success
- Error rate/ failure

		True Condition	
		+	−
Predicted condition	+	TP	FP
	−	FN	TN

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$MCC = \frac{TP . TN - FP . FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Overfitting in Decision Tree

- Fits the training data very well, but does not perform very well on test data

Overfitting can happen due to:

- Noisy traing example too.

Generally,

consider a hypothesis h and

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

h overfits the training data if

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

Amount of overfitting = $error_{true}(h) - error_{train}(h)$

Avoiding overfitting in Decision Tree : 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 node (and below)
2) Greedily remove the one that most improves validation accuracy

Guranteed to produce smallest version of most accurate subtree.