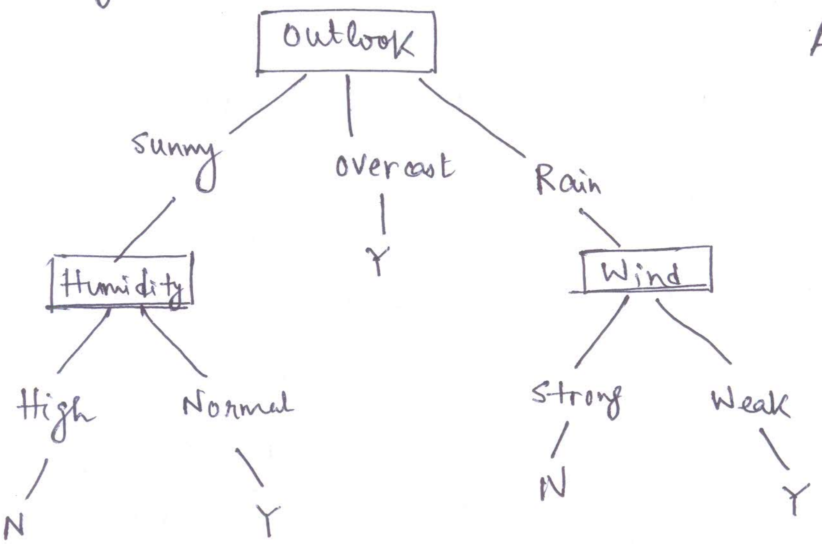
T:

# Decision Trees (DT) — ML-Lecture 2



Will I play tennis?




A data structure built on heuristics (Entropy) & Information Gain

Consider the logic function: $(\underbrace{A \vee B}_{A \text{ OR } B}) \wedge (A \vee C)$ we shall use for DT representation

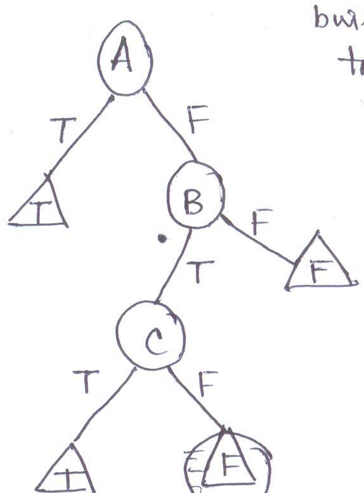
Ex:

| A | B | C | ? |
|---|---|---|---|
| T | T | T | T |
| F | T | F | T |
| T | F | T | T |
| T | F | F | T |
| F | T | T | T |
| F | T | F | F |
| F | F | T | F |
| F | F | F | F |

training data

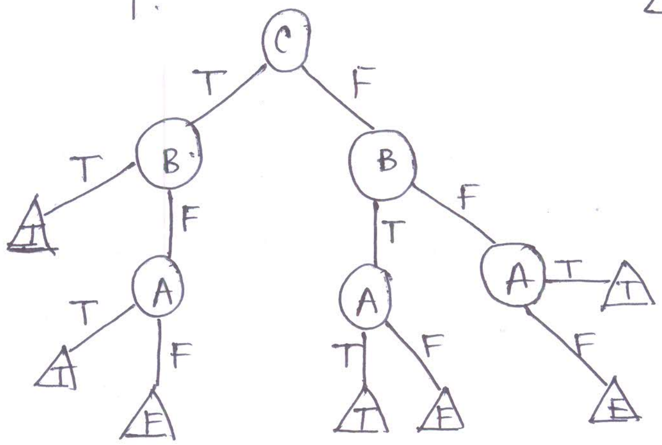
map this

Note: Whenever A is true, outcome is always true! ⇒ useful when we build a tree!




However, we can have another tree w/ node C at the top!

more complicated tree!




Occam's Razor

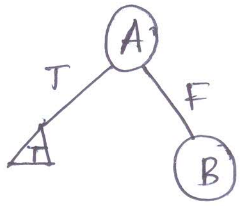
i) We want a smaller tree → less expensive computationally
ii) We want a systematic way to guess how to build it! → which one is the root node

Quinlan's Brilliant Paper on DT (must read) → Iterative Dichotomizer (ID3 – 1986)

Building a tree top-down

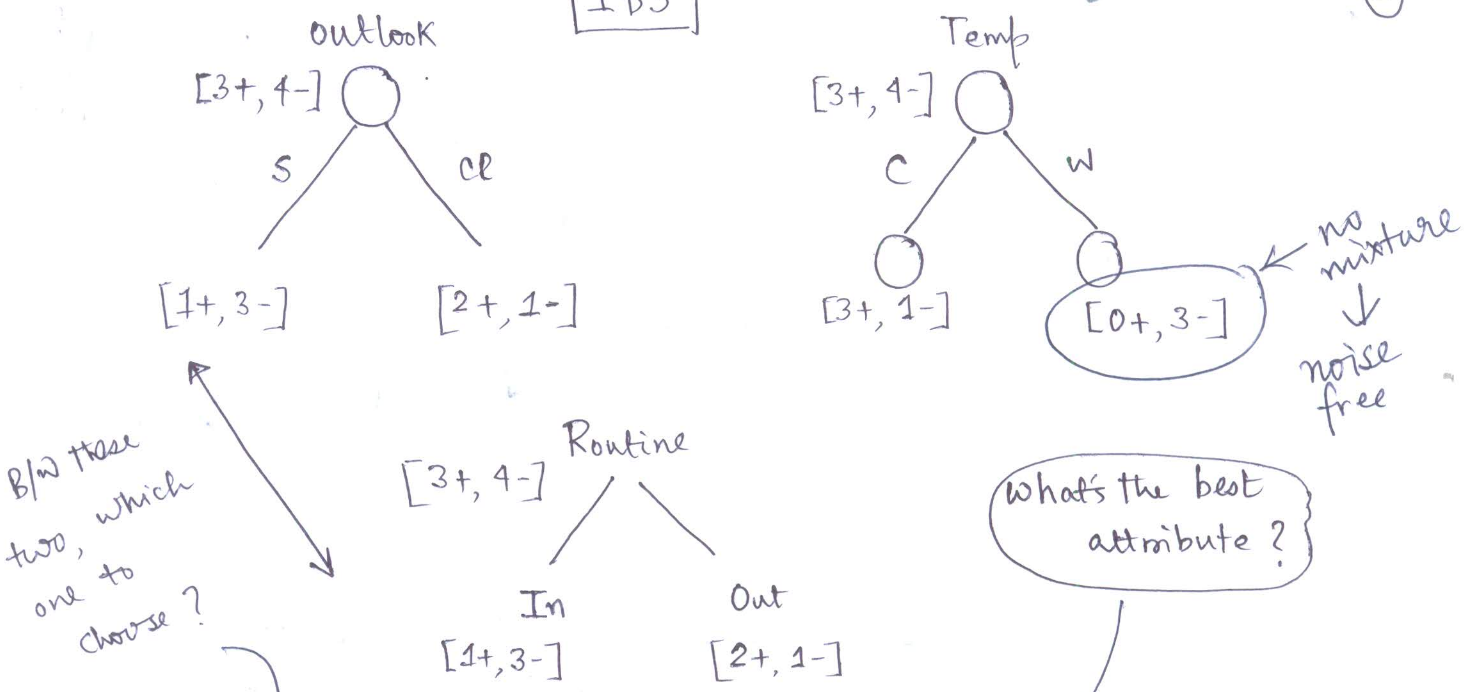
1. A ← the "best" decision attribute for next node ← How?
2. Assign A as decision attribute for next node
3. For each value of A, create new descendant of (node) (outlook < sunny, overcas[t], rainy)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, STOP, else iterate over new leaf nodes.

attribute classifies training data perfectly → put the value down




| Outlook | Temp | Routine | WearCoat |
|---|---|---|---|
| Sunny | Cold | InDoors | N |
| Sunny | warm | OutDoors | N |
| Cloudy | warm | InDoors | N |
| Sunny | warm | InDoors | N |
| Cloudy | Cold | InDoors | Y |
| Cloudy | Cold | OutDoors | Y |
| Sunny | Cold | OutDoors | Y |

# ID3




**Entropy:** Measure of chaos $\longrightarrow$ high entropy $\Rightarrow$ high chaos

we want to reduce entropy (chaos) by choosing the correct attribute (temp $\rightarrow$ warm).

→ Measures the impurity of samples; $S$ is a sample of training examples, $i \in$ set of outcomes (binary or otherwise)

→ For outcomes $i$: 1 to $n$

$$\text{Information Entropy}(S) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

($n = 2$, binary classification)

(prob of outcome)

$$\boxed{\text{Attribute w/ the lowest entropy will be chosen}}$$

**Note:** if $p_i = 0$, problem ~ set $0 \cdot \log 0 = 0$ → why? $x \log x \to 0$ as $x \to 0$ from 0

**Definition:** Entropy ($S$) = expected # of bits needed to encode class (⊕ or ⊖) of randomly drawn member of $S$ (under the optimal, shortest length code)

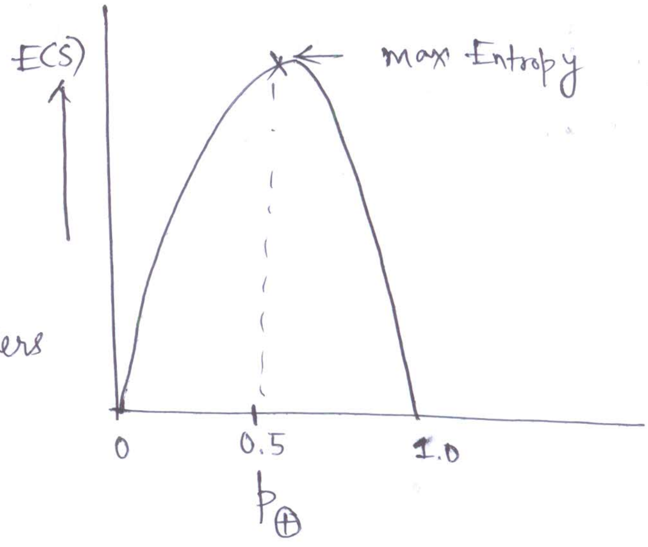
Why? optimal length code assigns $-\log_2 p$ bits to message having probability $p$

So, expected # of bits to encode ⊕ or ⊖ (two outcomes) of random member of $S$: $p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$ (Information Theory)

1) Entropy lies b/w 0 & 1

2) Max entropy is achieved when all outcomes are equally likely

3) Min entropy is achieved when one outcome is certain & the others are not.




Proof: Recall, $\log_2 p = \dfrac{\log p}{\log 2}$;

$$E(S) = -\frac{1}{\log 2}\left(p \log p + (1-p)\log(1-p)\right); \quad \begin{cases} p_\oplus = p \\ p_\ominus = 1-p \end{cases}$$

$$\frac{dE}{dp} = -\frac{1}{\log 2}\left(\log p + 1) - \log(1-p) - 1\right)$$

$$= -\frac{1}{\log 2}\log\left(\frac{p}{1-p}\right)$$

$dE/dp = 0 \Rightarrow p = 1-p \Rightarrow p = \frac{1}{2} \rightarrow$ critical point

$$d^2E/dp^2 = -\frac{1}{\log 2}\frac{d}{dp}\left(\log p/1-p\right) = -\frac{1}{\log 2}\frac{d}{dp}\left(\log p - \log(1-p)\right)$$

$$= -\frac{1}{\log 2}\left(\frac{1}{p} + \frac{1}{1-p}\right)$$

$$= -\frac{1}{\log 2}\,\frac{1}{p(1-p)}$$

$p = \frac{1}{2}$ is a maxima $\Leftarrow$ $< 0$ for $p = \frac{1}{2}$

$\downarrow$

E(S) is maximum when prob of classifying '+' examples i.e ~~neg~~ '−' examples is equally likely!!

Note: This is useless $\rightarrow$ truth table example; imagine an attribute where half the time the outcome is true & false the other half of the time $\Rightarrow$ can't use that information for classification or decide whether it's a good attribute (B & C in TT but A is good since $A \rightarrow T \Rightarrow$ outcome $\equiv$ T all the time)

Small or huge probability of classifying ~~so~~ examples as '+', entropy is low ⇒ useful !!

Consider a two-class problem; (+ & − training examples only)

$$E(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log p_{\ominus}$$

Next; how to reduce entropy? (High entropy is a problem ....)

Information Gain:

$$\text{Gain}(S, A) = \text{Expected reduction in entropy due to sorting on } A$$

$$= \text{Entropy}(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

where $S$ is a set of examples, $A$ is an attribute, $S_v$ is the subset of $S$ for which attribute $A$ has value $v$. Values(A) are all possible values that attribute A can take.

Ex: Routine < InDoors, OutDoors = values (A) ; A ≡ routine

For each of the attribute A, in Gain (S, A), we'll count how many examples are there in the set for that value

Values(outlook) = {Sunny, Cloudy}

$S = [3+, 4-]$

$S_v$ $\begin{cases} S_{Sunny} = [1+, 3-] \\ S_{Cloudy} \equiv [2+, 1-] \end{cases}$

$(S_{sunny} + S_{Cloudy})$

$$\text{Gain}(S, \text{outlook}) = \text{Entropy}(S) - \sum_{v \in \{Sunny, Cloudy\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

tip $\left\{ \boxed{\begin{array}{l} [10+, 0-] \Rightarrow E(S) = 0 \\ [0+, 10-] \Rightarrow E(S) = 0 \\ [5+, 5-] \Rightarrow E(S) = 1 \end{array}} \right.$

| Day | Outlook | Temp | Humdity | Wind | PlayTennis |
| --- | --- | --- | --- | --- | --- |
| D1 | Su | Hot | High | Weak | N |
| D2 | Su | Hot | High | Strong | N |
| D3 | O | Hot | High | Weak | Y |
| D4 | R | Mild | High | Weak | Y |
| D5 | R | Cool | Normal | Weak | Y |
| D6 | R | Cool | Normal | Strong | N |
| D7 | O | Cool | Normal | strong | Y |
| D8 | Su | Mild | High | Weak | N |
| D9 | Su | Cool | Normal | Weak | Y |
| D10 | R | Mild | Normal | weak | Y |
| D11 | Su | Mild | Normal | strong | Y |
| D12 | O | Mild | High | strong | Y |
| D13 | O | Hot | Normal | Weak | Y |
| D14 | R | Mild | High | strong | N |

Q: Can I build a DT? Which attribute is the best classifier?

~~Lets choose humidity~~;

Compute & explain explicitly

S: [9+, 5-] ; E = 0.940

$\boxed{\text{Humidity}}$

High → [3+, 4-], E = 0.985

Normal → [6+, 1-], E = 0.592

S: [9+, 5-]
E: 0.940

$\boxed{\text{wind}}$

weak → [6+, 2-], E = 0.811

strong → [3+, 3-], E = 1.0

$$\text{Gain}(S, \text{humidity}) = \text{Entropy}(\text{Sample}) \equiv E(S) - \sum_{v \in \{\text{high}, \text{normal}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$= 0.940 - \left(\frac{7}{14}\right) \text{Entropy}(\text{high}) - \left(\frac{7}{14}\right) \text{Entropy}(\text{normal})$$

$$= 0.940 - \frac{7}{14} * 0.985 - \frac{7}{14} * 0.592 = 0.151$$

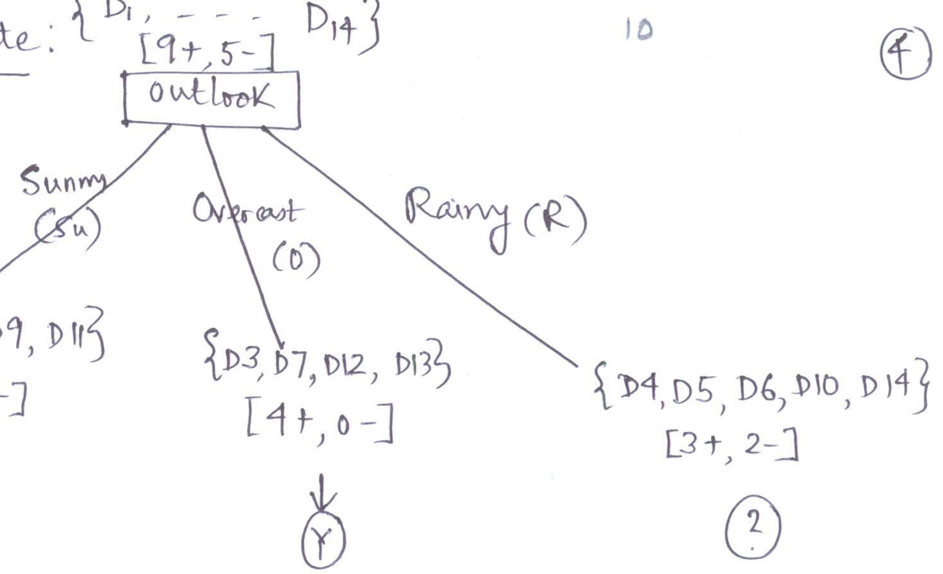
Gain(S, outlook) = 0.246
Gain(S, Humidity) = 0.151
Gain(S wind) = 0.048

⟹ pick outlook because gain is the largest

; Select the next attribute: {D1, ---- D14}

10

(4)




this → becomes dataset, S

which attribute should be tested here?

$S_{sunny} = \{D1, D2, D8, D9, D11\}$

identical results if we use $S_{rainy}$ in place of $S_{sunny}$

$Gain\{S_{sunny}, Humidity\} = 0.970 - (3/5)0 - (2/5).0. = 0.970$ ✓ pick

$Gain\{S_{sunny}, Temp\} = 0.970 - (2/5)0.0 - (2/5)1.0 = .570$

$Gain\{S_{sunny}, Wind\} = 0.970 - (2/5)1.0 - (3/5).918 = 0.019$

→ $S_{sunny}$, Humidity → do this recursively ✓ → ~~$S_{sunny}$~~, overcast (not required) → (rainy)

Prefer "shortest tree" → unbiased?

A) Short trees, & for those w/ high information gain attributes near the root

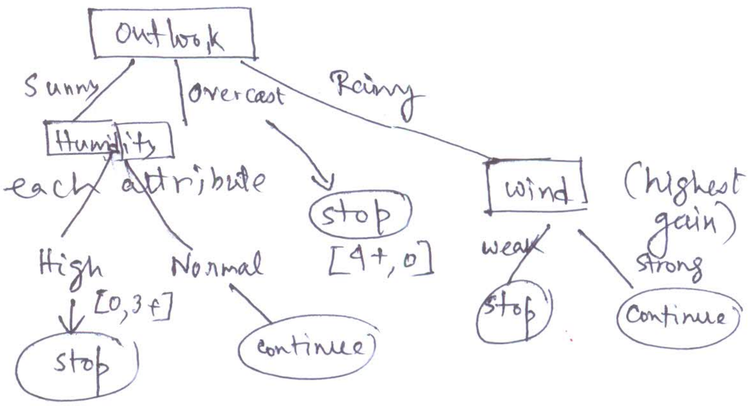
B) Bias → preference for some hypotheses in H.

---

ID3 Summary: 1) pick the attribute w/ the highest gain as root node.

2) pick the one w/ the highest gain as root node for each attribute value of the root node.

3) Keep checking until we reach stopping condition for all branches.



## Notes about Entropy:

X D.R.V, $H(x) = -\sum_{x \in X} p(x) \log_2 p(x)$

Recall,
$E(f(x)) = \sum_{x \in X} f(x)\, p(x)$
set $f(x) = -\log_2 p(x)$

1.) $0 \log 0 \rightarrow 0$

2) $H(x) \geq 0$ (why?)

3) $H(x) = E_p(-\log p(x))$
↑
Expectation

4) $H(x)$ depends on $p(x)$
prob. distribution

5) $H(x)$ is Concave

## Issues

A) overfitting → prune the tree!

B) Bias-variance trade off in DT can be derived for the # of leaves in the tree.
As we increase the # of leaves, the bias in the tree decreases (Recall: bias is always for the shortest tree) ⇒ As we grow the tree ⇒ bias decreases but variance may increase as well.

C) Use a diff set for validation to check for overfitting.

## How to select best tree?

a) Measure performance over training data

b) Measure performance over separate validation data set

c) Minimize size (tree) & Minimize (misclassifications)