Machine Learning

- Two Spaces for a Learner
  - **X** Example Space – all possible inputs
  - **H** Hypothesis (concept) Space
    - All possible outputs (concepts, patterns)
    - Unfortunately H is also the standard symbol for entropy…
- Mathematical spaces, defined by well-formedness constraints
- Each element of X is a collection of features
- Each element of H is a partition function over X

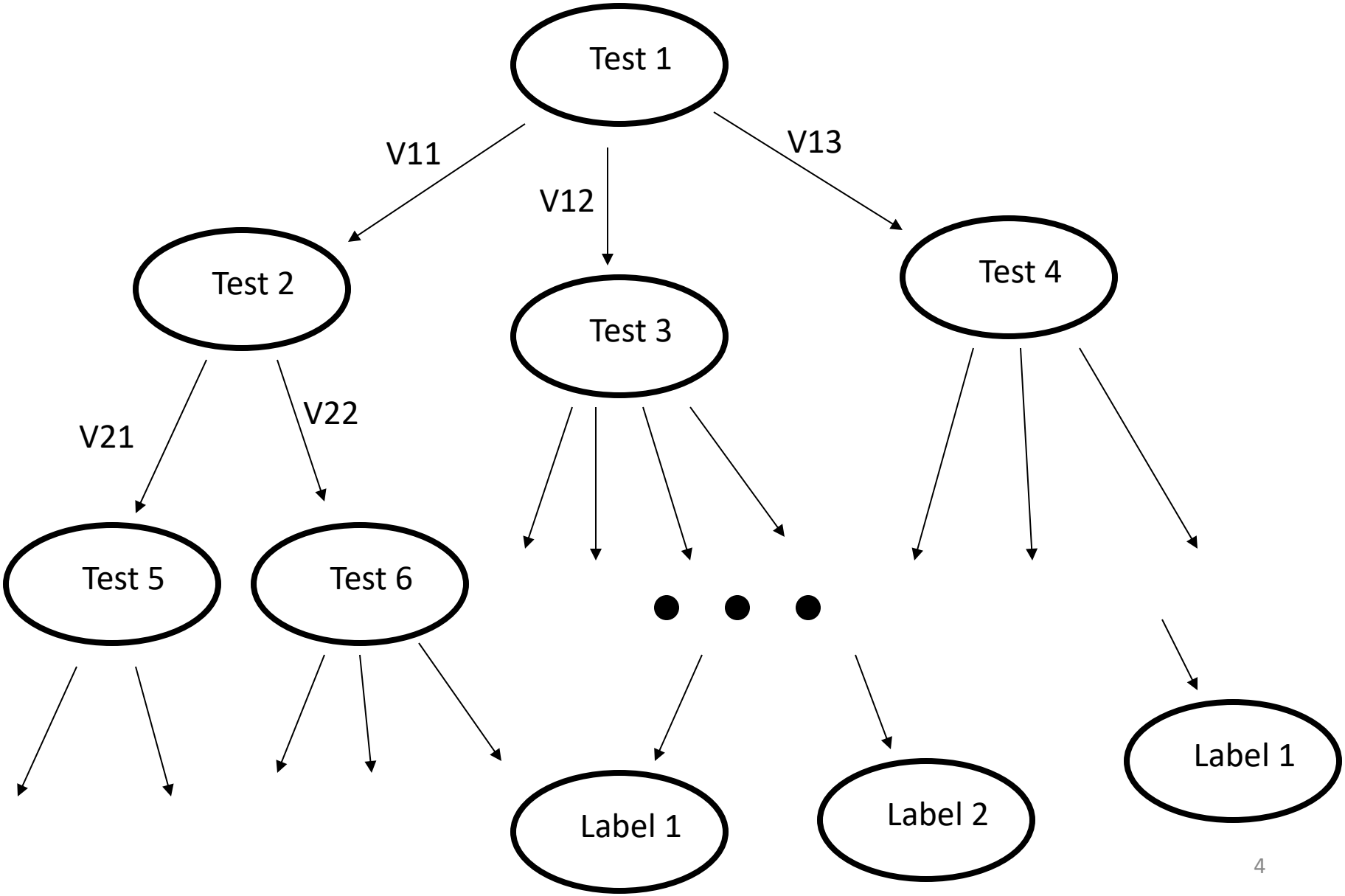
Distinctions

- Type of output
  - Classification
    - Binary
    - Multiclass
  - Regression
    - Learning a function
- Type of training
  - Supervised; examples all have training labels
  - Semi-supervised; some examples have training labels
  - Unsupervised (aka Clustering); no training labels

Decision Tree Learning

- Supervised multiclass classification
- H = decision trees
- Set of tests (splits)
  - Evaluated on the features of X
  - Finite set of outcomes
  - Applied to a set of X elements
  - Splits the set into subsets
- Construct a tree of tests that (fit / describe / summarize) the training examples

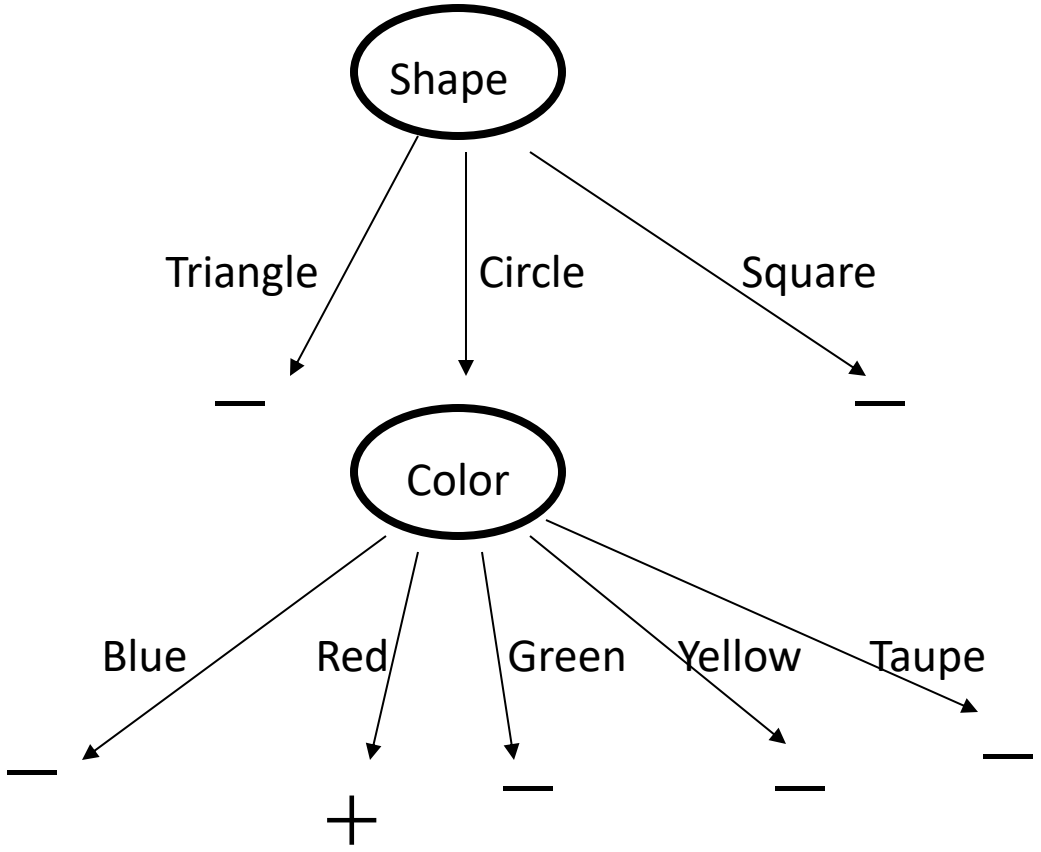
Decision Tree



Suppose I like circles that are red

(I might not be aware of the rule)

- Features:
  - Owner
    - John, Mary, Sam
  - Size
    - Large, Small
  - Shape
    - Triangle, Circle, Square
  - Texture
    - Rough, Smooth
  - Color
    - Blue, Red, Green, Yellow, Taupe



$$\forall x\,[\mathrm{Like}(x) \Leftrightarrow (\mathrm{Circle}(x) \wedge \mathrm{Red}(x))]$$

Decision Tree Learning by Hill Climbing

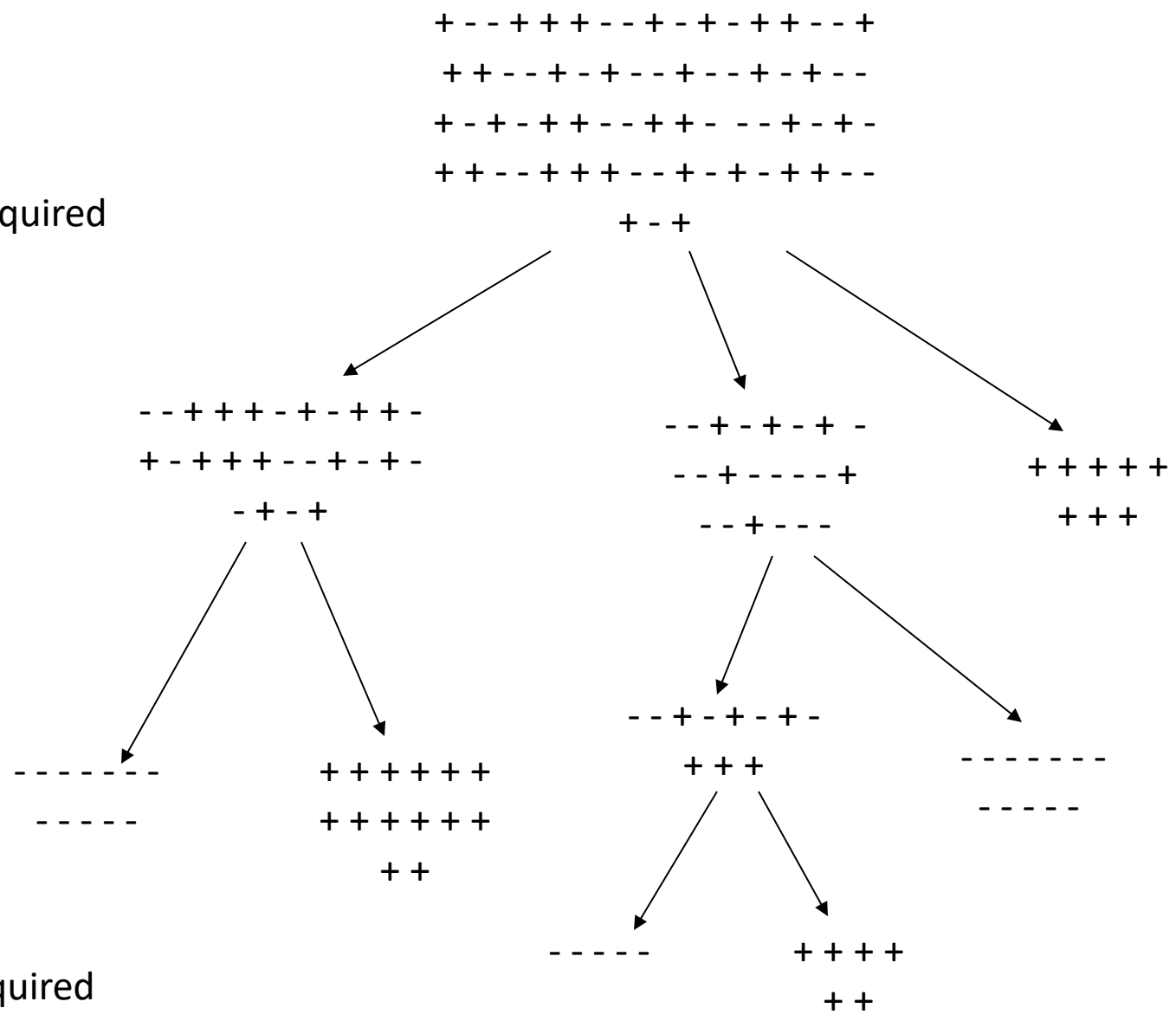
Initialize with all training examples in a set

1. STOP if the set is homogeneous (or good enough)
2. Choose most useful test on which to split
   - form subsets from the original set
   - useful = improved class label purity
3. Recur on the split subsets

- We will use *entropy* to measure purity
  - Many alternative
  - Choose to fit task
  - Often makes little difference

Training Data

Highly Disorganized
Low Purity
High Entropy
Much Information Required



Improved Purity
Lower Entropy

Highly Organized
High Purity
Low Entropy
Little Information Required

Greedy Search to Add Tests

- Improved homogeneity
  - Entropy reduction
  - Information gain
- To evaluate a split utility:
  - Measure entropy (information required) before
  - Measure entropy (information required) after
  - Subtract
- Information Required = Expected number of *bits* to communicate the label of an item chosen randomly from a set
- Often involves *fractional bits*
  - just a measure
  - not a memory configuration or number of flip-flops

Measuring Information

H denotes *Information Need* or *Entropy*

- H(S) = bits required to label some $x \in S$
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?

S_1 = + + +

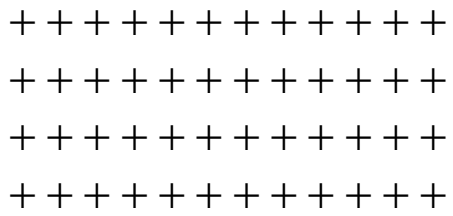
Measuring Information

- H(S) = bits required to label some x $\in$ S
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?
- What is $H(S_2)$? S_2 = − −
  − −

Measuring Information

- H(S) = bits required to label some $x \in S$
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?
- What is $H(S_2)$?
- What is $H(S_3)$?

S_3 =



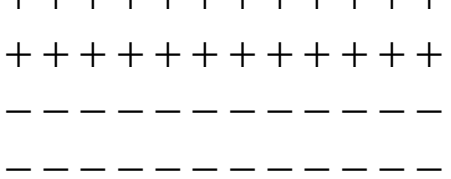
Measuring Information

- H(S) = bits required to label some $x \in S$
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?
- What is $H(S_2)$? S_4 = + −
- What is $H(S_3)$?
- What is $H(S_4)$?

Measuring Information

- H(S) = bits required to label some $x \in S$
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?
- What is $H(S_2)$?
- What is $H(S_3)$?
- What is $H(S_4)$?
- What is $H(S_5)$?

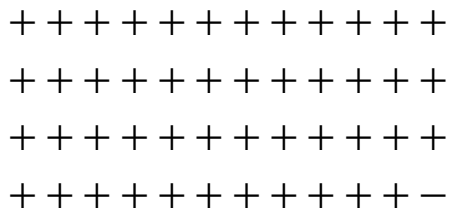
S_5 =



Measuring Information

- H(S) = bits required to label some $x \in S$
- What is the upper bound if label $\in$ {+,-}
- What is $H(S_1)$?
- What is $H(S_2)$?
- What is $H(S_3)$?
- What is $H(S_4)$?
- What is $H(S_5)$?
- What is $H(S_6)$?

S_6 =



Think of *expected* number of bits

$H(S_6)$ should be closer to 0 than to 1
Information theory / coding theory is relevant

Measuring Information

- What is $H(S_7)$?

S_7 =

F A B B A A B A D
A A A D A B E A F
A A B B A C A E B
A A A B C

=

A A A A A A A A A
A A A A A A A A A
B B B B B B B B B
C C D D E E F F

16

8

2 2 2 2

- H(S) = Expected bits *required* to label x, given that x is drawn randomly from S_7
- Labels $\in$ {A,B,C,D,E,F}; Six labels (instead of 2)
- Certainly 3 bits suffice, but are 3 required?

Measuring Information

- x is drawn randomly from S_7

S_7 =

F A B B A A B A D
A A A D A B E A F
A A B B A C A E B
A A A B C

=

A A A A A A A A A
A A A A A A A A A
B B B B B B B B B
C C D D E E F F

16

8

2 2 2 2

- Half of the time x is likely to be an A
- Half of the remaining time, a B; etc.
- We can improve on 3 bits

Measuring Information

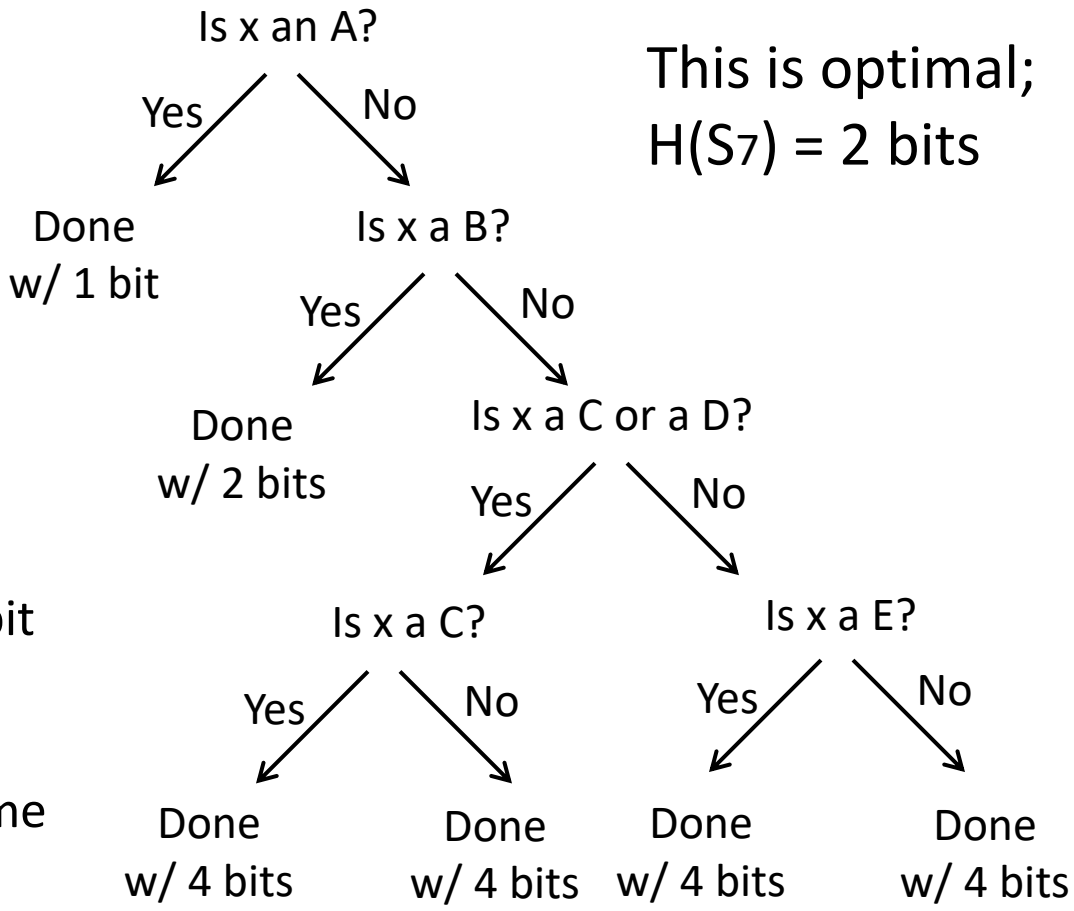
Do NOT think of this as a decision tree that we are trying to learn

S_7 =

F A B B A A B A D
A A A D A B E A F
A A B B A C A E B
A A A B C

=

A A A A A A A A	16
A A A A A A A A	
B B B B B B B B	8
C C D D E E F F	2 2 2 2



This is optimal;
$H(S_7) = 2$ bits

Half the time we are done in 1 bit
A quarter of the remaining time we are done in 2 bits
The remaining quarter of the time costs 4 bits

Expected number of bits: 0.5*1 + 0.25*2 + 0.25*4 = 2 bits !

Measuring Information

- But instead of asking Yes/No questions, now think coding theory
- Fixed set of messages: {A,B,C,D,E,F} w/ likelihoods as in S_7
  - A: No activity
  - B: Someone is crossing the street
  - C: Car accident
  - D: Bicycle accident
  - E: Water main break
  - F: Martians landing
- To transmit each message, we could send its 3 bit code:
  - 000 means A, no activity; 001 means B; etc.
- For n messages, we send 3*n bits. On average 3 bits / message
- But some messages are more likely than others
- We can use shorter codes for more likely messages
- After many messages we may hope to average < 3 bits / message

Measuring Information

Expected number of bits:

- 16/32 use 1 bit
- 8/32 use 2 bits
- 4 x 2/32 use 4 bits

$S_7 =$

A A A A A A A A	16
A A A A A A A A	
B B B B B B B B	8
C C D D E E F F	2 2 2 2

0.5(1) + 0.25(2) + 0.0625(4) +
0.0625(4) + 0.0625(4) + 0.0625(4)

= 0.5 + 0.5 + 0.25 + 0.25 + 0.25 + 0.25

= 2

FOR	SEND
A	1
B	01
C	0000
D	0001
E	0010
F	0011

$$H(S) = \sum_{v \in Labels} -\Pr(v) \cdot \log_2(\Pr(v))$$

H depends on *probabilities*
NOT counts

Measuring Information

- Other measures of non-homogeneity are sometimes used
- The Gini Index is popular
  - From economics
  - Measures income / wealth disparity in a population
  - What % of the population holds what % of the wealth
- We will use entropy

Information Gain w/ Entropy

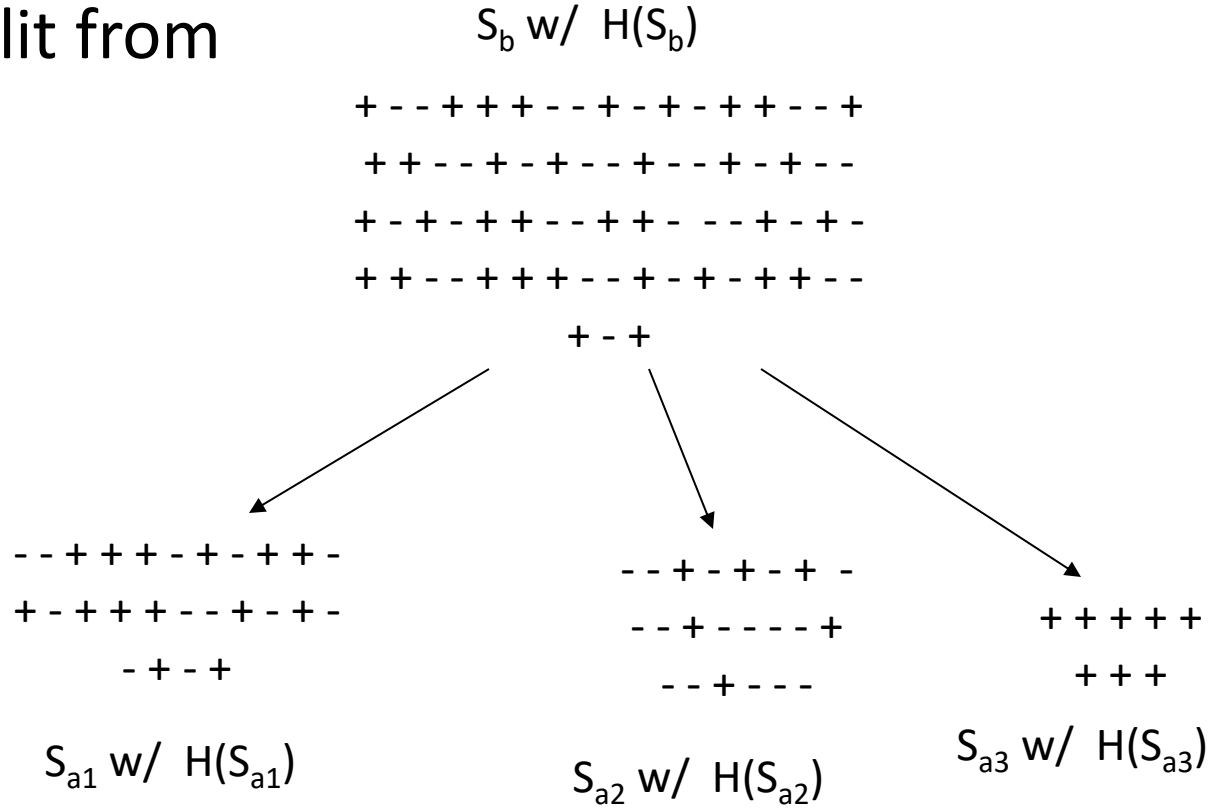
Subtract Information required after split from before

Information required:

Before $H(S_b)$

After $Pr(S_{a1}) \cdot H(S_{a1}) + Pr(S_{a2}) \cdot H(S_{a2}) + Pr(S_{a3}) \cdot H(S_{a3})$

Estimate probabilities using sample counts



Estimated Information Gain = $$\mathbf{H}(S_b) - \sum_i \mathbf{H}(S_{ai}) \frac{|S_{ai}|}{|S_b|}$$

Choosing the Most Useful Test

- Estimate information gain for each test

$$\mathbf{H}(S_b) - \sum_i \mathbf{H}(S_{ai}) \frac{|S_{ai}|}{|S_b|}$$

- Choose the highest

Example

- Restaurant example in text
- Tennis example

Will I Play Tennis?

- Features:
  - Outlook Sun, Overcast, Rain
  - Temp. Hot, Mild, Cool
  - Humidity High, Normal, Low
  - Wind Strong, Weak
  - Label +, -
- Features are evaluated in the morning
- Tennis is played in the afternoon

Machine Learning

- Example space:
  - Each example is a vector of features
  - Outlook, Temperature, Humidity, Wind
  - Such as:
    - Sun, Mild, Low, Strong
    - Overcast, Hot, High, Weak
- Class labels
  - I can play tennis +
  - I cannot play tennis: -
- Hypothesis space:
  - All decision trees
  - Each node tests one feature
  - As many node outcomes as values for that feature

Training Set

1.	S H H W	-
2.	S H H S	-
3.	O H H W	+
4.	R M H W	+
5.	R C N W	+
6.	R C N S	-
7.	O C N S	+
8.	S M H W	-
9.	S C N W	+
10.	R M N W	+
11.	S M N S	+
12.	O M H S	+
13.	O H N W	+
14.	R M H S	-

Outlook: S, O, R

Temp: H, M, C

Humidity: H, N, L

Wind: S, W

9 + 5 -

$H(9/14) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14)$

≈ 0.94

Two Classes: From N_+,N_- to H(P)

Entropy of a *distribution* H(P)

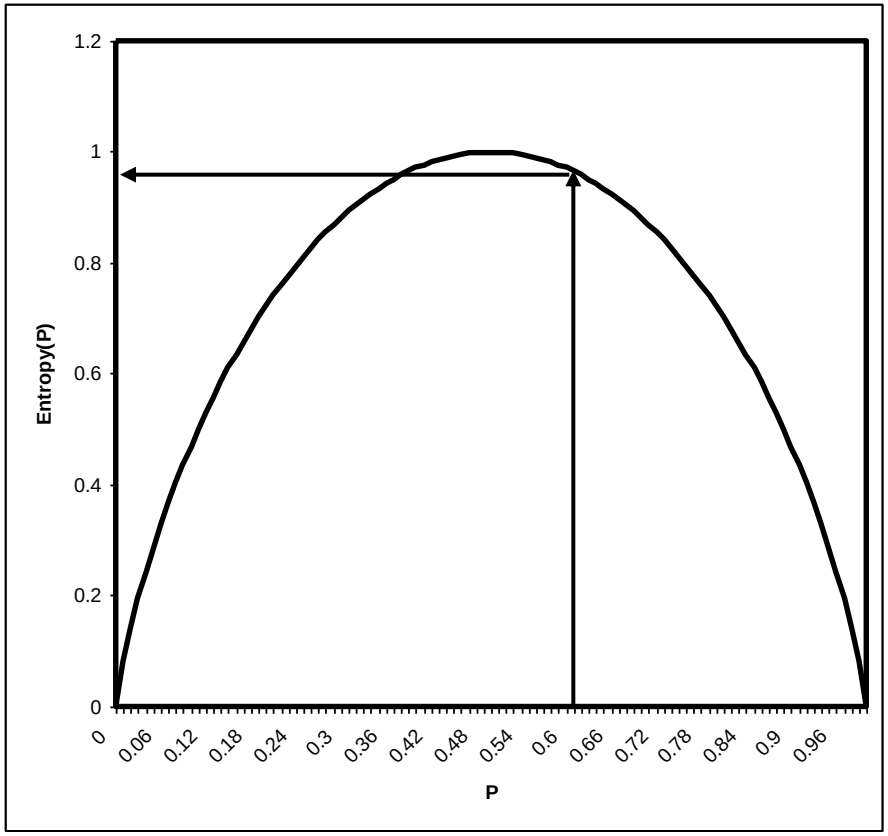
For Binomial:

$P = N_+ / (N_+ + N_-)$

$-P \log_2(P) – (1-P) \log_2(1-P)$

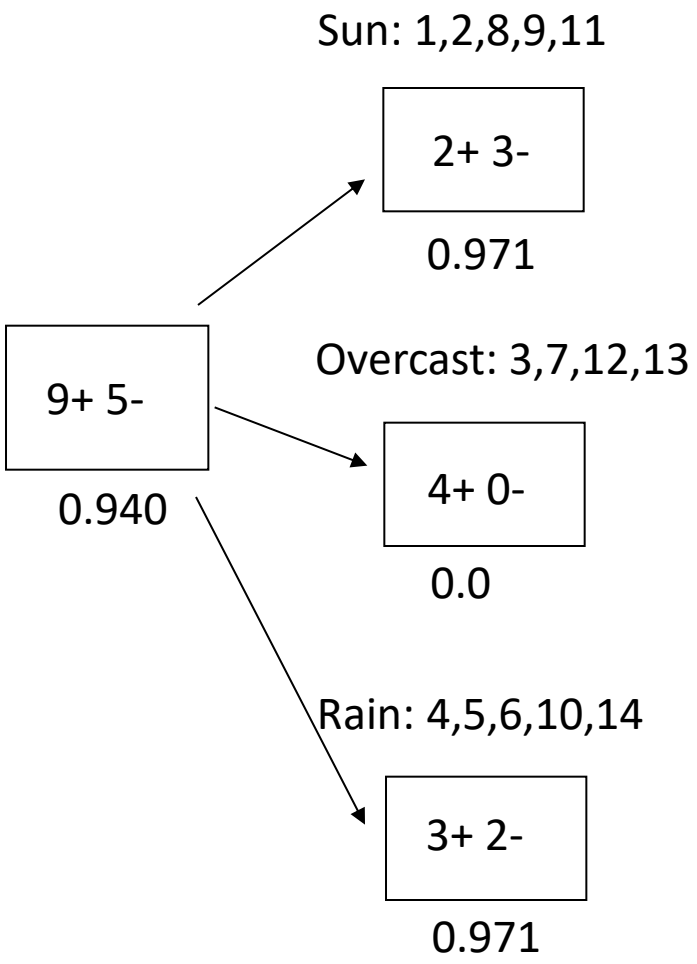
H(9/14) = Entropy(0.64)

0.940



Outlook Gain = 0.246

1.	S	H H W	-
2.	S	H H S	-
3.	O	H H W	+
4.	R	M H W	+
5.	R	C N W	+
6.	R	C N S	-
7.	O	C N S	+
8.	S	M H W	-
9.	S	C N W	+
10.	R	M N W	+
11.	S	M N S	+
12.	O	M H S	+
13.	O	H N W	+
14.	R	M H S	-



Information After:

0.971 * 5/14 +

0.0 * 4/14 +

0.971 * 5/14

= 0.694

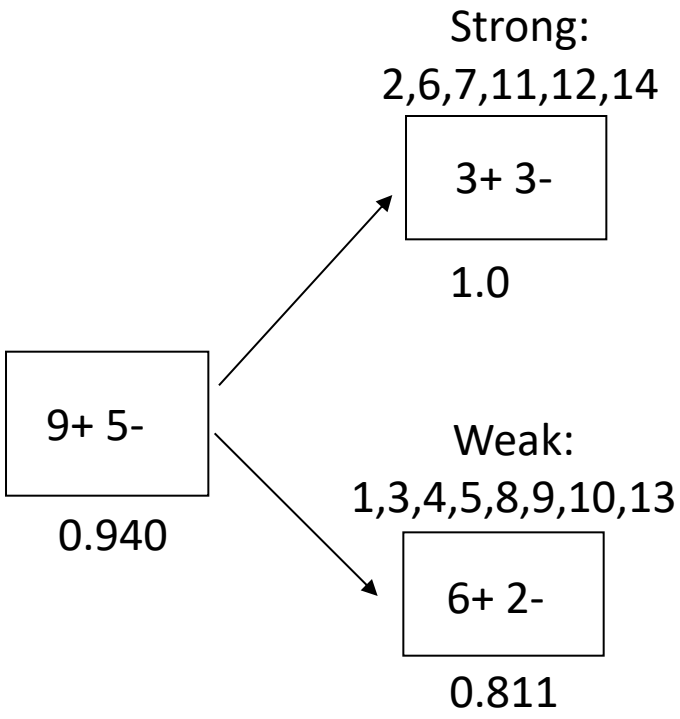
Information Gain:

0.940 – 0.694

=0.246

Wind Gain = 0.048

1.	S H H W	-
2.	S H H S	-
3.	O H H W	+
4.	R M H W	+
5.	R C N W	+
6.	R C N S	-
7.	O C N S	+
8.	S M H W	-
9.	S C N W	+
10.	R M N W	+
11.	S M N S	+
12.	O M H S	+
13.	O H N W	+
14.	R M H S	-



Information After:

1.0 * 6/14 +

0.811 * 8/14

= 0.892

Information Gain:

0.940 – 0.892

=0.048

iClicker!

1.	S H H W	-
2.	S H H S	-
3.	O H H W	+
4.	R M H W	+
5.	R C N W	+
6.	R C N S	-
7.	O C N S	+
8.	S M H W	-
9.	S C N W	+
10.	R M N W	+
11.	S M N S	+
12.	O M H S	+
13.	O H N W	+
14.	R M H S	-

Strong: 2,6,7,11,12,14

3+ 3-

1.0

What is our best estimate of H(Strong)?

A. 1.0
B. Greater than 1.0
C. Less than 1.0
D. None of the above
E. It depends on other things

What is H(Strong)

- We computed it
  - As the value 1.000000
  - P(+ | Strong) = 0.5 Since 3+ and 3-
- This is our *estimate* of the true H
- Is it our *best* estimate?
- Suppose we had 3,000+ and 3,000-
  - Same computation: 1.000000
  - But more confident
- Should confidence change our estimate?

Confidence in H(P)

Entropy H(P) for Binomial:

$P = N_+ / (N_+ + N_-)$

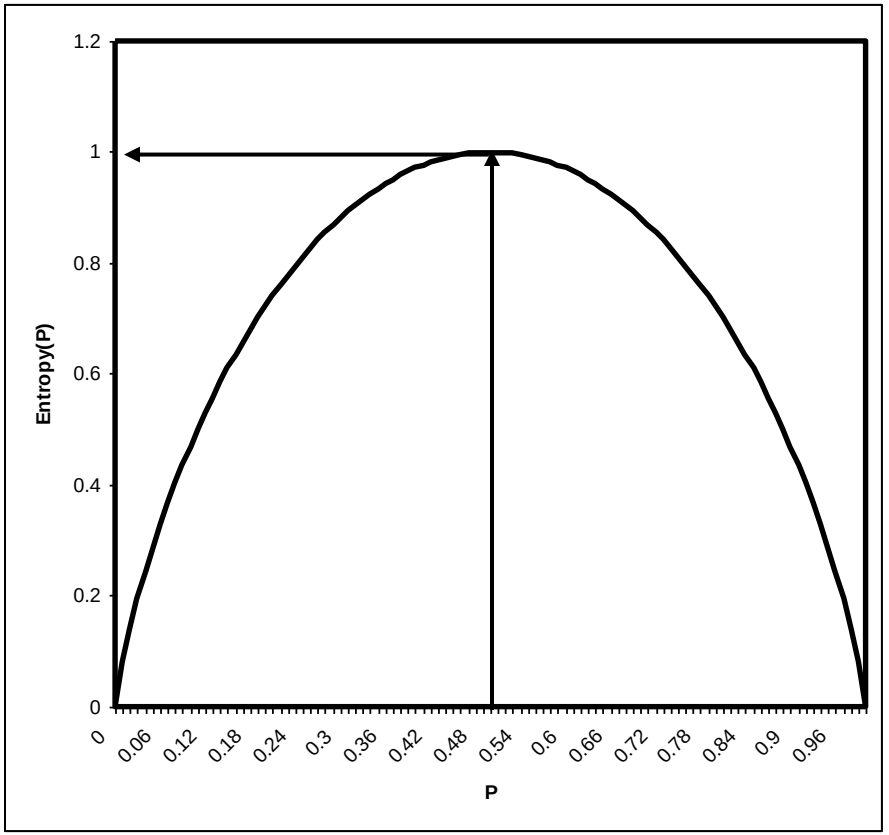
$-P \log_2(P) - (1-P) \log_2(1-P)$

H(3/6) = Entropy(0.5)

= 1.000000

= our *estimate* of H(Strong)

But it is a *biased* estimate



Best guess at P does not yield best guess at H

Binomial / Beta conjugate distributions

Compensate by averaging over our uncertainty

Information Gain

- Outlook 0.25
- Temperature 0.03
- Humidity 0.15
- Wind 0.05

Outlook provides greatest local gain

Split on Outlook

S H H W	-	R C N W	+	S C N W	+	O H N W	+
S H H S	-	R C N S	-	R M N W	+	R M H S	-
O H H W	+	O C N S	+	S M N S	+		
R M H W	+	S M H W	-	O M H S	+		

Sunny

S H H W	-
S H H S	-
S M H W	-
S C N W	+
S M N S	+

Overcast

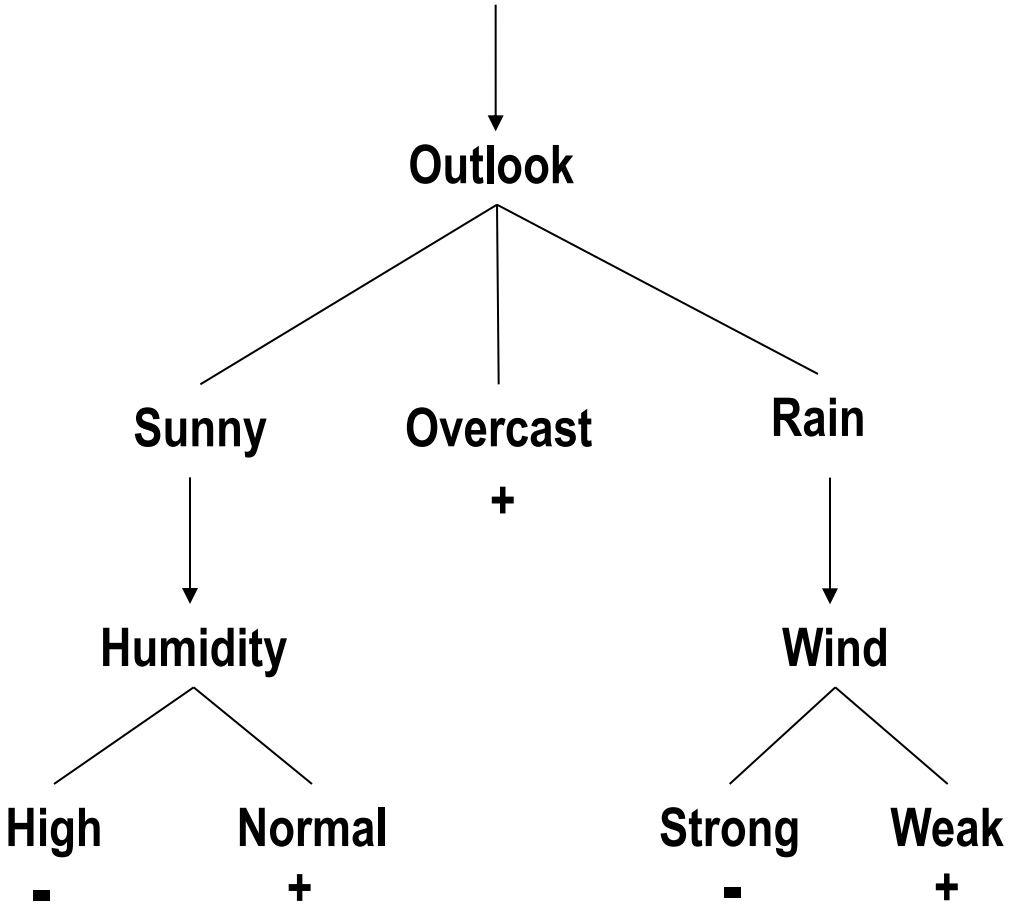
O H H W	+
O C N S	+
O M H S	+
O H N W	+

Rain

R M H W	+
R C N W	+
R C N S	-
R M N W	+
R M H S	-

Now recur on each smaller set

Final Decision Tree



Suppose under Sunny we split on Outlook (again) instead of Humidity?

What can we say about entropy as we measure additional features?