CSC 480/580 Principles of Machine Learning

01 Supervised learning; Decision Trees

Jason Pacheco

Department of Computer Science

THE UNIVERSITY OF ARIZONA

Administrivia

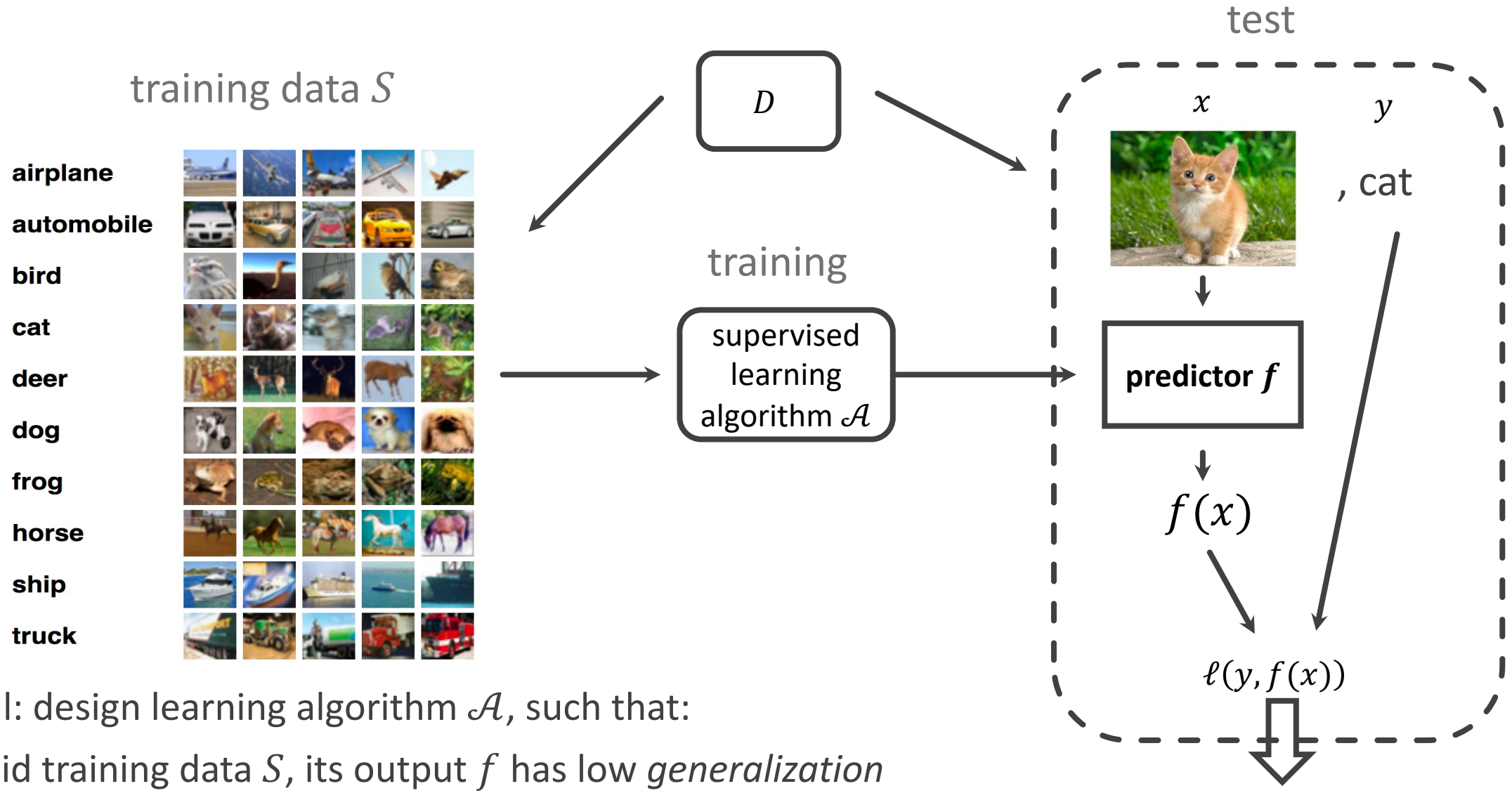
Office hours:

- Jason : Mon/Wed 3-4pm : In-person (GS Room 707)
- Emiliano : Tue/Thurs 2-3pm : In-person (GS Room 934)
- Mahshad : Fri 1-2pm : Zoom (Link on Piazza)

All details posted on Piazza...

The supervised learning problem

Supervised learning setup: putting it together



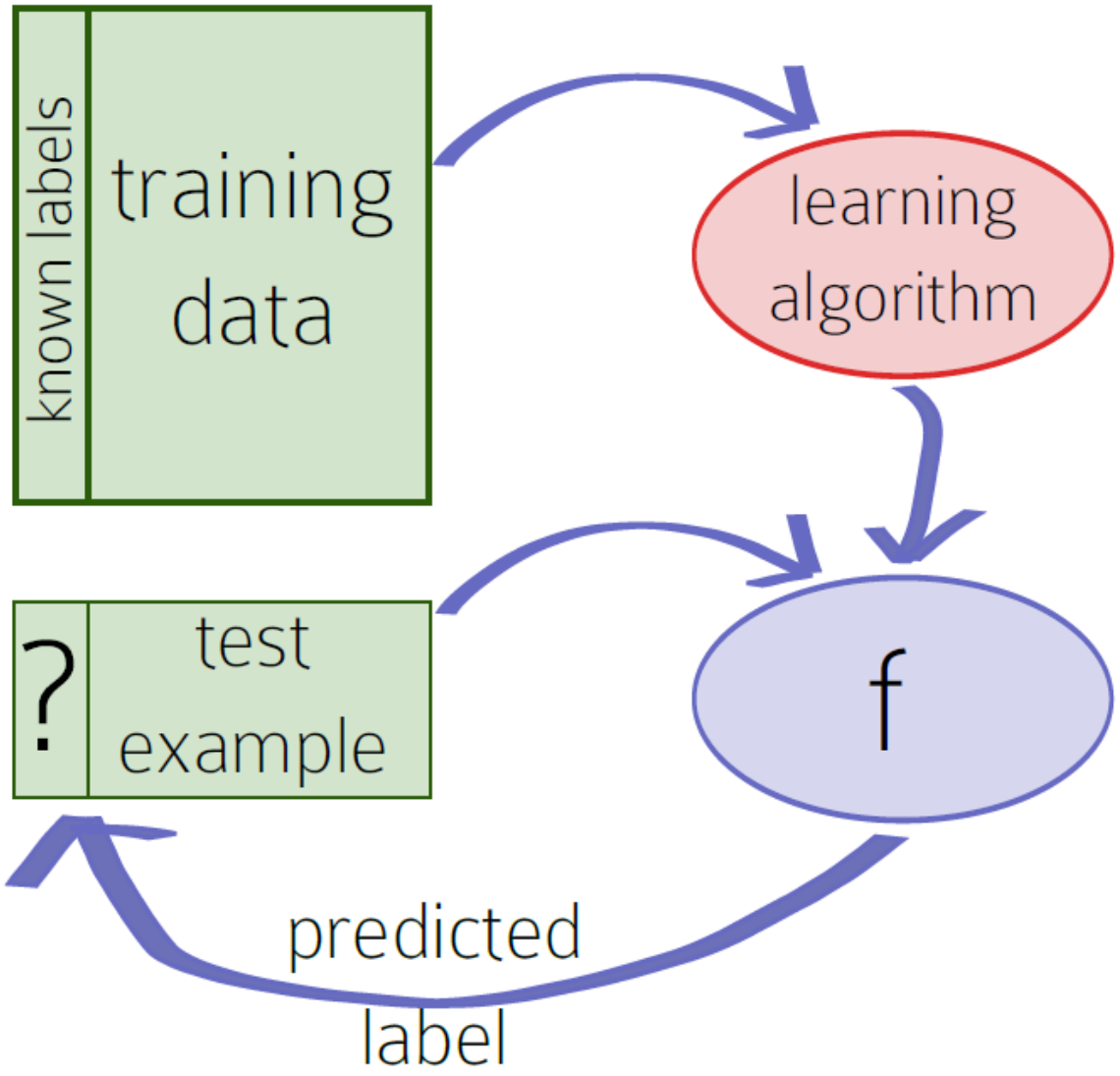
- Goal: design learning algorithm $\mathcal{A}$, such that:

  on iid training data S, its output f has low *generalization error* $L_D(f)$

Generalization error: $L_D(f) = \mathrm{E}_{(x,y)\sim D}\, \ell(y, f(x))$

First supervised learning algorithm: learning decision trees

Supervised Learning



Goal Learn function *f* from training data that makes predictions on unseen test data

Prototypical supervised learning problems:

- Regression
- Classification (binary / multiclass)
- Ranking
- …

This lecture will focus on binary classification…

Example: course recommendation

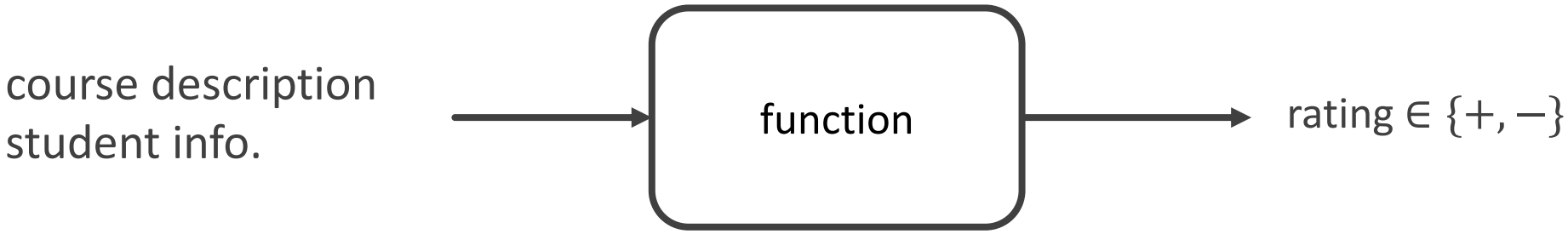
Task *Given a student, recommend a set of courses that s/he would like*

We are allowed to ask a sequence of yes / no questions…

You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: *I predict this student will not like this course.*

The Machine Learning approach:

learned from previous students' course survey results



Model: Decision Tree

Use our questions to build a binary tree:

You: Is the course under consideration in Systems?
Me: Yes
You: Has this student taken any other Systems courses?
Me: Yes
You: Has this student liked most previous Systems courses?
Me: No
You: *I predict this student will not like this course.*

Terminology:

- Question & Answer → Feature
- Set of Question & Answers → Training Data
- "Like" / "Not-like" → Labels

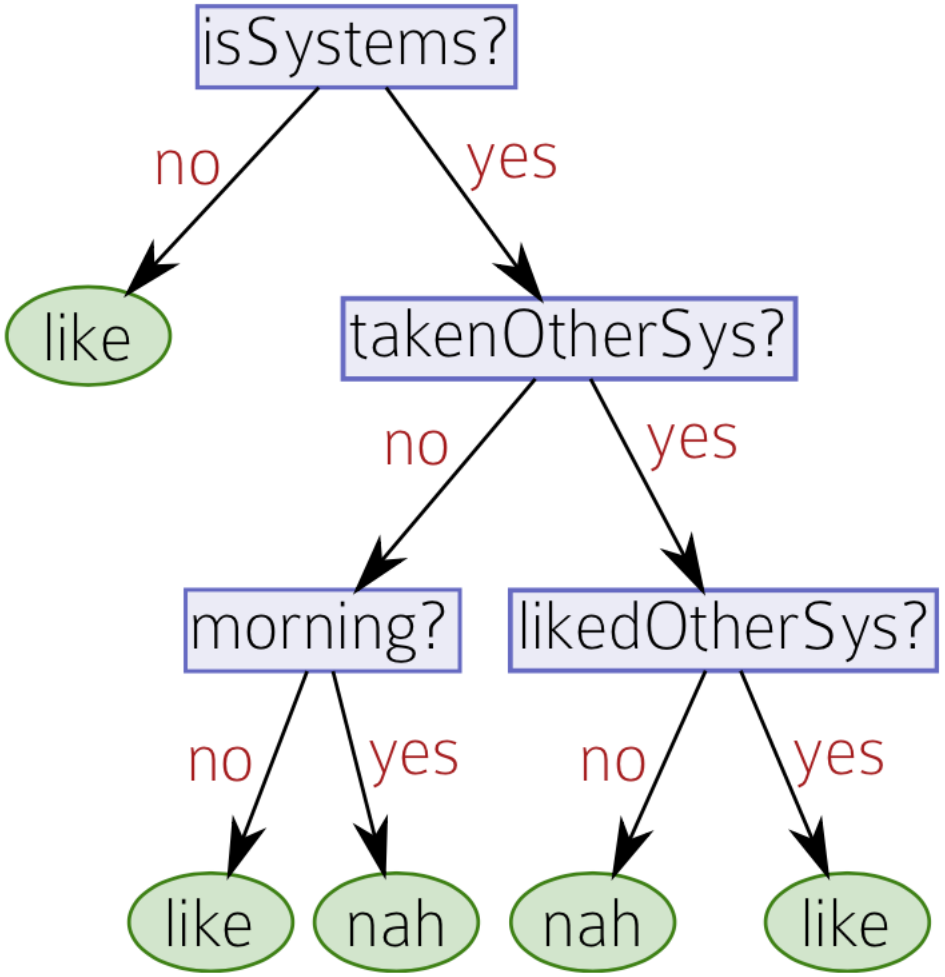


Figure 1.2: A decision tree for a course recommender system, from which the in-text "dialog" is drawn.

Training Dataset: previous course survey results

Define the labeled training dataset $S = \{(x_i, y_i)\}_{i=1}^{m}$

To make this a binary classification we set
"Like" = {+2,+1,0}
"Not-like" = {-1,-2}

Features

Feature Values

Labels

Data Point

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Decision trees: basic terminology

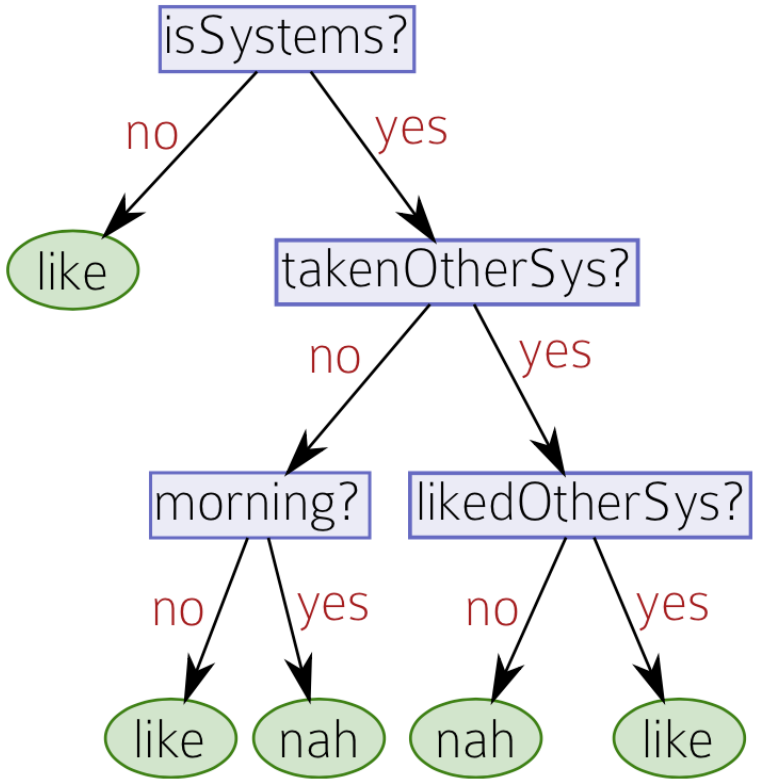


Figure 1.2: A decision tree for a course recommender system, from which the in-text "dialog" is drawn.

node
root node
leaf node
internal node

parent
children
subtree
depth

- Key advantage of using decision trees for decision making: *intepretability*
- Useful in consequential settings, e.g. medical treatment, loan approval, etc.

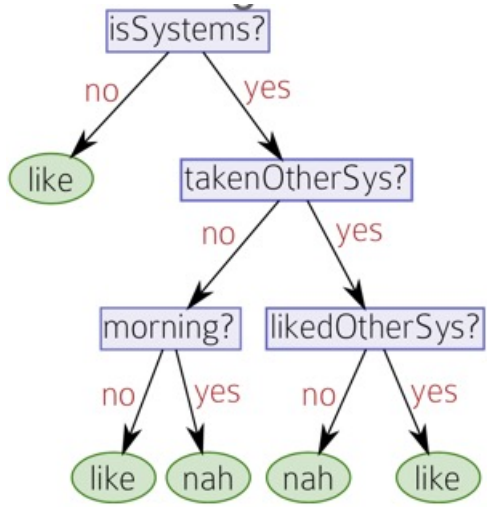
nodes organized in a tree-based structure, leading to a prediction (Fig. 1). The interpretability of decision trees allows physicians to understand why a prediction or stratification is being made, providing an account of the reasons behind the decision to subsequently accept or override the model's output. This interaction between humans and algorithms can provide

Prediction using decision trees

- Test: predict using a decision tree:

Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
1: if tree is of the form Leaf(guess) then
2:     return guess
3: else if tree is of the form Node(f, left, right) then
4:     if f = no in test point then
5:         return DecisionTreeTest(left, test point)
6:     else
7:         return DecisionTreeTest(right, test point)
8:     end if
9: end if
```



guess=prediction

left=no
right=yes

- Training: apply a learning algorithm that can build trees f from training data
- How to design the algorithm?

Learning Decision Trees

Example Guess a number between 1 and 100. Which set of questions are better? Why?

Set 1	**Set 2**
1) Greater than 20? **Y**	1) Greater than 50? **Y**
2) Has a 7 in it? **N**	2) Greater than 75? **N**
3) Odd? **N**	3) Greater than 63? **N**

How many questions should this problem require?

Key Intuition: The decision tree should try to ask informative questions

Training accuracy / error

- The training data $S = \{(x_i, y_i)\}_{i=1}^{m}$

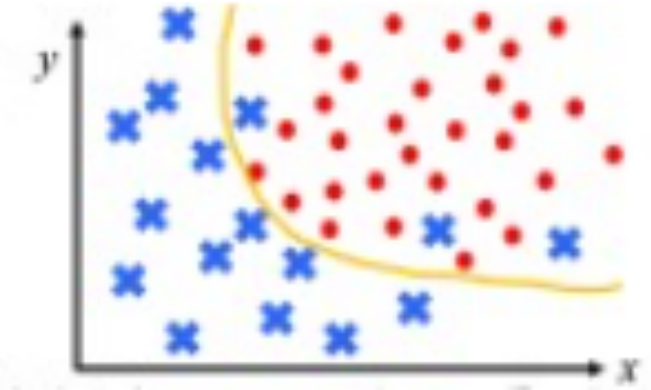
- Predictor f's training accuracy = fraction of examples in S that are **correctly classified** by f
- In formula,

$$A_S(f) = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) = y_i)$$

$I(A) = 1$ if A happens; =0 otherwise

- Training error $L_S(f) = 1 - A_S(f) = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) \neq y_i)$

training accuracy = 47 / 50



- The training error minimization approach:
  - Idea: f has low $L_S(f) \Rightarrow f$ likely has low $L_D(f)$
  - Approach: Find f with lowest $L_S(f) \Leftrightarrow f$ with highest $A_S(f)$
  - Problem with this approach?

May overfit – we'll come back to this

Decision tree training: zero-level case

- Q: if I would like to find a depth-0 tree with the highest training accuracy, what would that tree look like?

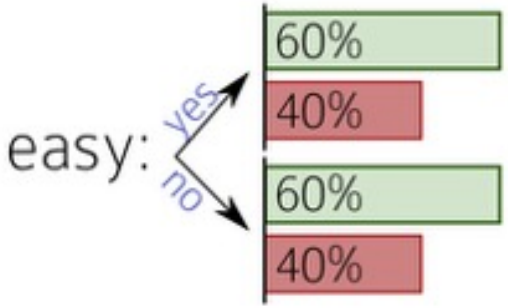
- The tree will not have internal nodes (no questions asked), and just has a leaf

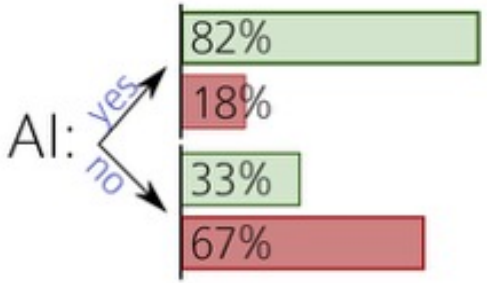
- Choosing leaf to be ”like” – training accuracy = 60%
- Choosing leaf to be ”not-like” – training accuracy = 40%
- The optimal leaf would be ”like”

Decision tree training: single level case

- Q: if I could only ask one question (design a depth-1 tree), what question would I ask?
- Perhaps useful: look at the *histograms* of labels (for like & not-like), grouped by values taken by the feature

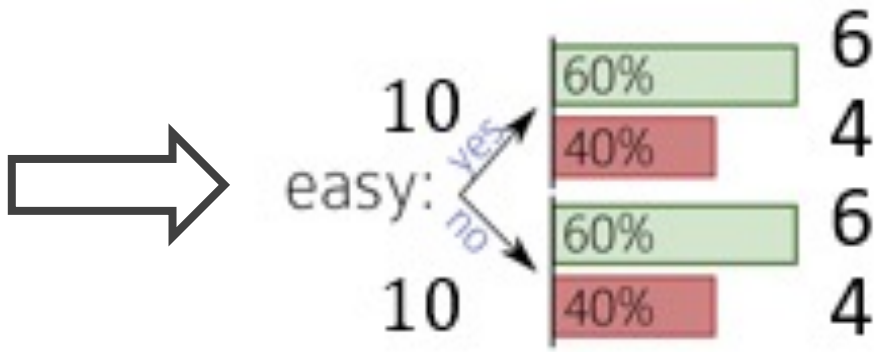
- Example: feature “easy” vs. feature “AI”





Histogram calculation

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y



Decision tree training: single level case

- Q: if I could only ask one question (design a depth-1 tree), what question would I ask?

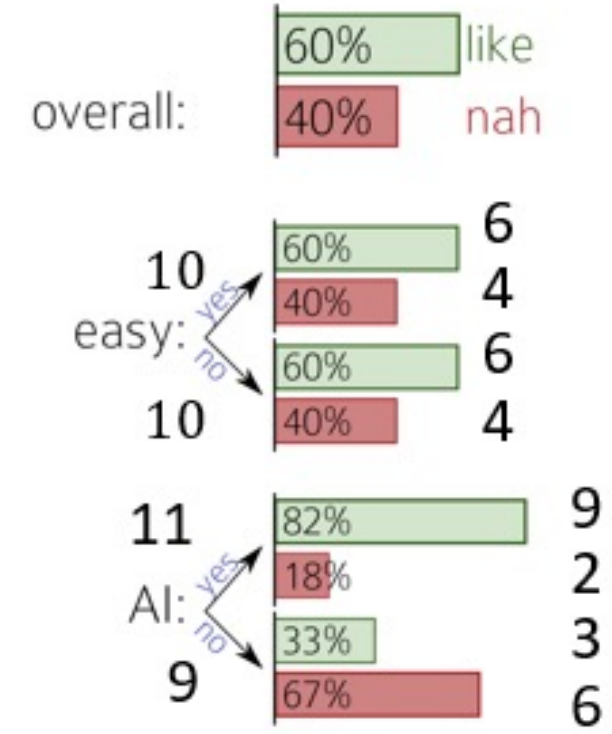
- Intuition: look at the histograms of labels for each feature
- Which feature (question) is better, ‘easy’ or ‘AI’?

- Best training accuracy using ‘easy’:

  (max(6,4) + max(6,4)) / 20 = 0.6

- Best training accuracy using ‘AI’:

  (max(9,2) + max(3,6)) / 20 = 0.75

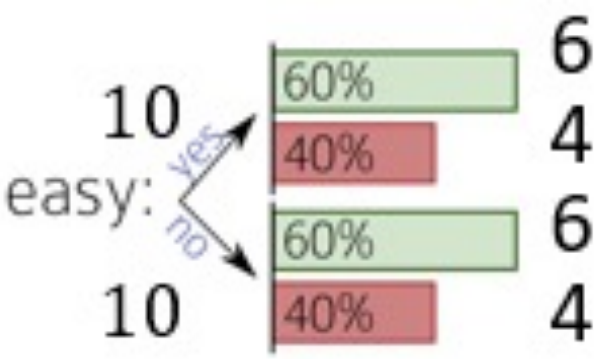


- Thus, ‘AI’ is more informative (more useful in telling apart like & not-like)

Decision tree training: single level case

- In formula:
- $\text{Score}(f, S) := \max\left(P_S(y = +, x_f = yes), P_S(y = -, x_f = yes)\right)$

$$+\max\left(P_S(y = +, x_f = no), P_S(y = -, x_f = no)\right)$$



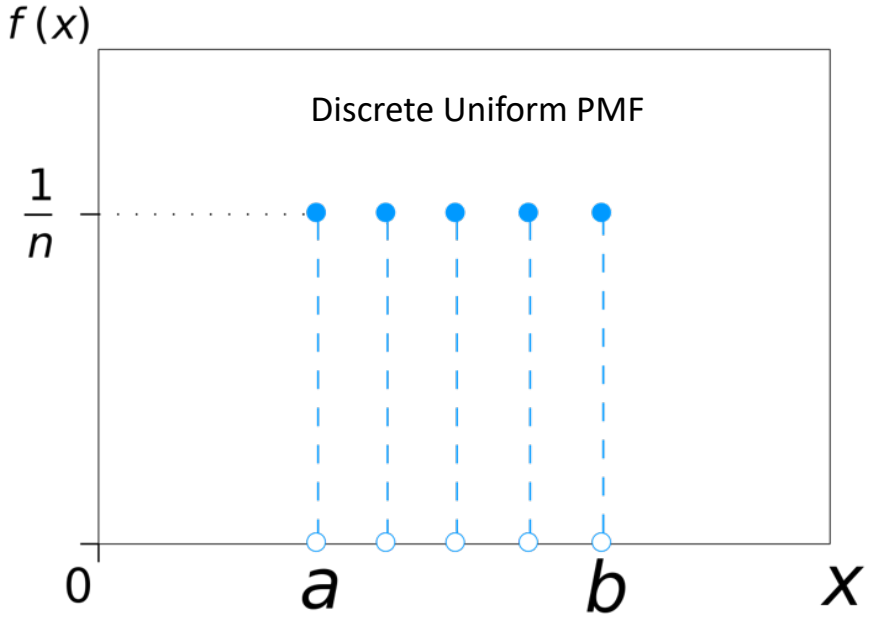
- Here, $P_S(E)$ denotes the probability of event E if an example (x, y) is chosen uniformly from S
- In other words: the frequency of E in sample S
- E.g.

$$\text{Score}('\text{easy}', S) = \max\left(\frac{6}{20}, \frac{4}{20}\right) + \max\left(\frac{6}{20}, \frac{4}{20}\right) = 0.6$$

Math interlude: Uniform distribution over a set

Given set $S = \{v_1, v_2, \ldots, v_N\}$; Z is drawn from the uniform distribution of S, then

$$P(Z = k) = \begin{cases} \frac{1}{N}, & k \in \{v_1, v_2, \ldots, v_N\} \\ 0, & \text{otherwise} \end{cases}$$



We write this as $Z \sim \text{Uniform}(S)$

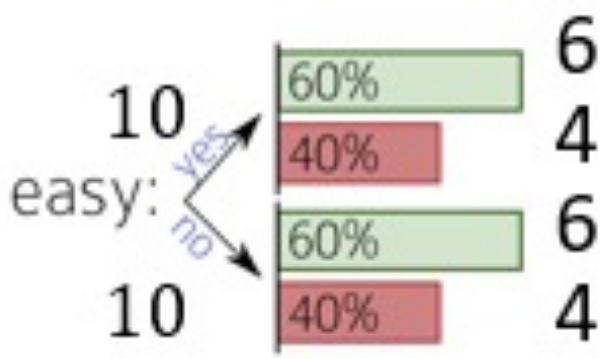
- Selecting a student from a class
- Choosing an example from a dataset

- $P_{Z \sim \text{Uniform}(S)}$ is often abbreviated as P_S
- $P_S(E)$ = fractions of elements in S that satisfies E

Decision tree training: single level case

- In formula:
- $\text{Score}(f, S) := \max\left(P_S(y = +, x_f = yes), P_S(y = -, x_f = yes)\right)$

$$+\max\left(P_S(y = +, x_f = no), P_S(y = -, x_f = no)\right)$$



- Written in conditional probability form: A measure of label "purity"

$P(AB) = P(A)P(B \mid A)$

$$= \max\left(P_S(y = +|\, x_f = yes), P_S(y = -|\, x_f = yes)\right) \cdot P_S(x_f = yes)$$

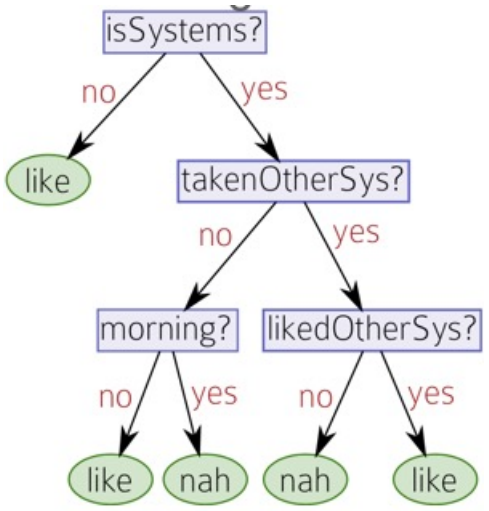
$$+ \max\left(P_S(y = +|\, x_f = no), P_S(y = -|\, x_f = no)\right) \cdot P_S(x_f = no)$$

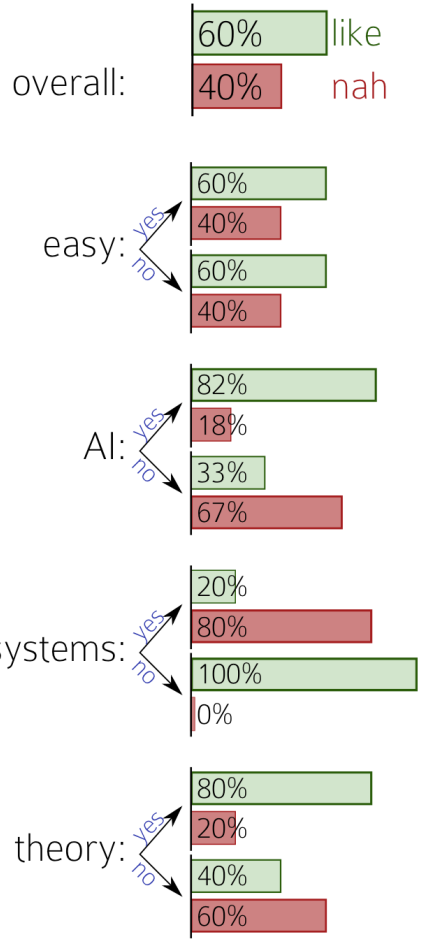
- e.g. $\text{Score}('\text{easy}', S) = \max(0.6, 0.4) \times 0.5 + \max(0.6, 0.4) \times 0.5 = 0.6$
- Interpretation: $\text{Score}(f, S)$ measures the ability of feature f in predicting label y – **informativeness** of feature f

Decision tree training: general level case

- Key idea: divide & conquer

- Build the root of the tree greedily
- Build the left and right subtrees *recursively*
- When to stop the recursion?



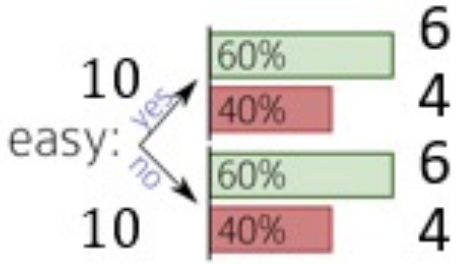


Algorithm 1 DECISIONTREETRAIN(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return LEAF(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return LEAF(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DECISIONTREETRAIN(NO, remaining features \ {f})
    right ← DECISIONTREETRAIN(YES, remaining features \ {f})
    return NODE(f, left, right)
end if
```

answer=label

Unambiguous = all identical



Score(f, S) = informativeness of feature f for dataset S

Q: does this algorithm always terminate?

A: Yes, since each recursive call decreases the number of available features by 1

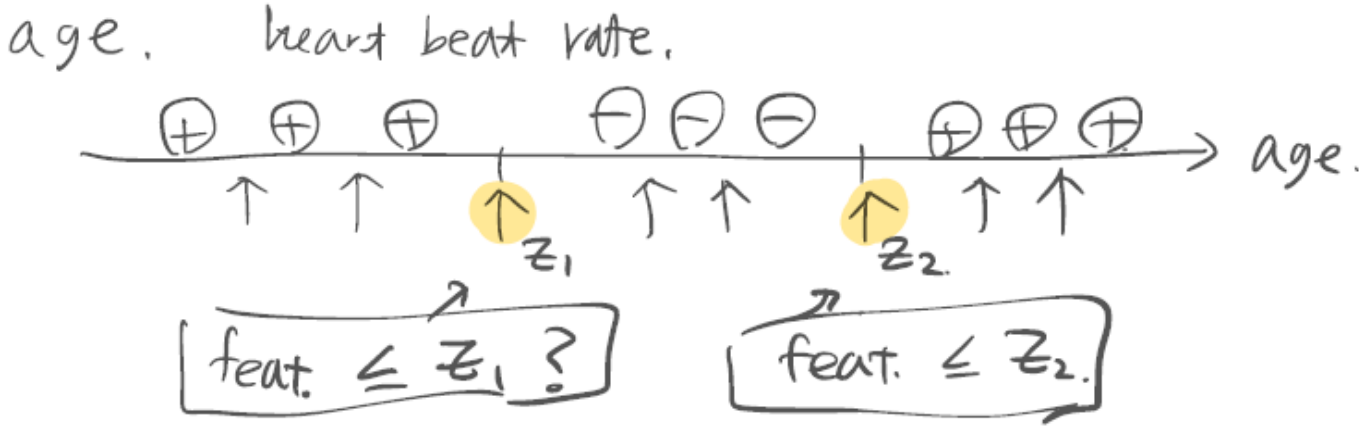
Dealing with various types of features

- Binary: $x_f \in \{0,1\}$
  - Node: $x_f = 0$?

Easy?	AI?	Sys?	Thy?	Morning?
y	y	n	y	n

- Categorical: $x_f \in \{1,2,\dots,C\}$
  - Node: $x_f \in \{i_1,\dots,i_l\}$?

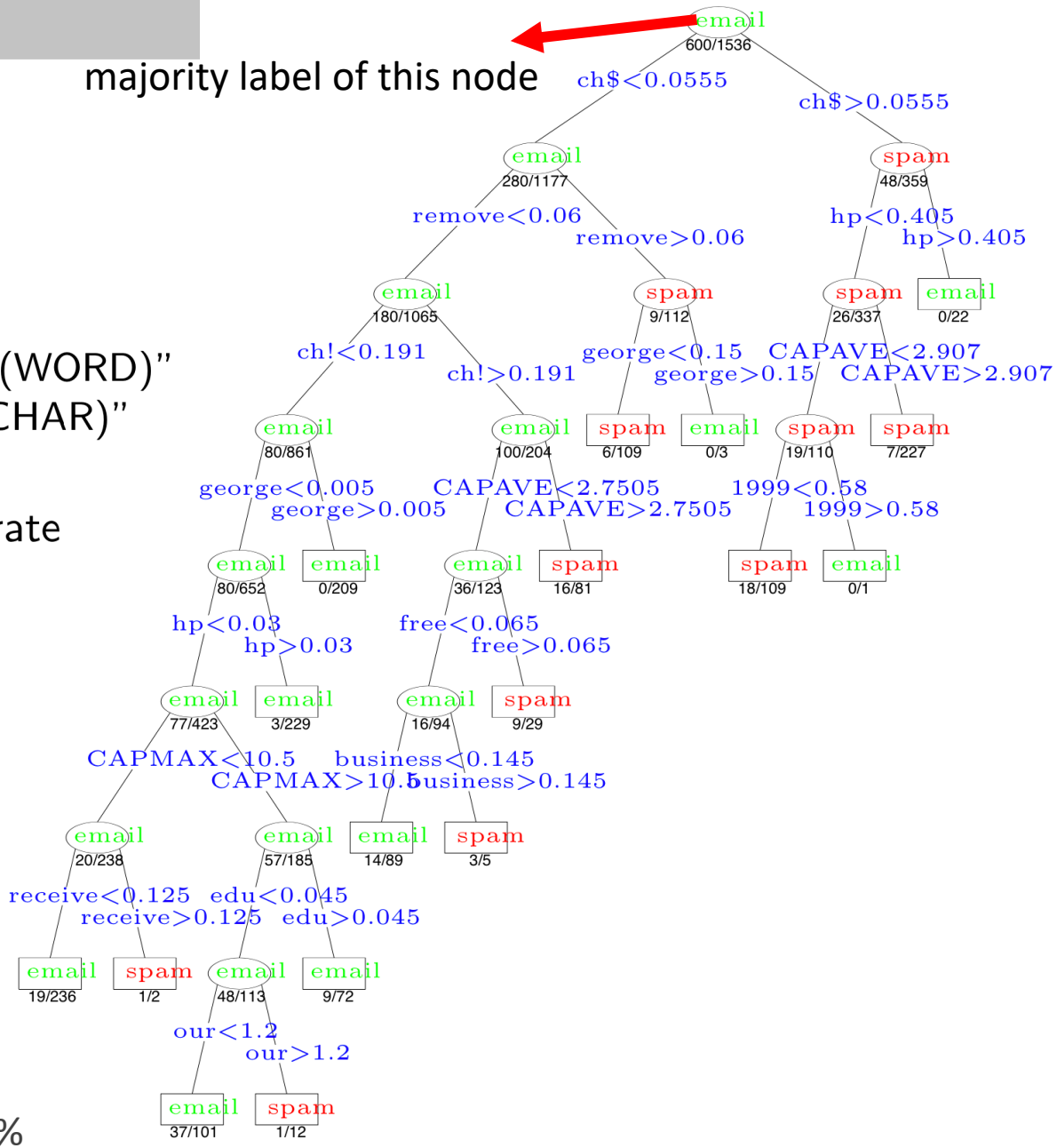
- real value: $x_f \in \mathbb{R}$
  - Node: $x_f \leq z$?



https://medium.com/@data.science.enthusiast/ways-to-handle-categorical-data-in-python-e89d25c40338

Example: spam filtering I

- Spam dataset
- 4601 email messages, about 39% are spam
- Classify message by spam and not-spam
- 57 features
  - 48 are of the form "percentage of email words that is (WORD)"
  - 6 are of the form "percentage of email characters is (CHAR)"
  - 3 other features (e.g., "longest sequence of all-caps")
- Final tree after pruning has 17 leaves, 9.3% test error rate



The meaning of numbers: e.g., 600 / 1536

test examples passing this node with minority label

test examples passing through this node

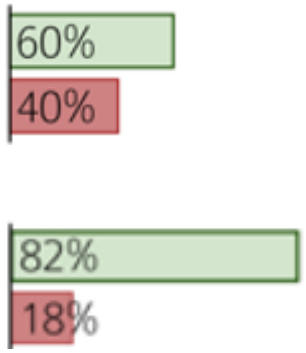
Q: What is the best depth-0 decision tree, and what is its accuracy?

A: predicting "email" for all examples, accuracy = 736 / 1536 = 47.9%

Decision tree training: generalized informativeness scores

- $\text{Score}(f)$ = a measure of **informativeness** of f
- Alternative view -- **uncertainty reduction**: how much uncertainty about y can be reduced if we know f
- Uncertainty measures of population:

We used classif. accuracy: max(p, 1-p),



Notions of uncertainty: binary case ($\mathcal{Y} = \{0, 1\}$)

Suppose in a set of examples $S \subseteq \mathcal{X} \times \{0, 1\}$, a p fraction are labeled as 1

1. **Classification error**:

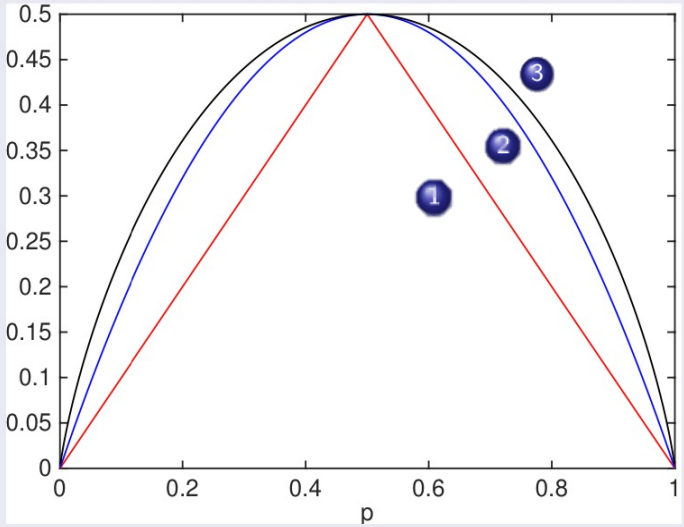
$$u(S) := \min\{p, 1 - p\}$$

2. **Gini index**:

$$u(S) := 2p(1 - p)$$

3. **Entropy**:

$$u(S) := p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

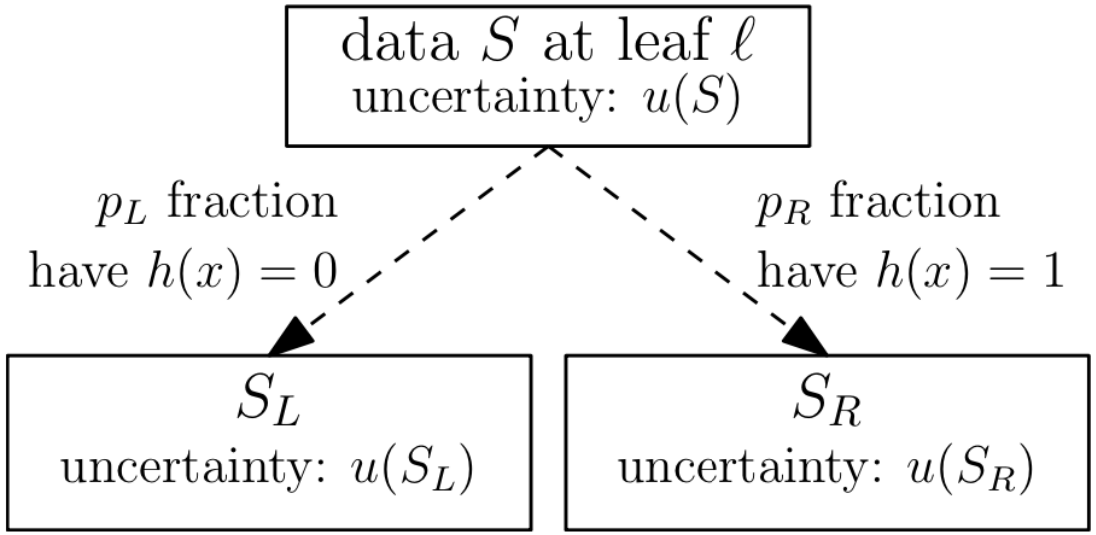


(3) is divided by 2 so the plot looks comparable

log here is base-2

Decision tree training: generalized informativeness scores

Suppose the data S at a leaf ℓ is split by a rule h into S_L and S_R, where $p_L := |S_L|/|S|$ and $p_R := |S_R|/|S|$



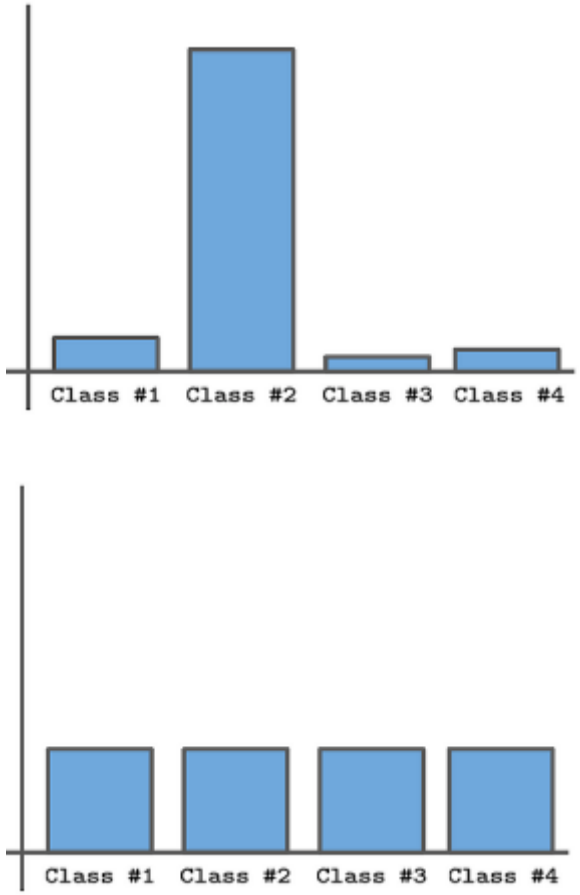
The **reduction in uncertainty** from using rule h at leaf ℓ is

$$u(S) - \Big(p_L \cdot u(S_L) + p_R \cdot u(S_R)\Big)$$

=: Score(h, S) (Generalized)

Decision tree training: generalized informativeness scores

- Multiclass classification setting: $\mathcal{Y} = \{1, \ldots, K\}$



Notions of uncertainty: general case

Suppose in $S \subseteq \mathcal{X} \times \mathcal{Y}$, a p_k fraction are labeled as k (for each $k \in \mathcal{Y}$).

1. **Classification error**:
$$u(S) := 1 - \max_{k \in \mathcal{Y}} p_k$$
2. **Gini index**:
$$u(S) := 1 - \sum_{k \in \mathcal{Y}} p_k^2$$
3. **Entropy**:
$$u(S) := \sum_{k \in \mathcal{Y}} p_k \log \frac{1}{p_k}$$

Each is *maximized* when $p_k = 1/|\mathcal{Y}|$ for all $k \in \mathcal{Y}$
(i.e., equal numbers of each label in S)

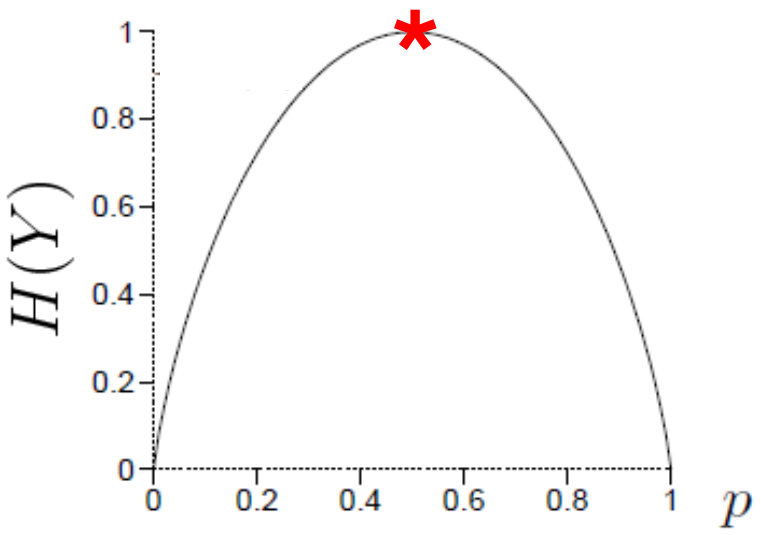
Each is *minimized* when $p_k = 1$ for a single label $k \in \mathcal{Y}$
(so S is **pure** in label)

https://towardsdatascience.com/confidence-calibration-for-deep-networks-why-and-how-e2cd4fe4a086

Entropy Uncertainty

Entropy of random variable Y: $H(Y) = \sum_y P(Y = y) \log \frac{1}{P(Y=y)}$

Coin Flip Example: $Y \sim \text{Bernoulli}(p)$



- Key object studied in **information & coding theory**
- Interpretations:
  - the minimum number of bits needed to reliably encode Y
  - the uncertainty about outcome of Y

Maximum uncertainty when coin is fair.

Information Theory

- Shannon's seminal paper, "A Mathematical Theory of Communication", Bell Labs Technical Journal, 1948

- Claude Shannon (1916-2001): pioneered AI research

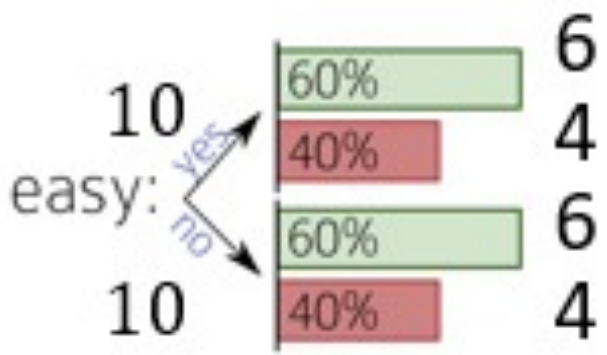
corridors until it found the target.[65] Having travelled through the maze, the mouse could then be placed anywhere it had been before, and because of its prior experience it could go directly to the target. If placed in unfamiliar territory, it was programmed to search until it reached a known location and then it would proceed to the target, adding the new knowledge to its memory and learning new behavior.[65] After much trial and error, this device would learn the shortest

- "Prediction and Entropy of Printed English" – early work on the n-gram model in NLP

Generalized informativeness score in action

- Example: consider **entropy uncertainty**, on dataset S

  $u(S) = \sum_{l\in\{+,-\}} P_S(y = l) \log \frac{1}{P_S(y=l)}$



- $\text{Score}(S, f) = u(S) - \left(p_L\, u(S_L) + p_R\, u(S_R)\right)$

$P_S(x_f = no)$

$$\sum_{l\in\{+,-\}} P_S(y = l \mid x_f = no) \log \frac{1}{P_S(y = l \mid x_f = no)}$$

- In the above example:
  - $u(S) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$,
  - $p_L = 0.5, p_R = 0.5$
  - $u(S_L) = u(S_R) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$

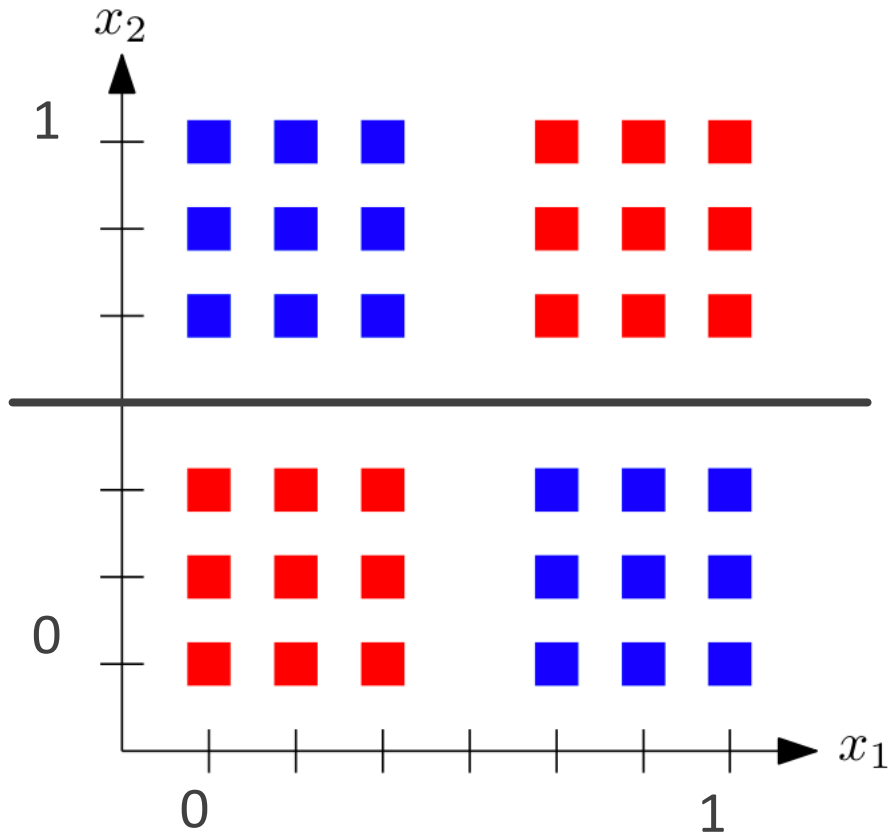
  $\Rightarrow \text{Score}(' \text{ easy}', S) = 0$
- In this case, $\text{Score}(S, f)$ is also known as the **mutual information** between x_f and y (under P_S)

Stop splitting when there is no reduction in uncertainty? This is a bad idea!

The 'XOR' data:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

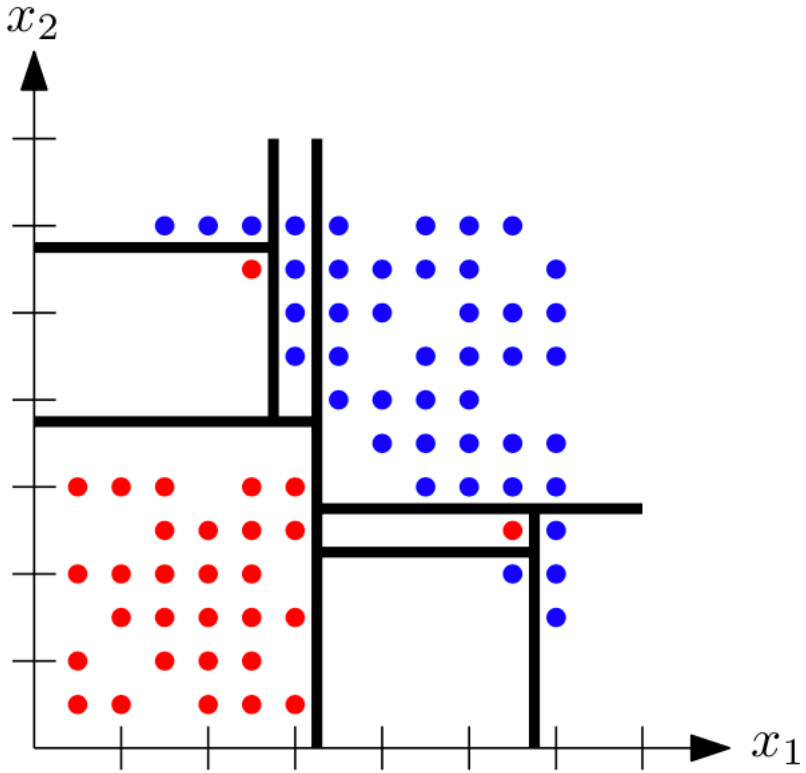
Suppose $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{Y} = \{\text{red}, \text{blue}\}$, and the data is as follows:



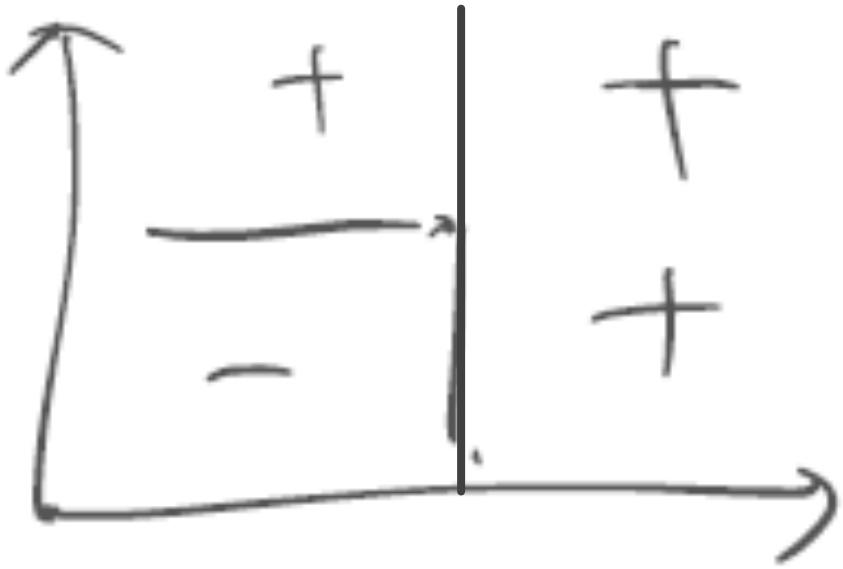
- Any axis-aligned split has no reduction on uncertainty on S
- However, a depth-2 decision tree (with axis-aligned splits) has zero training error

Overfitting can happen

- "Spurious" patterns can be learned.



A better alternative:



- A fix to the training algorithm: stop recursion & return leaf once we reach depth k (say $k = 2$)
- Alternatively: do **pruning** on trained decision tree – a popular heuristic

Next lecture

- Supervised learning: what to do if the data distribution is *known*?
- Models, parameters, hyperparameters
- Practical considerations
- Assigned reading: CIML Chap. 2 (Limits of learning)