

CSC 580 Principles of Machine Learning

# 01 Supervised learning; Decision Trees

**Chicheng Zhang**

**Department of Computer Science**

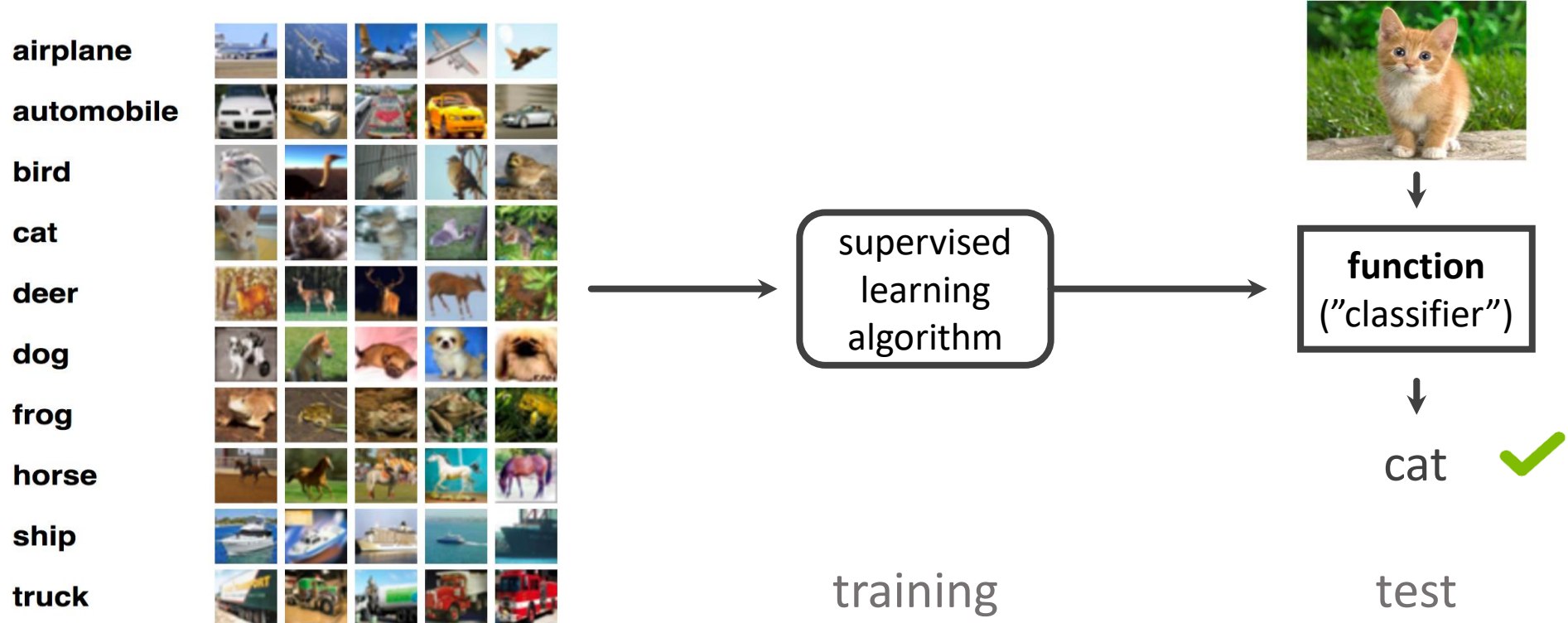


\*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

# The supervised learning problem

# Recap: Supervised learning

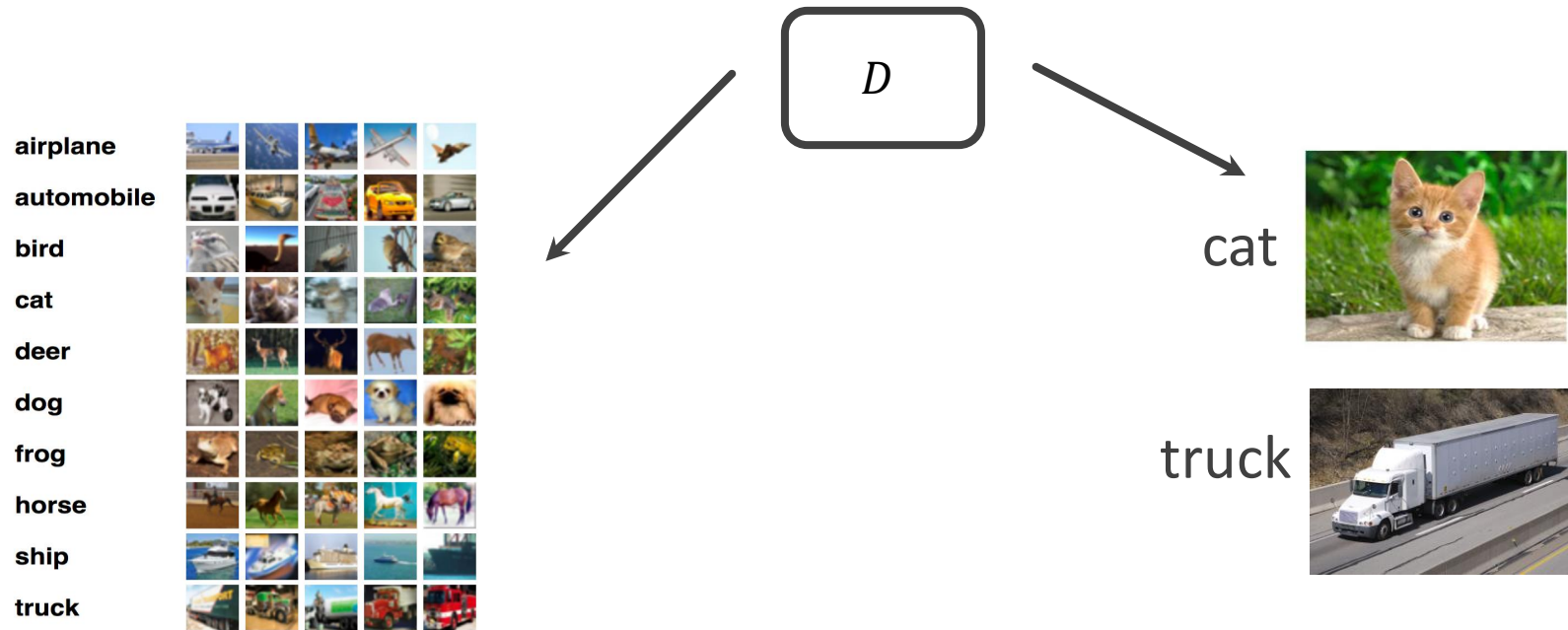
- Training / test data: datasets comprised of labeled examples: pairs of (feature, label)



- Question: what makes a test procedure “reasonable”?
  - Test data: should it come from some other population? Should it overlap with the training data?
  - Compare predicted labels with true labels: how?

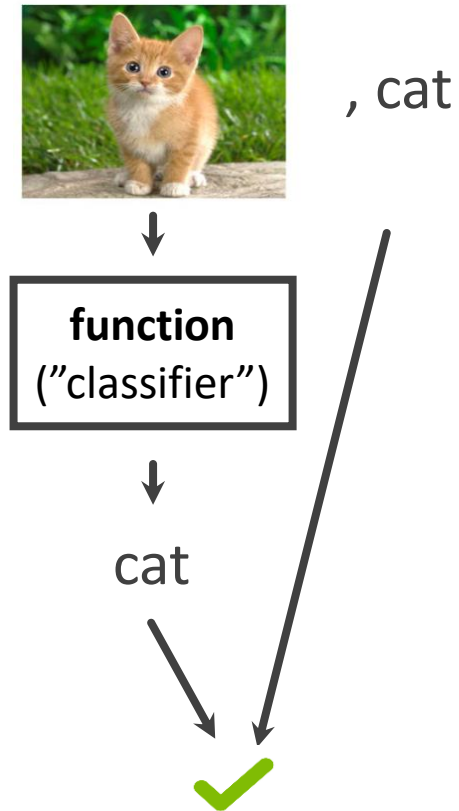
# Supervised learning: formal setup

- Training and test data are drawn independently from the same *data generating distribution*  $D$ 
  - IID: independent and identically distributed
- Training and test data are independent

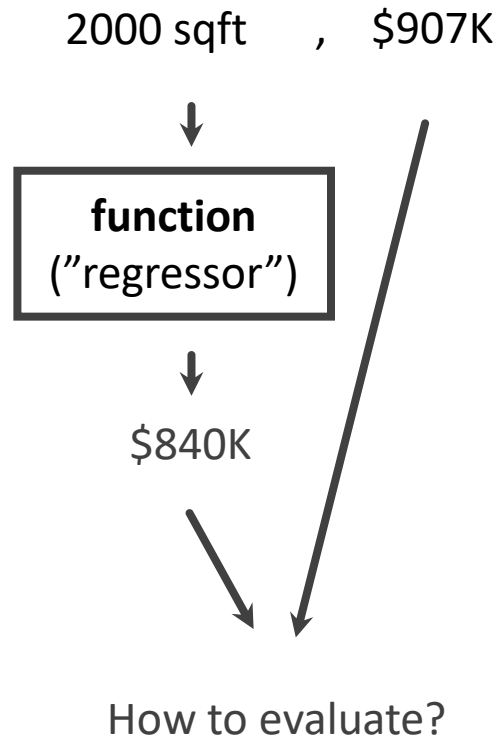


# Supervised learning: formal setup (cont'd)

- Scenario 1: classification



- Scenario 2: regression



- Loss function  $\ell$ : measuring the prediction quality with respect to ground truth label

- Examples:

- Zero-one loss

$$\ell(y, \hat{y}) = I(y \neq \hat{y}) - \text{classification}$$

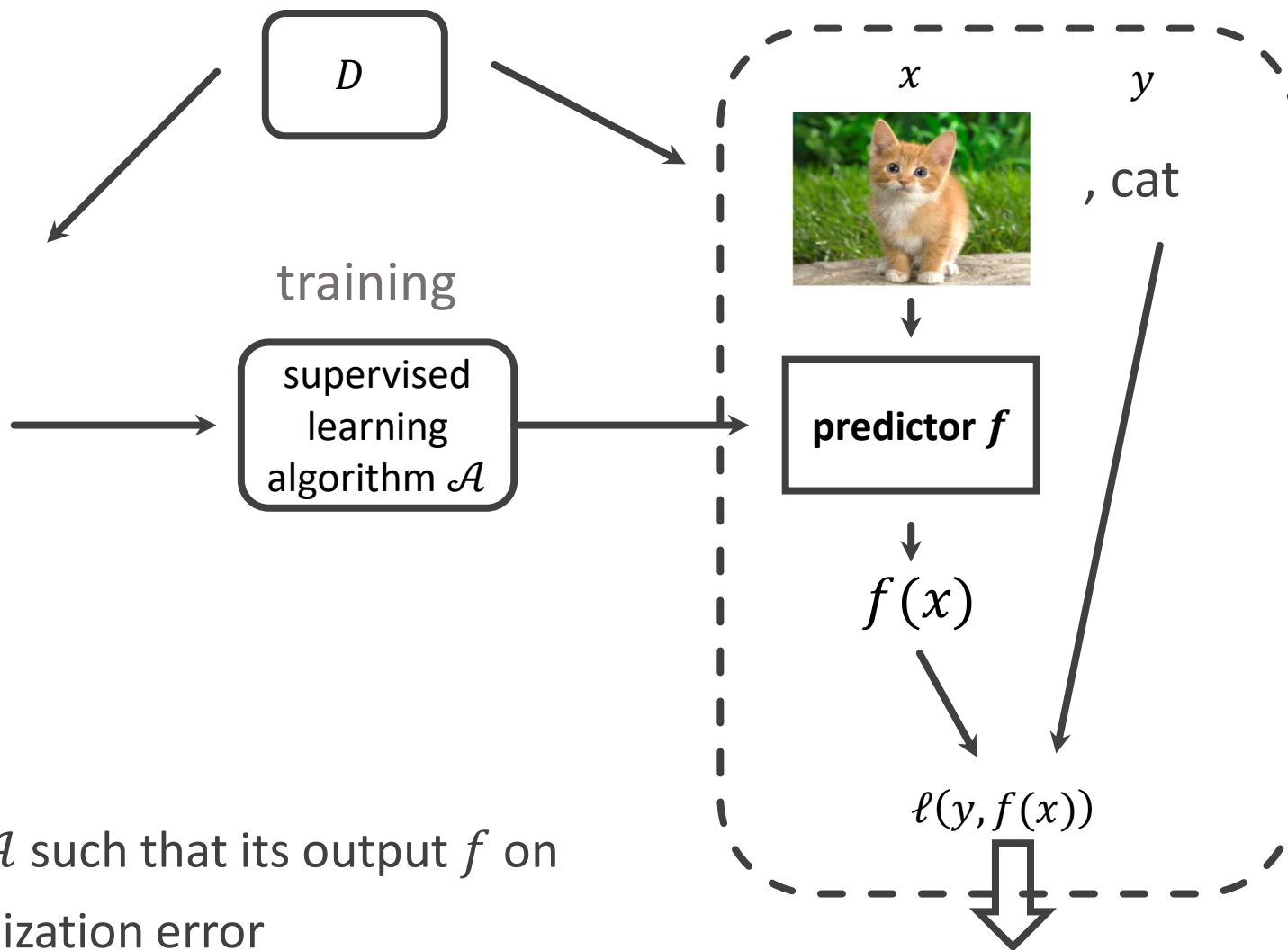
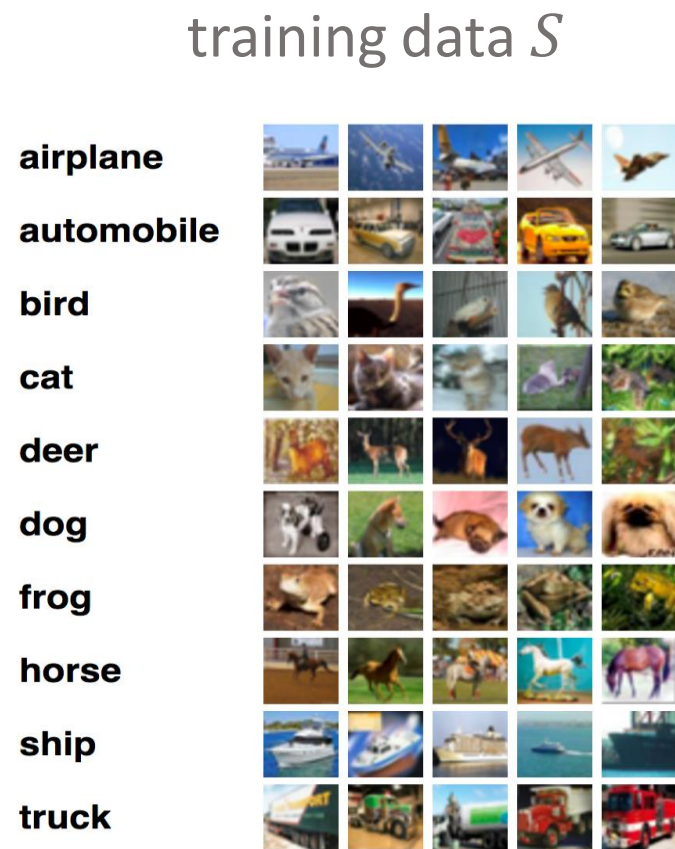
- Square loss

$$\ell(y, \hat{y}) = (y - \hat{y})^2 - \text{regression}$$

- Absolute loss:

$$\ell(y, \hat{y}) = |y - \hat{y}| - \text{regression}$$

# Supervised learning setup: putting it together



- Goal: design learning algorithm  $\mathcal{A}$  such that its output  $f$  on iid training data  $S$  has low generalization error

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

Supervised learning algorithm: decision trees

# Example: course recommendation

- Build a software: given a student, recommend a set of courses that s/he likes
- Induced supervised learning problem:





# Decision trees: basic terminology

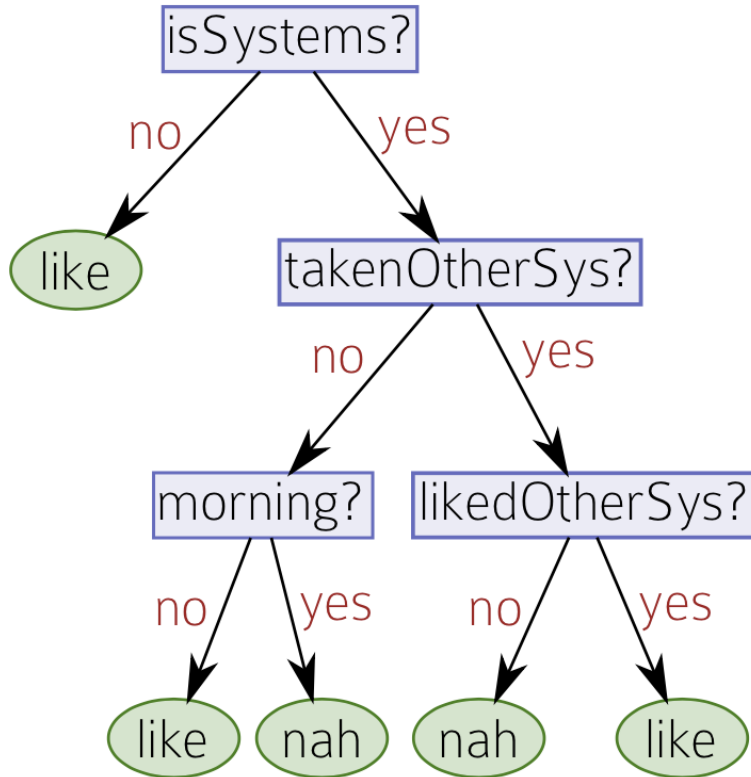


Figure 1.2: A decision tree for a course recommender system, from which the in-text “dialog” is drawn.

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:

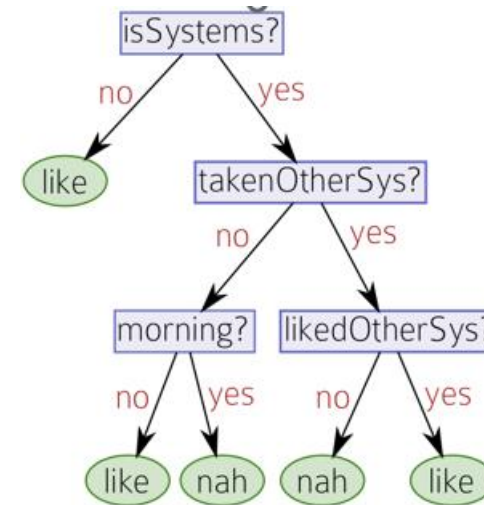
---

**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: how to design a learning algorithm  $\mathcal{A}$  that can build trees  $f$  from training data?

# Course recommendation: simplified setup

training data  $S$

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

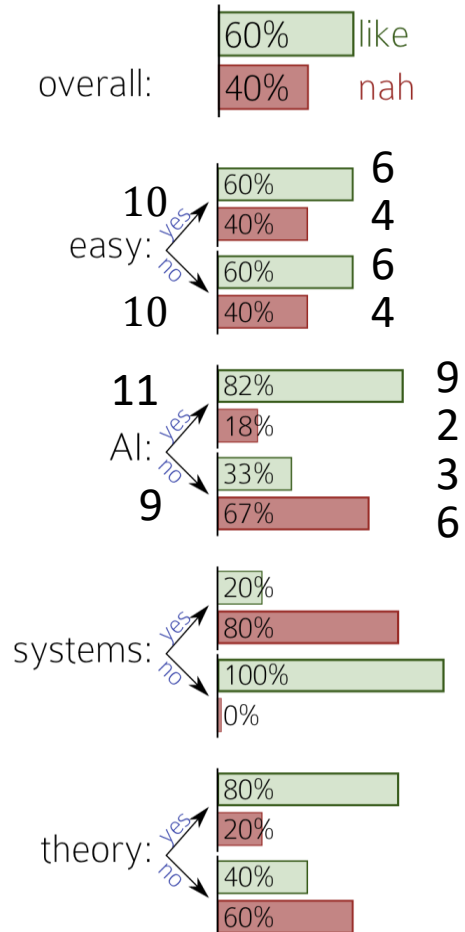
data point  
feature  
label

# Generalization error vs. training error

- The training data  $S = \{(x_i, y_i)\}_{i=1}^m$
- Given a predictor  $f$ , its training error  $L_S(f) = \mathbb{E}_{(x,y) \sim S} \ell(y, f(x)) = \frac{1}{m} \sum_{i=1}^m \ell(y_i, f(x_i))$
- Consider zero-one loss,  $\ell(y, \hat{y}) = I(y \neq \hat{y}) \Rightarrow L_S(f) = \mathbb{E}_{(x,y) \sim S} I(y \neq f(x)) = P_{(x,y) \sim S} (y \neq f(x))$
- Heuristic:  $f$  with low  $L_S(f) \Rightarrow f$  with low  $L_D(f)$ 
  - Also known as the “Empirical risk minimization” (ERM) approach
  - Issues with ERM?
- How easy is it to compute a decision tree  $f$  that minimize  $L_S(f)$ ?
  - $k$ -node decision tree,  $d$ -dimensional data  $\Rightarrow$  at least  $O(d^k)$  time complexity
  - Can we design efficient algorithms?

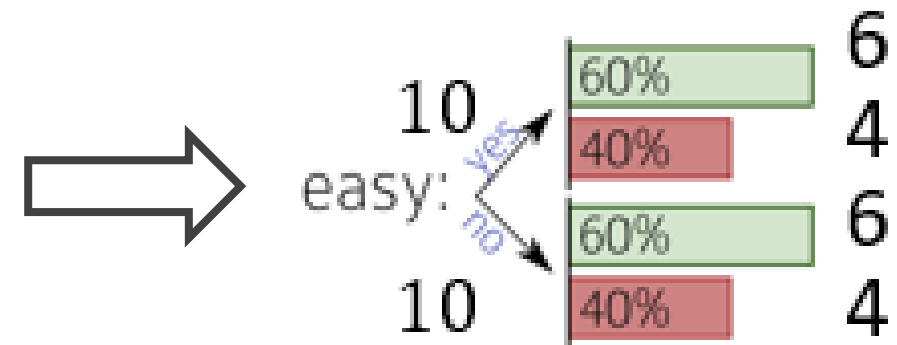
# Decision tree training: single level case

- 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 is better, 'easy' or 'AI'?
- Best training accuracy using 'easy':  
 $(\max(6,4) + \max(6,4)) / 20 = 0.6$
- Best training accuracy using 'AI':  
 $(\max(9,2) + \max(3,6)) / 20 = 0.75$



# Histogram calculation

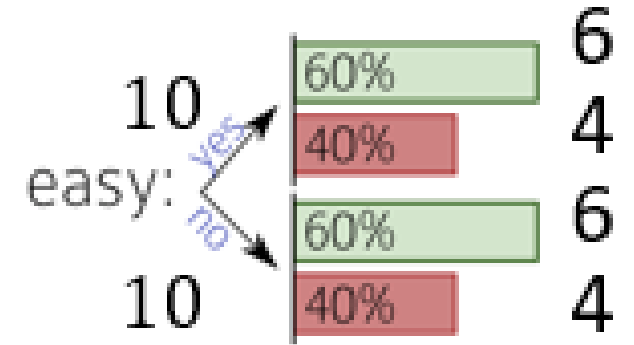
Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
o	n	n	n	n	y
o	y	n	n	y	y
o	n	y	n	y	n
o	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y



# Decision tree training: single level case

- In formula:

$$\text{Score}(f, S) := \max\left(P_S(y = +, x_f = \text{yes}), P_S(y = -, x_f = \text{yes})\right) \\ + \max\left(P_S(y = +, x_f = \text{no}), P_S(y = -, x_f = \text{no})\right)$$



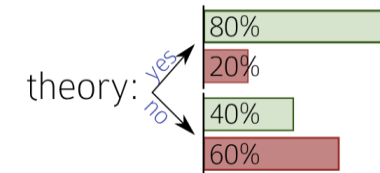
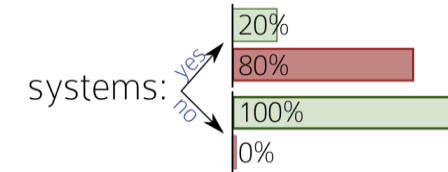
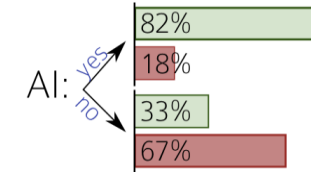
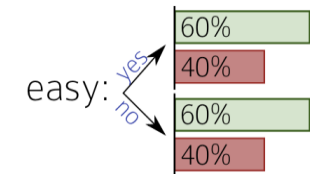
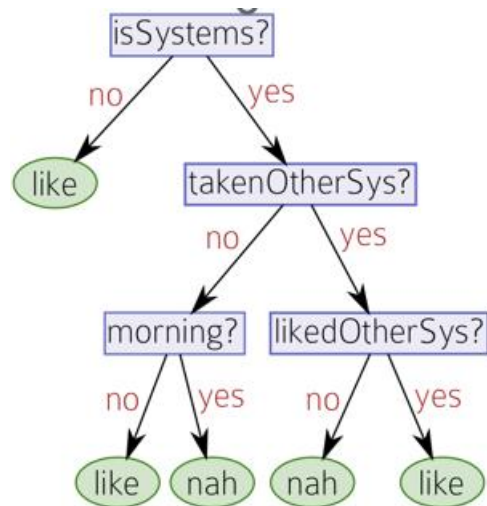
- Written in conditional probability:

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

- e.g.  $\text{Score}(\text{'easy'}, S) = \max(0.6, 0.4) \times 0.5 + \max(0.6, 0.4) \times 0.5 = 0.6$

# Decision tree training: general level case

- High-level idea: greedy + divide & conquer
- Build the root of the tree greedily
- Build the left and left subtrees *recursively*
- When to stop the recursion?





---

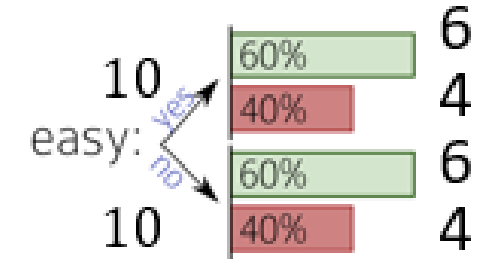
**Algorithm 1** `DECISIONTREETRAIN`(*data*, *remaining features*)

---

```
1: guess ← most frequent answer in data           // default answer for this data
2: if the labels in data are unambiguous then
3:   return LEAF(guess)                          // base case: no need to split further
4: else if remaining features is empty then
5:   return LEAF(guess)                          // base case: cannot split further
6: else                                             // we need to query more features
7:   for all  $f \in \textit{remaining features}$  do
8:     NO ← the subset of data on which  $f=no$ 
9:     YES ← the subset of data on which  $f=yes$ 
10:    score[ $f$ ] ← # of majority vote answers in NO
11:                + # of majority vote answers in YES
12:  end for
13:   $f \leftarrow$  the feature with maximal score( $f$ )
14:  NO ← the subset of data on which  $f=no$ 
15:  YES ← the subset of data on which  $f=yes$ 
16:  left ← DECISIONTREETRAIN(NO, remaining features \ { $f$ })
17:  right ← DECISIONTREETRAIN(YES, remaining features \ { $f$ })
18:  return NODE( $f$ , left, right)
19: end if
```

answer=label

unambiguous=achieves 100% acc.



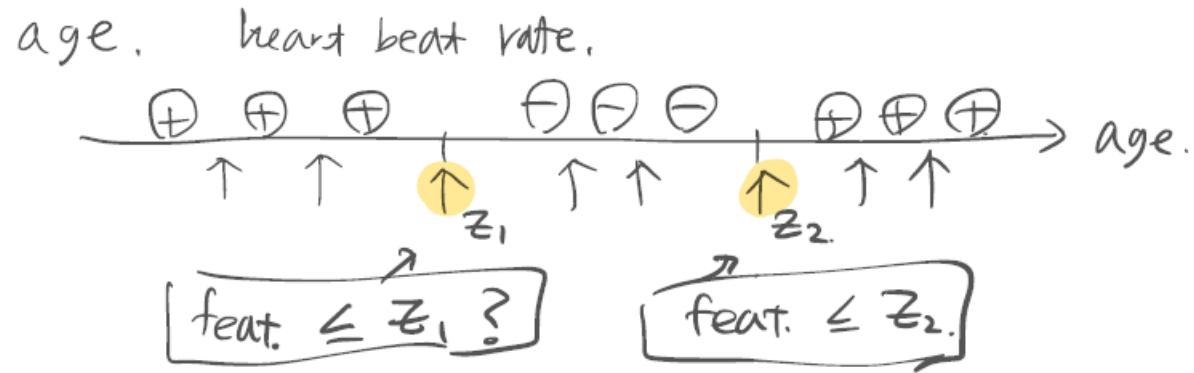
$$\text{Score}(f, S) := \max \left( P_S(y = +, x_f = \text{yes}), P_S(y = -, x_f = \text{yes}) \right) + \max \left( P_S(y = +, x_f = \text{no}), P_S(y = -, x_f = \text{no}) \right)$$

// the accuracy we would get if we only queried on  $f$

Q: is this algorithm guaranteed to terminate?

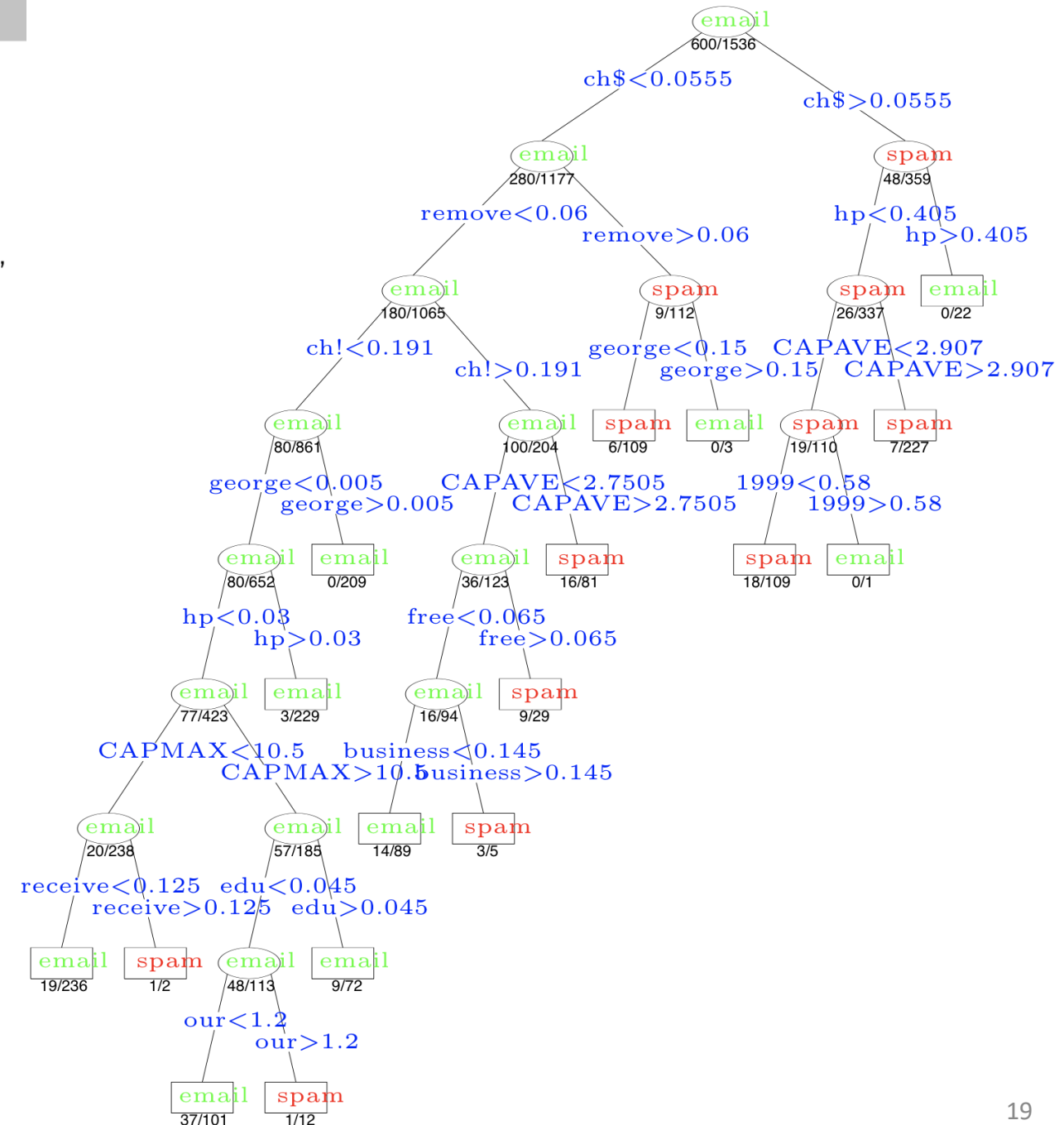
# Dealing with various types of features

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



# 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



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

# Decision tree training: generalized scores

- New viewpoint: the majority vote accuracy = a measure of informativeness
- Approach: find a split that maximizes informativeness / reduces uncertainty, i.e. “chaoticness”
- Uncertainty measures of population:

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:**

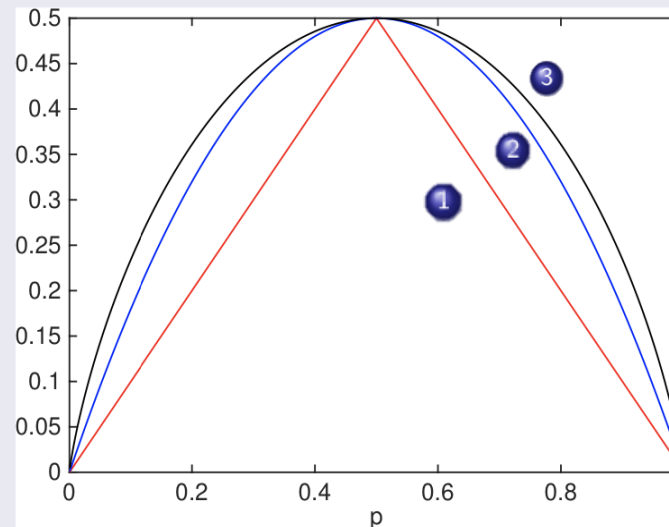
$$u(S) := \min\{p, 1 - p\}$$

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 scores

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

## 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:

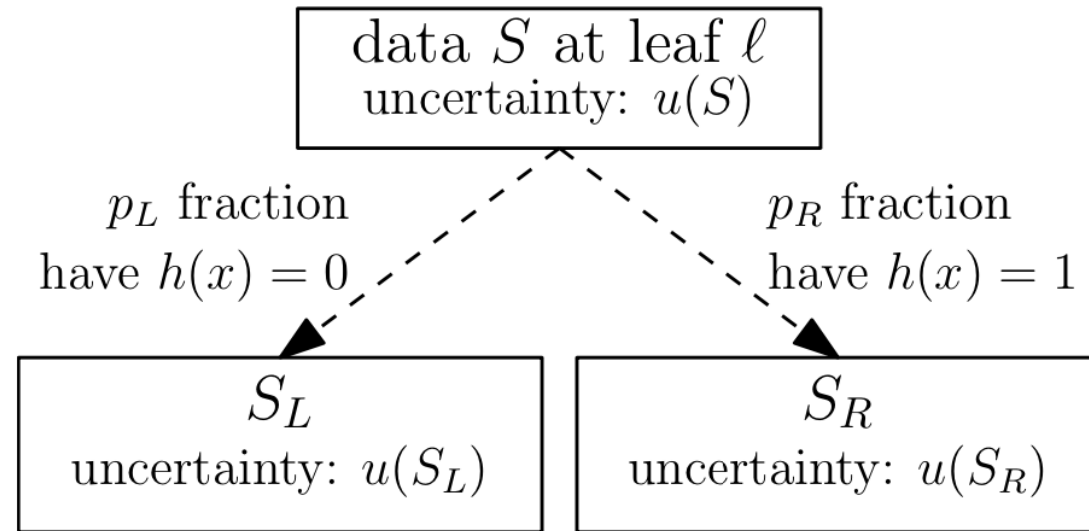
$$u(S) := \sum_{k \in \mathcal{Y}} p_k \log \frac{1}{p_k}$$

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)

# Decision tree training: generalized scores

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



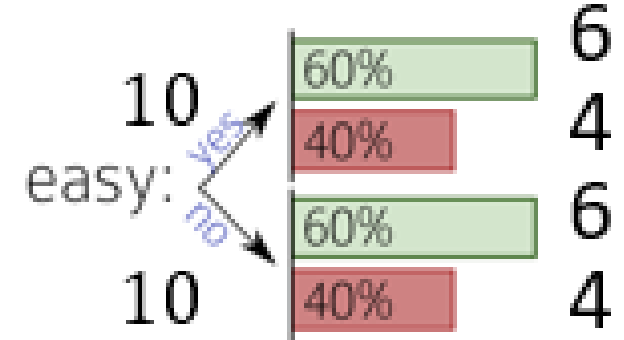
The **reduction in uncertainty** from using rule  $h$  at leaf  $\ell$  is

$$u(S) - \left( p_L \cdot u(S_L) + p_R \cdot u(S_R) \right) \boxed{=: \text{Score}(h, S) \text{ (Generalized)}}$$

# Generalized splitting criteria in action

- Entropy uncertainty:

$$u(S) = \sum_{y_0 \in \{+, -\}} P_S(y = y_0) \log \frac{1}{P_S(y=y_0)}$$



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

Diagram illustrating the components of the score function:

- $P_S(x_f = \text{no})$  (pointing to the left branch of the score function)
- $\sum_{y_0 \in \{+, -\}} P_S(y = y_0 \mid x_f = \text{no}) \log \frac{1}{P_S(y = y_0 \mid x_f = \text{no})}$  (pointing to the right branch of the score function)

- E.g. for the above  $S$ ,  $\text{Score}(\text{'easy'}, S) = 0$ , because:

- $u(S) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$ ,  $p_L = 0.5, p_R = 0.5$

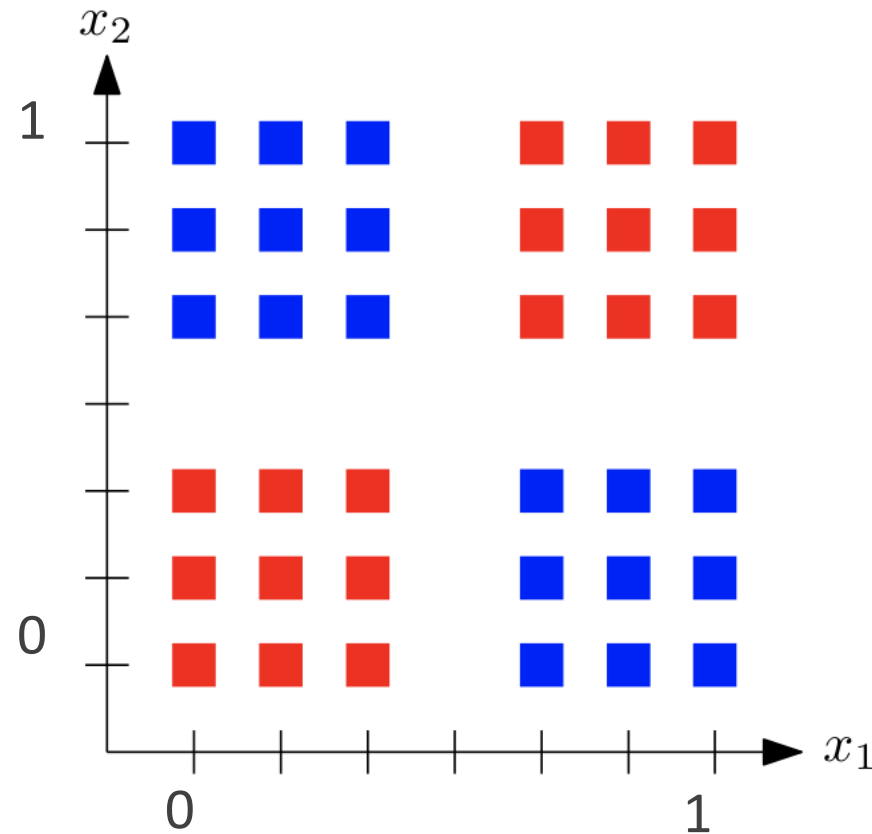
- $u(S_L) = u(S_R) = 0.6 \log \frac{1}{0.6} + 0.4 \log \frac{1}{0.4}$

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

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

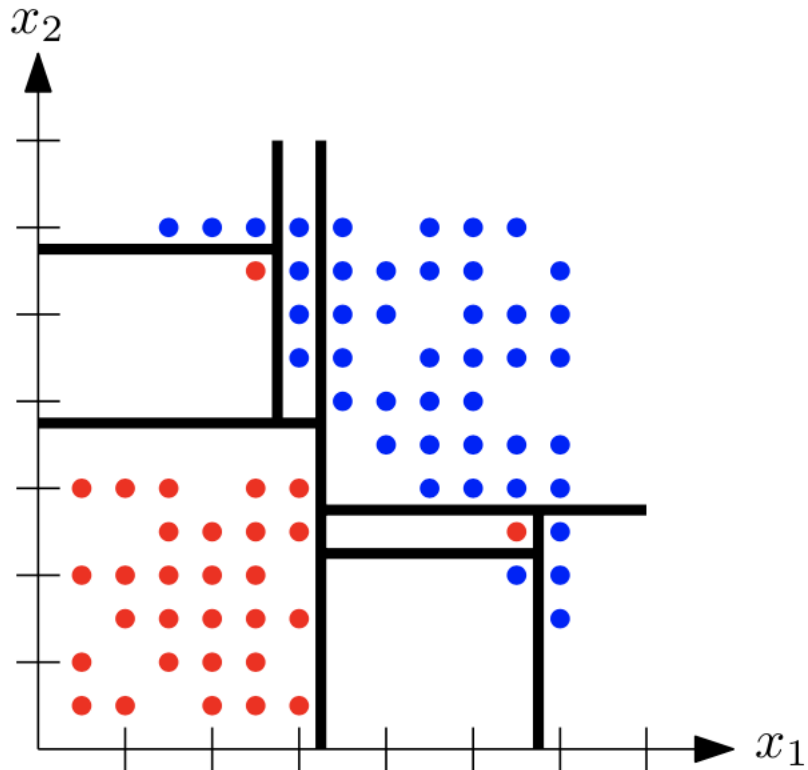


- Any axis-aligned split has no reduction on uncertainty on  $S$
- However, a depth-2 decision tree (with axis-aligned splits) has zero training error

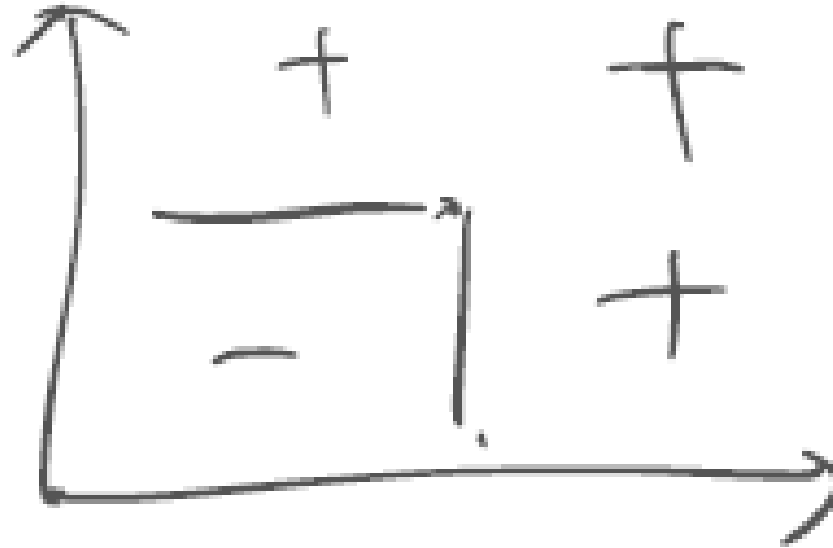


# Overfitting can happen

- “Spurious” patterns can be learned.



A better alternative:

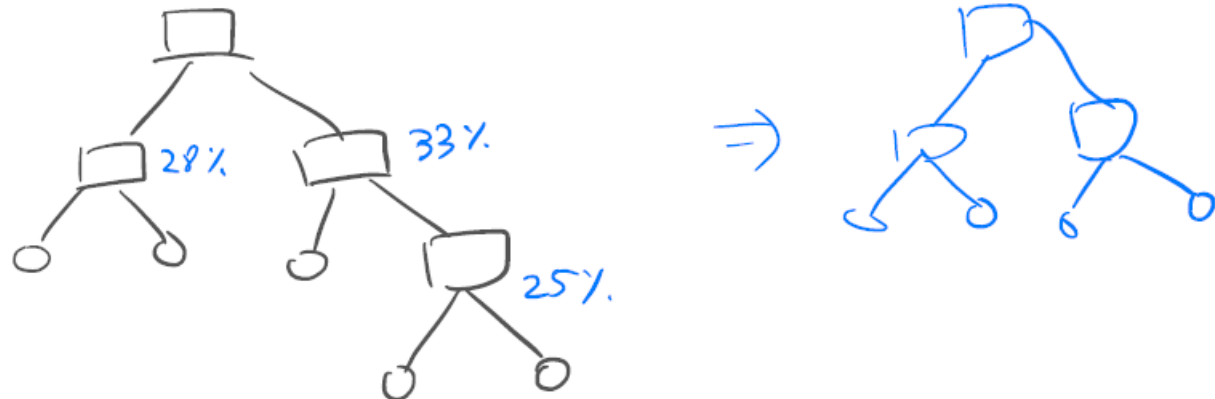


# Prevent overfitting by pruning

- split the data into training set and validation set (aka development set)



- build a decision tree based on the training set; compute the base validation set error
- while true
  - for each parent-to-leaf node, compute the validation set error when pruned
  - if no one reduces the validation set error, break
  - otherwise, prune the one that reduces the validation set error the most



- Q: why don't we prune the tree based on training error?

# Next lecture (8/29)

- Supervised learning: what to do if the data distribution is *known*?
- Models, parameters, hyperparameters
- Practical considerations
- Assigned reading: CIML Chap. 2 (Limits of learning)
- HW 0 due

Thank you!  
Questions?