University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 2: Decision Tree

Slides adapted from Noah Smith

Administrivia

- Make sure that you enroll in Moodle and have access to Piazza
- Email me to introduce yourself, one of your core values, and a machine learning application you care about

Learning Objectives

- Understand the difference between memorization and generalization
- Understand feature extraction
- Understand the basics of decision tree

Outline

- Supervised learning
- Features
- Decision tree
- Information gain

Outline

- Supervised learning
- Features
- Decision tree
- Information gain

Memorization vs. Generalization

- What do you think are the differences?

Memorization vs. Generalization

- Consider the task of learning to drive given all the drivers' behavior data and sensor data in Boulder
  - Drive from Pearl street to Engineering center
  - Drive from the central park in NYC to Boulder

Memorization vs. Generalization

- Training data
- Testing data

Supervised learning

Hutzler #571 Banana Slicer

The only banana slicer you will ever need.

Gourmac's easy-to-use Banana Slicer provides a quick solution to slice a banana uniformly each and every time. Simply press the slicer on a peeled banana and the work is done. Safe, fun and easy for children to use. Kids just love eating bananas with this as their favorite kitchen tool. The Banana Slicer may also be used as a quick way to add healthy bananas to breakfast cereal or to make uniform slices for a fruit salad or ice cream dessert.

Data: X Labels: Y

- **Supervised methods** find patterns in **fully observed** data and then try to predict something from **partially observed** data.

Formal definition of supervised learning

- Labels Y, e.g., binary labels $y \in \{+1, -1\}$
- Instance space X, all the possible instances (based on data representation)
- Target function f: $X \to Y$ (f is unknown)

Formal definition of supervised learning

- Labels Y, e.g., binary labels $y \in \{+1, -1\}$
- Instance space X, all the possible instances (based on data representation)
- Target function f: $X \to Y$ (f is unknown)
- Example/instance $(\boldsymbol{x}, y)$
- Training data S_{train}: collection of examples observed by the algorithm

Formal definition of supervised learning

- Goal of a learning algorithm:

  Find a function $h : X \rightarrow Y$ from training data S_{train} so that h approximates f

performance

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

but we only have access to training error for a training sample $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{1}^{N} l(h(\boldsymbol{x}_i), y_i).$$

Supervised learning in a nutshell

$$S_{\text{train}} = \{(\boldsymbol{x}, y)\} \rightarrow h$$

Given loss function l and $(\boldsymbol{x}, y) \sim D$, expected loss is defined as:

$$\epsilon \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim D}\left[l(h(\boldsymbol{x}), y)\right] = \sum_{\boldsymbol{x}\in X, y\in Y} D(X = \boldsymbol{x}, Y = y) l(h(\boldsymbol{x}), y),$$

but we only have access to training error for a training sample $S_{\text{train}} = \{(\boldsymbol{x}_i, y_i), i = 1, \ldots, N\}$:

$$\hat{\epsilon} \triangleq \frac{1}{N} \sum_{1}^{N} l(h(\boldsymbol{x}_i), y_i).$$

Minimizing training error is not ideal.

No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.

No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.
  Corollary I: there is no single ML algorithm that works for everything.
  Corollary II: every successful ML algorithm makes assumptions.

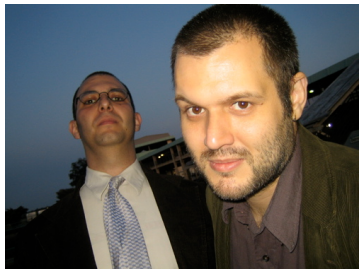
No Free Lunch Theorems

- No free lunch for supervised machine learning [Wolpert, 1996]:
  in a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no a priori distinctions between learning algorithms.
  Corollary I: there is no single ML algorithm that works for everything.
  Corollary II: every successful ML algorithm makes assumptions.
- No free lunch for search/optimization [Wolpert and Macready, 1997]: All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions.

Outline

- Supervised learning
- Features
- Decision tree
- Information gain

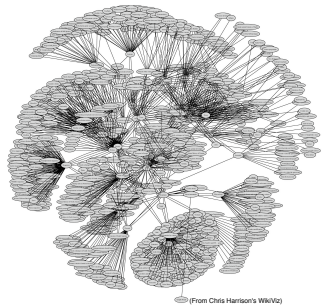
Features



$\rightarrow \langle 1.5, 3.2, -5.1, \ldots, 4.2 \rangle$

Republican nominee George Bush said he felt nervous as he voted today in his adopted home state of Texas, where he ended...

$\rightarrow \langle 1, 0, 0, 0, 5, 0, 9, 3, 1, \ldots, 0 \rangle$



$\rightarrow \begin{bmatrix} 1 & 0 & 1 & \ldots & 0 \\ 0 & 1 & 1 & \ldots & 0 \\ 1 & 0 & 0 & \ldots & 1 \\ & \ldots & & & \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}$

Features

Let ϕ be a function that maps from inputs (x) to values.

Features

Let ϕ be a function that maps from inputs (x) to values.

- If ϕ maps to $\{0, 1\}$, we call it a “binary feature (function).”

Features

Let ϕ be a function that maps from inputs (x) to values.

- If ϕ maps to $\{0, 1\}$, we call it a “binary feature (function).”
- If ϕ maps to $\mathbb{R}$, we call it a “real-valued feature (function).”

Features

Let ϕ be a function that maps from inputs (x) to values.

- If ϕ maps to $\{0, 1\}$, we call it a “binary feature (function).”
- If ϕ maps to $\mathbb{R}$, we call it a “real-valued feature (function).”
- Feature functions can map to categorical values, ordinal values, integers, and more.

Features

- Let us have an interactive example to think through data representation!

Features

- Let us have an interactive example to think through data representation!

id	rent	income	urban	state	car value	car year
1	yes	50,000	no	CO	20,000	2010
2	yes	70,000	no	CO	30,000	2012
3	no	250,000	yes	CO	55,000	2017
4	yes	200,000	yes	NY	50,000	2016

Understanding assumptions in features

- The methods we'll study make assumptions about the data on which they are applied. E.g.,
  - Documents can be analyzed as a sequence of words;
  - or, as a "bag" of words.
  - Independent of each other;
  - or, as connected to each other
- What are the assumptions behind the methods?
- When/why are they appropriate?
- Much of this is an art, and it is inherently dynamic

Outline

- Supervised learning
- Features
- Decision tree
- Information gain

Features

Data derived from

`https://archive.ics.uci.edu/ml/datasets/Auto+MPG`

mpg; cylinders; displacement; horsepower; weight; acceleration; year; origin

```
18.0   8   307.0      130.0      3504.      12.0   70  1
15.0   8   350.0      165.0      3693.      11.5   70  1
18.0   8   318.0      150.0      3436.      11.0   70  1
16.0   8   304.0      150.0      3433.      12.0   70  1
17.0   8   302.0      140.0      3449.      10.5   70  1
15.0   8   429.0      198.0      4341.      10.0   70  1
14.0   8   454.0      220.0      4354.       9.0   70  1
14.0   8   440.0      215.0      4312.       8.5   70  1
14.0   8   455.0      225.0      4425.      10.0   70  1
15.0   8   390.0      190.0      3850.       8.5   70  1
15.0   8   383.0      170.0      3563.      10.0   70  1
14.0   8   340.0      160.0      3609.       8.0   70  1
15.0   8   400.0      150.0      3761.       9.5   70  1
14.0   8   455.0      225.0      3086.      10.0   70  1
24.0   4   113.0      95.00      2372.      15.0   70  3
22.0   6   198.0      95.00      2833.      15.5   70  1
18.0   6   199.0      97.00      2774.      15.5   70  1
21.0   6   200.0      85.00      2587.      16.0   70  1
27.0   4   97.00      88.00      2130.      14.5   70  3
26.0   4   97.00      46.00      1835.      20.5   70  2
25.0   4   110.0      87.00      2672.      17.5   70  2
24.0   4   107.0      90.00      2430.      14.5   70  2
```

Goal: predict whether mpg is < 23 (“bad” = 0) or above (“good” = 1) given other attributes (other columns).

201 “good” and 197 “bad”; guessing the most frequent class (good) will get 50.5% accuracy.

Contingency Table

values of y	values of feature ϕ			
	v_1	v_2	$\cdots$	v_K
0				
1				

Decision Stump Example

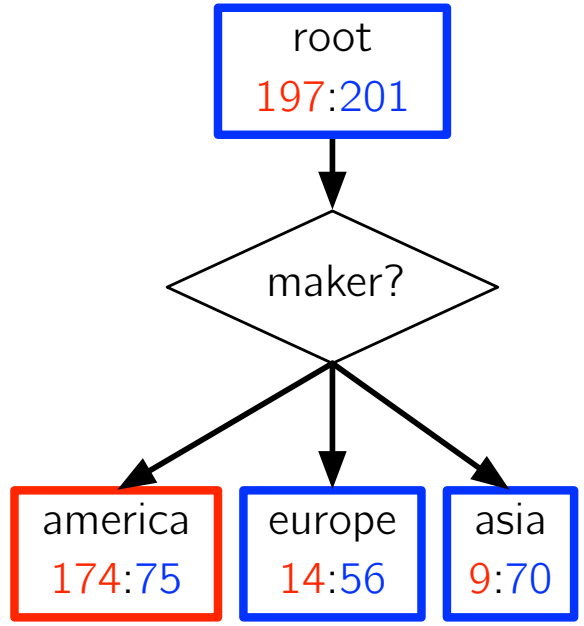
y	maker		
	america	europe	asia
0	174	14	9
1	75	56	70

Decision Stump Example

y	maker		
	america	europe	asia
0	174	14	9
1	75	56	70
	↓	↓	↓
	0	1	1

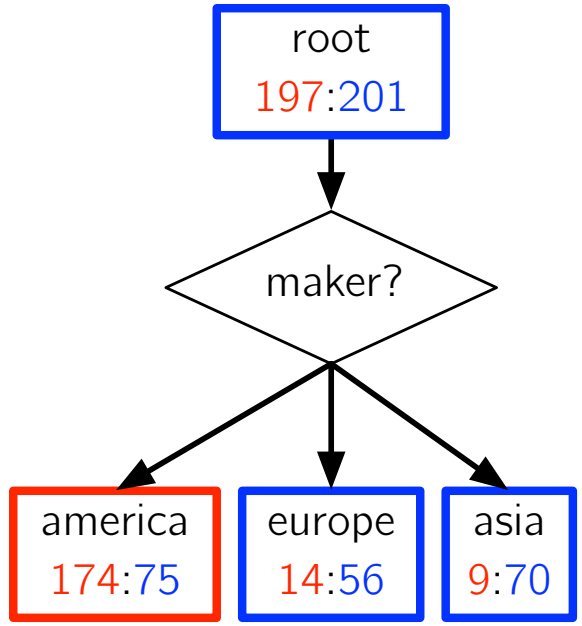
Decision Stump Example

y	maker		
	america	europe	asia
0	174	14	9
1	75	56	70
	↓	↓	↓
	0	1	1



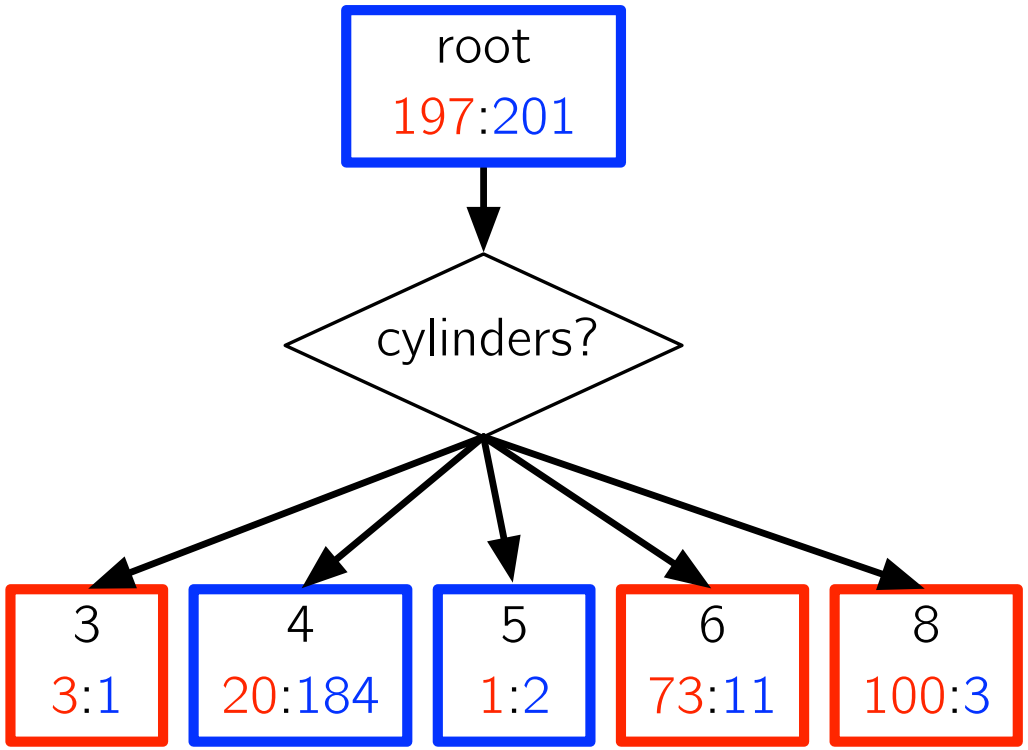
Decision Stump Example

y	maker		
	america	europe	asia
0	174	14	9
1	75	56	70
	↓	↓	↓
	0	1	1

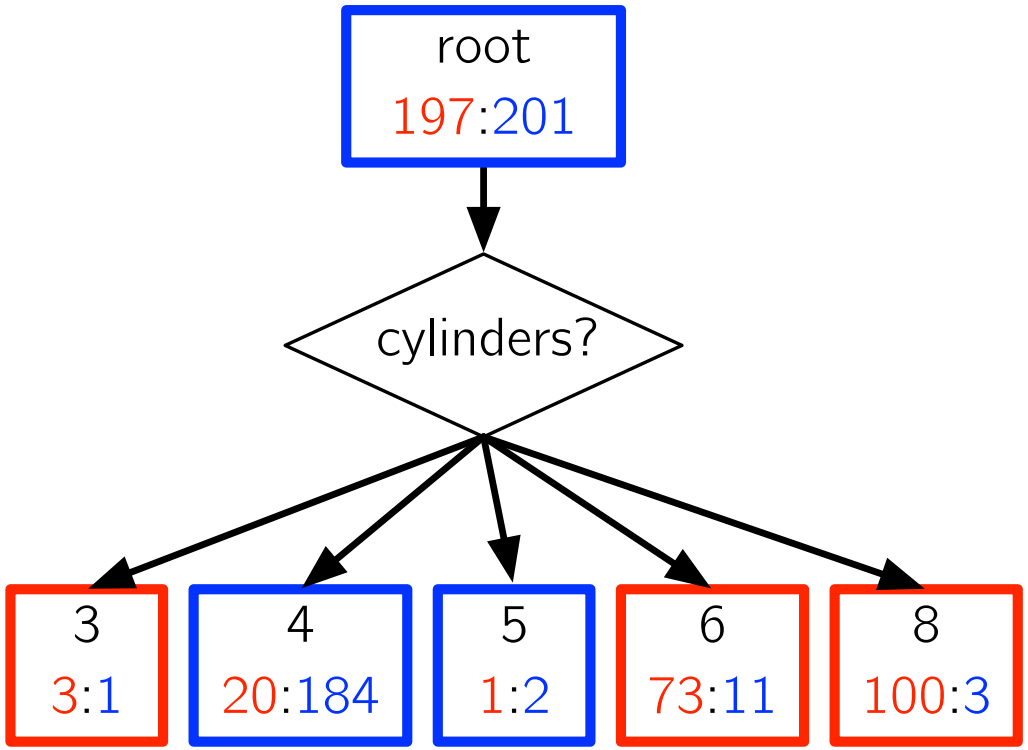


Errors: 75 + 14 + 9 = 98 (about 25%)

Decision Stump Example



Decision Stump Example

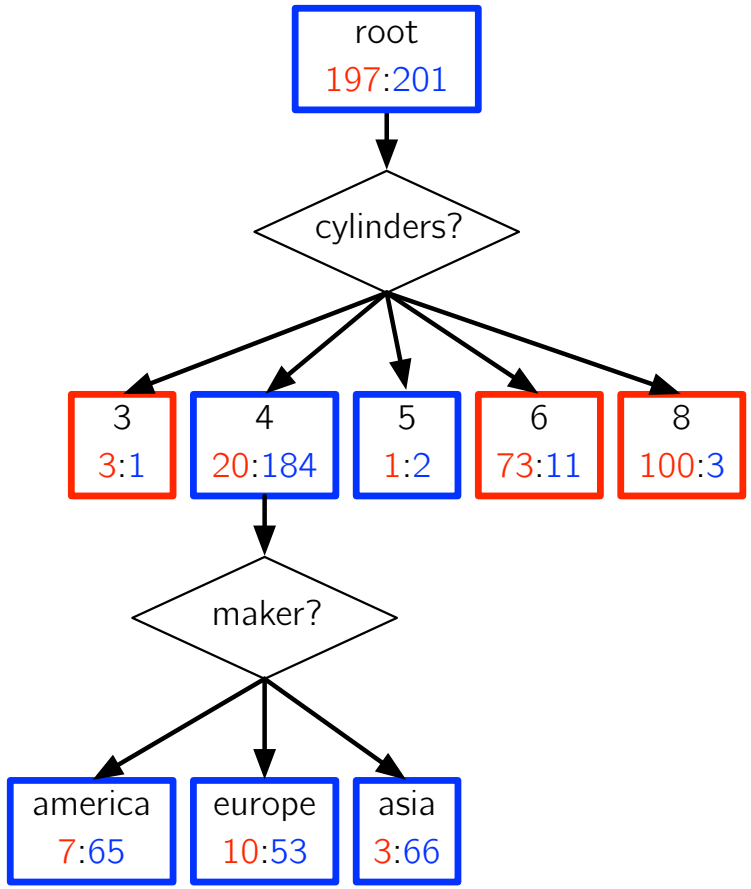


Errors: 1 + 20 + 1 + 11 + 3 = 36 (about 9%)

Key Idea: Recursion

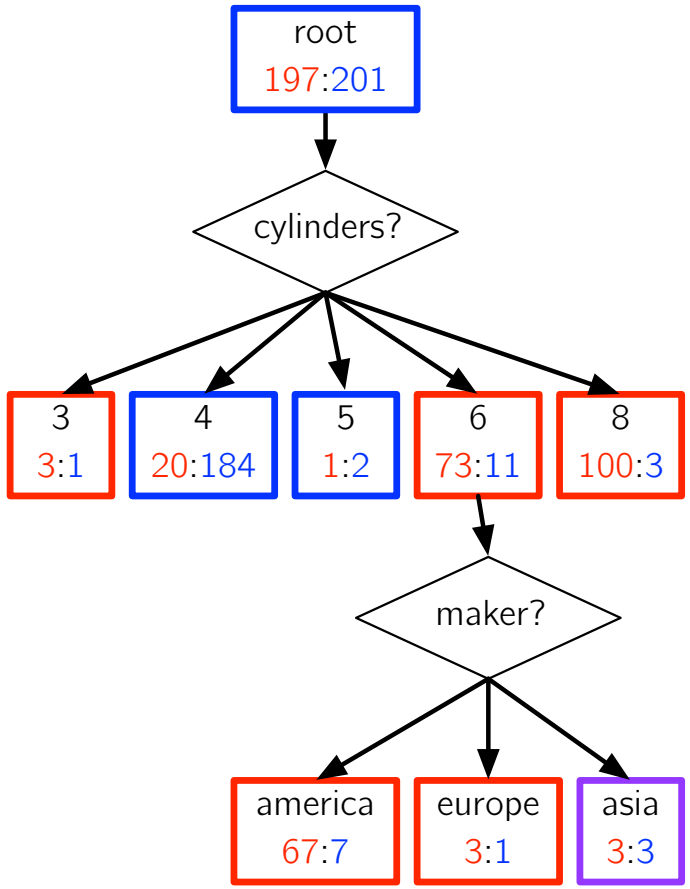
- A single feature partitions the data.
- For each partition, we could choose another feature and partition further.
- Applying this recursively, we can construct a decision tree.

Decision Tree Example



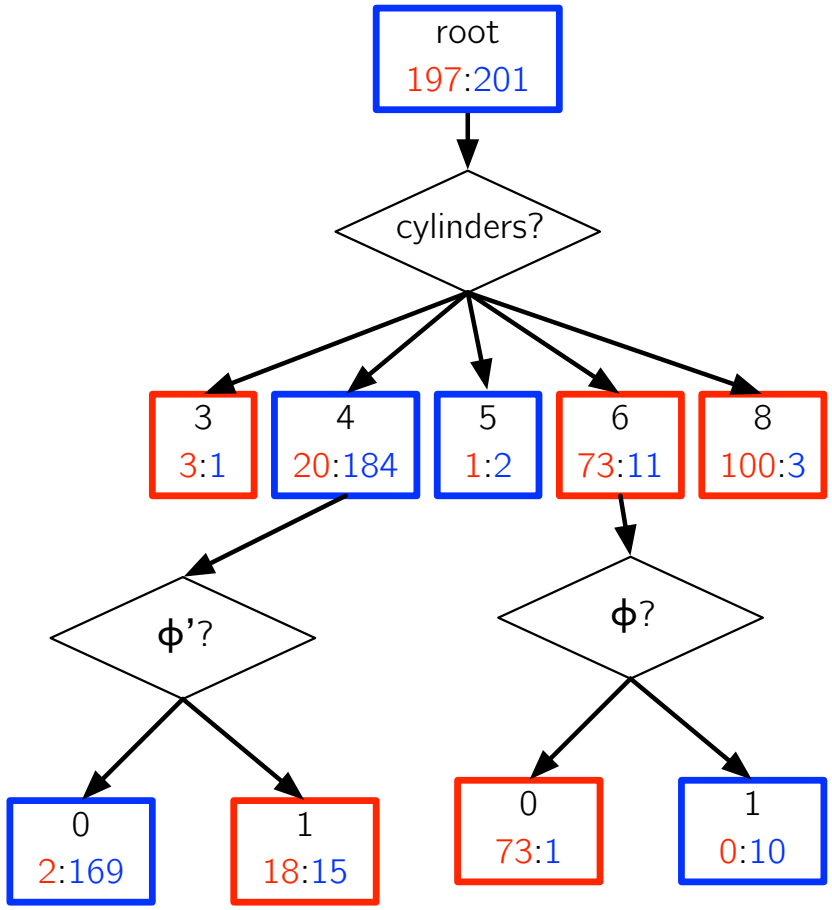
Error reduction compared to the cylinders stump?

Decision Tree Example



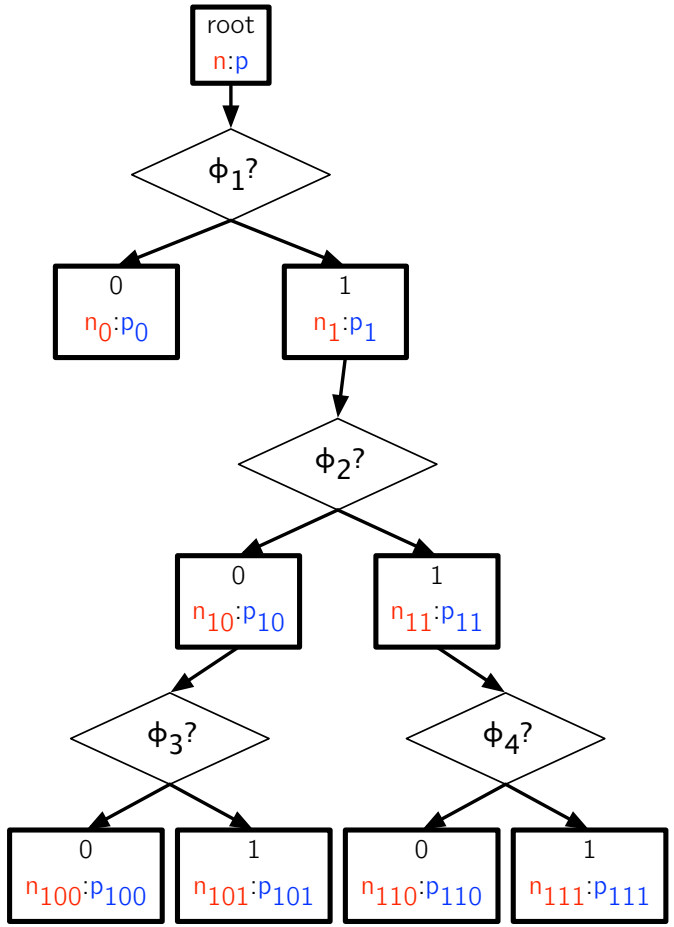
Error reduction compared to the cylinders stump?

Decision Tree Example

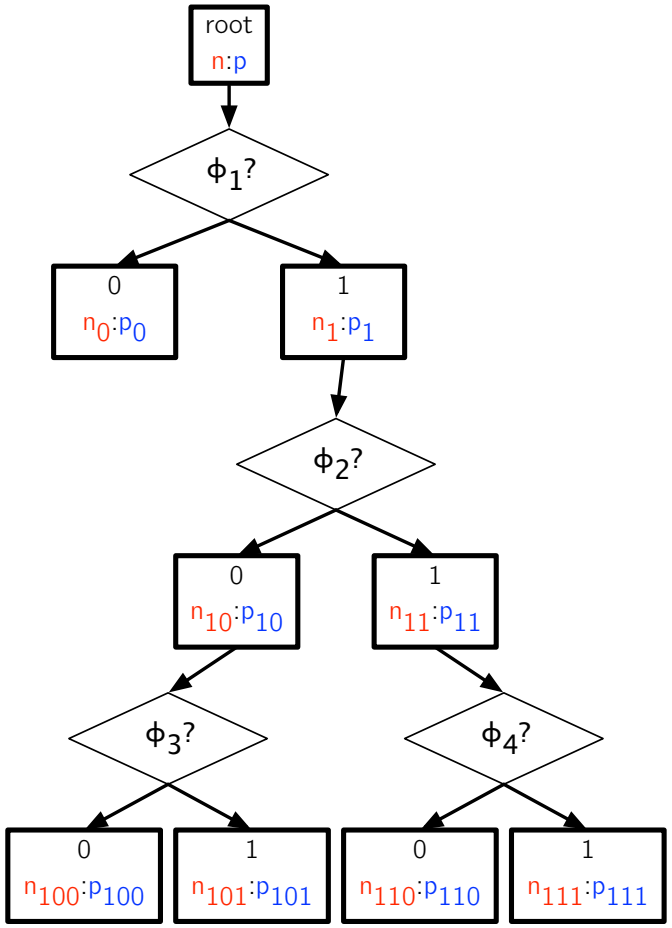


Error reduction compared to the cylinders stump?

Decision Tree: Making a Prediction



Decision Tree: Making a Prediction



Algorithm: DTREETEST

Data: decision tree t, input example x

Result: predicted class

if t *has the form* LEAF(y) **then**

    return y;

else

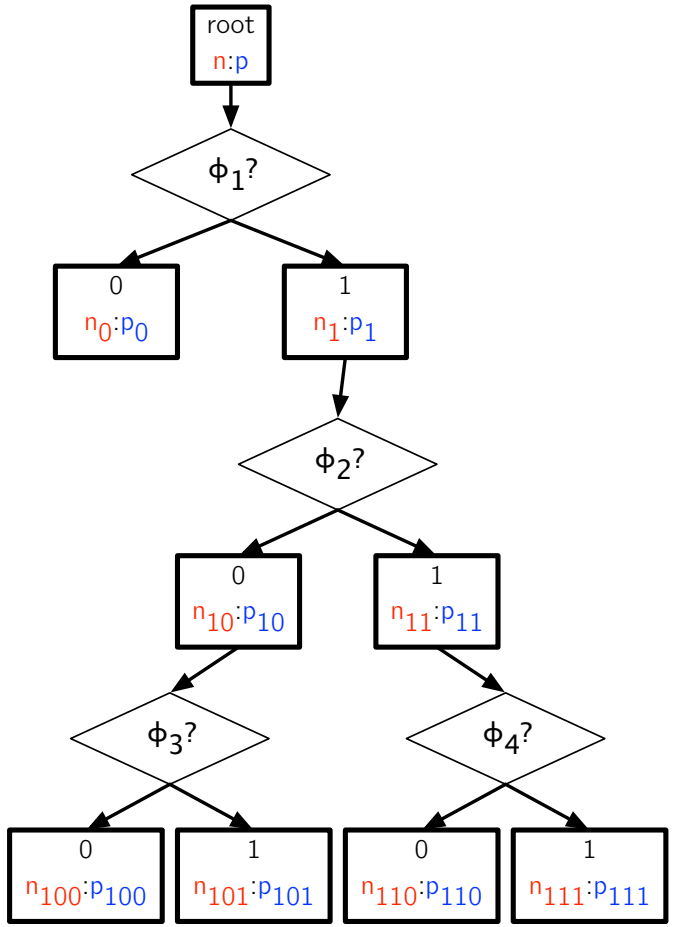
    # $t.\phi$ is the feature associated with t;

    # t.child(v) is the subtree for value v;

    return DTREETEST(t.child($t.\phi(x)$), x));

end

Decision Tree: Making a Prediction



Equivalent boolean formulas:

$$(\phi_1 = 0) \Rightarrow [\![n_0 < p_0]\!]$$
$$(\phi_1 = 1) \wedge (\phi_2 = 0) \wedge (\phi_3 = 0) \Rightarrow [\![n_{100} < p_{100}]\!]$$
$$(\phi_1 = 1) \wedge (\phi_2 = 0) \wedge (\phi_3 = 1) \Rightarrow [\![n_{101} < p_{101}]\!]$$
$$(\phi_1 = 1) \wedge (\phi_2 = 1) \wedge (\phi_4 = 0) \Rightarrow [\![n_{110} < p_{110}]\!]$$
$$(\phi_1 = 1) \wedge (\phi_2 = 1) \wedge (\phi_4 = 1) \Rightarrow [\![n_{111} < p_{111}]\!]$$

Tangent: How Many Formulas?

Assume we have D binary features.

Tangent: How Many Formulas?

Assume we have D binary features.

Each feature could be set to 0, or set to 1, or excluded (wildcard/don't care).

Tangent: How Many Formulas?

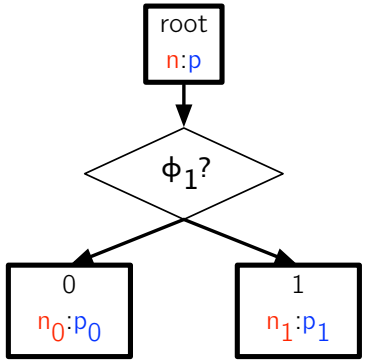
Assume we have D binary features.

Each feature could be set to 0, or set to 1, or excluded (wildcard/don't care).

3^D formulas.

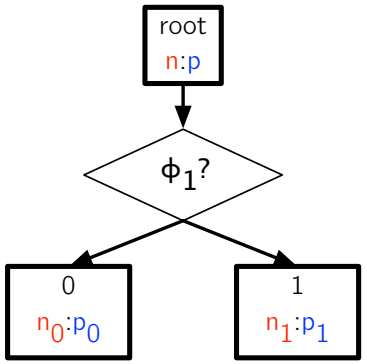
Growing a Decision Tree

Growing a Decision Tree



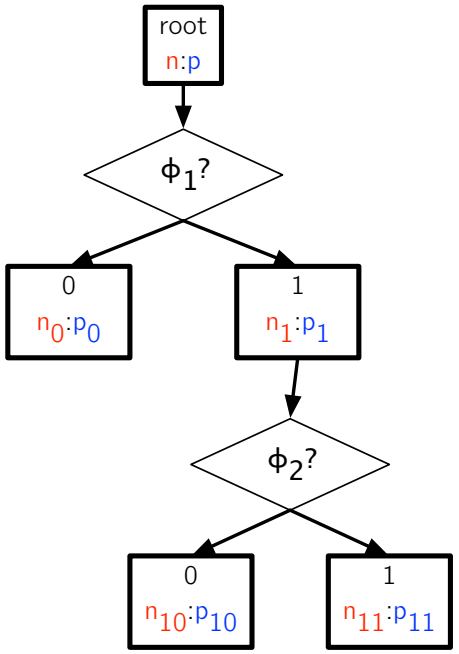
We chose feature ϕ_1. Note that $n = n_0 + n_1$ and $p = p_0 + p_1$.

Growing a Decision Tree

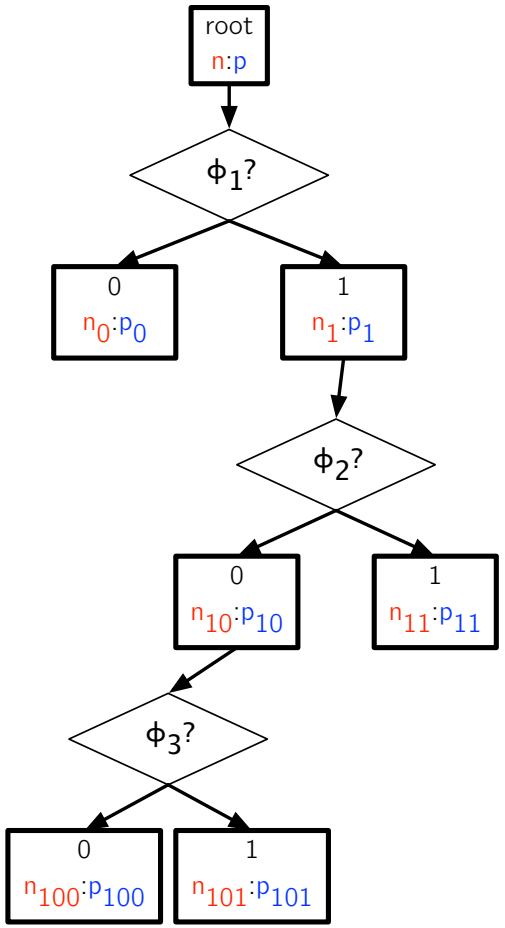


We chose not to split the left partition. Why not?

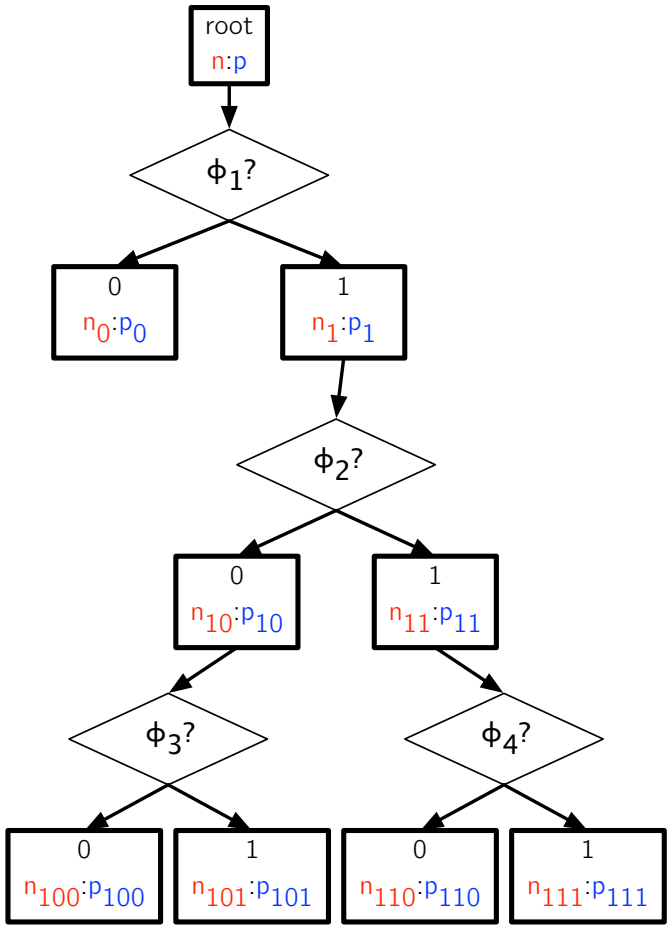
Growing a Decision Tree



Growing a Decision Tree



Growing a Decision Tree



Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN

Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$)});
end

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN

Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$)});
end

Does this algorithm always terminate? Why?

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN
Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$)});
end

Φ is finite and every call will either reach a leaf node or reduce the size of feature set by 1.

Greedily Building a Decision Tree (Binary Features)

Algorithm: DTREETRAIN

Data: data D, feature set Φ
Result: decision tree
if *all examples in* D *have the same label* y*, or* Φ *is empty and* y *is the best guess*
then
  return LEAF(y);
else
  for *each feature* ϕ *in* Φ **do**
    partition D into D_0 and D_1 based on ϕ-values;
    let mistakes(ϕ) = (non-majority answers in D_0) + (non-majority answers in D_1);
  end
  let ϕ^* be the feature with the smallest number of mistakes;
  return NODE(ϕ^*, {0 $\rightarrow$ DTREETRAIN($D_0, \Phi \setminus \{\phi^*\}$), 1 $\rightarrow$ DTREETRAIN($D_1, \Phi \setminus \{\phi^*\}$)});
end

Is the algorithm guaranteed to find optimal decision tree?

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?

One can always get the optimal training error by splitting based on all features.

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?
One can always get the optimal training error by splitting based on all features.

Optimal decision tree refers to the smallest tree that minimizes the training error.

A	B	C	$-$	$+$
0	0	0	10	5
0	0	1	5	20
0	1	0	5	20
0	1	1	10	5
1	0	0	30	5
1	0	1	5	10
1	1	0	5	10
1	1	1	30	5

Greedily Building a Decision Tree (Binary Features)

What is the optimal training error?

One can always get the optimal training error by splitting based on all features.

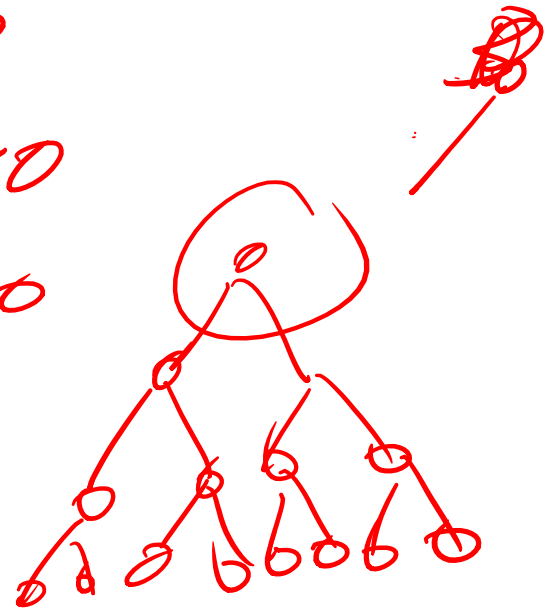
Optimal decision tree refers to the smallest tree that minimizes the training error.

A	B	C	$-$	$+$
0	0	0	10	5
0	0	1	5	20
0	1	0	5	20
0	1	1	10	5
1	0	0	30	5
1	0	1	5	10
1	1	0	5	10
1	1	1	30	5

A: 60

B: 80

C: 80



The greedy algorithm splits by A first, but the optimal tree only needs B and C.

Outline

- Supervised learning
- Features
- Decision tree
- Information gain

Information gain as a splitting criterion

- Inspired by information theory
- Entropy: measure of impurity of set of examples

Entropy

$$H(X) = -\sum_c p_c \log_2(p_c),$$

where p_c is the fraction of examples in class c. Note that for binary classification, let p be the fraction in the positive class, then

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Entropy

$$H(X) = -\sum_c p_c \log_2(p_c),$$

where p_c is the fraction of examples in class c. Note that for binary classification, let p be the fraction in the positive class, then

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

What is the largest/smallest entropy?

Entropy

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

What is the largest/smallest entropy?

$$0 \leq p \leq 1$$

Entropy

$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$$

What is the largest/smallest entropy?

$$0 \leq p \leq 1$$

- When all examples are in the same class, entropy is 0
- When samples are equally balanced, entropy is 1

Splitting

Consider the tennis problem now with binary features

sun	wind	humidity	tennis
sunny	windy	not humid	tennis
sunny	not windy	not humid	tennis
not sunny	not windy	humid	no tennis
sunny	windy	humid	no tennis

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Easy: the root node is balanced, so the entropy is 1

Splitting

Converting to features and labels

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is the entropy of the root node?

Easy: the root node is balanced, so the entropy is 1

$$p = \frac{1}{2} \Rightarrow H(X) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Information gain

The higher entropy is, the lower the information is.

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children
Splitting based on feature i

- X_{parent}: training subset of the parent node
- $X_{i,\text{left}}$: training subset of the left node
- $X_{i,\text{right}}$: training subset of the right node

Information gain

The higher entropy is, the lower the information is.
Information gain is defined as the difference between impurity at the parent and (weighted average) of impurity at the children
Splitting based on feature i

- X_{parent}: training subset of the parent node
- $X_{i,\text{left}}$: training subset of the left node
- $X_{i,\text{right}}$: training subset of the right node

$$IG(X_{\text{parent}}, i) = H(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} H(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} H(X_{\text{right}})$$

Splitting

What is $IG(X, \text{sun})$?

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,sun}} : \{1, 1, 0\}$
- $X_{\text{right,sun}} : \{0\}$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,sun}} : \{1, 1, 0\}$
- $X_{\text{right,sun}} : \{0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,sun}}) = 0.918$
- $H(X_{\text{right,sun}}) = 0$

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{sun})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,sun}} : \{1, 1, 0\}$
- $X_{\text{right,sun}} : \{0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,sun}}) = 0.918$
- $H(X_{\text{right,sun}}) = 0$

$$IG(X, \text{sun}) = 1 - \frac{3}{4} * 0.918 - \frac{1}{4} * 0 = 0.3112$$

Splitting

What is $IG(X, \text{wind})$?

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{wind})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,wind}} : \{1, 0\}$
- $X_{\text{right,wind}} : \{1, 0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,wind}}) = 1$
- $H(X_{\text{right,wind}}) = 1$

$$IG(X, \text{wind}) = 1 - \frac{1}{2} * 1 - \frac{1}{2} * 1 = 0$$

Splitting

What is $IG(X, \text{humid})$?

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

Splitting

sun	wind	humidity	tennis
1	1	0	1
1	0	0	1
0	0	1	0
1	1	1	0

What is $IG(X, \text{humid})$?

- $X_{\text{parent}} : \{1, 1, 0, 0\}$
- $X_{\text{left,humid}} : \{1, 1\}$
- $X_{\text{right,humid}} : \{0, 0\}$

- $H(X_{\text{parent}}) = 1$
- $H(X_{\text{left,humid}}) = 0$
- $H(X_{\text{right,humid}}) = 0$

$$IG(X, \text{humid}) = 1 - \frac{1}{2} * 0 - \frac{1}{2} * 0 = 1$$

Splitting

- $IG(X, \text{sun}) = 0.3112$
- $IG(X, \text{wind}) = 0$
- $IG(X, \text{humid}) = 1$

Which feature should we split on?

Splitting

- $IG(X, \text{sun}) = 0.3112$
- $IG(X, \text{wind}) = 0$
- $IG(X, \text{humid}) = 1$

Which feature should we split on?
humid, since it brings the greatest information gain.

Different splitting criteria

$$IG(X_{\text{parent}}, i) = I(X_{\text{parent}}) - \frac{|X_{i,\text{left}}|}{|X_{\text{parent}}|} I(X_{\text{left}}) - \frac{|X_{i,\text{right}}|}{|X_{\text{parent}}|} I(X_{\text{right}})$$

- Entropy
- Misclassification error ($I = \min_c p_c$)
- Gini index ($I = 1 - \sum_c p_c^2$)

Recap

- Supervised learning
- Features
- Decision tree
- Information gain