Machine Learning (CSE 446): Decision Trees

Noah Smith

University of Washington
nasmith@cs.washington.edu

September 29, 2017

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

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

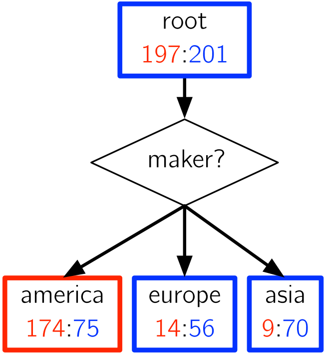
		values of feature ϕ			
		v_1	v_2	$\cdots$	v_K
values of y	0				
	1				

Decision Stump Example

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

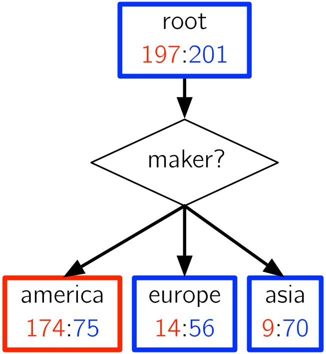
Decision Stump Example

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



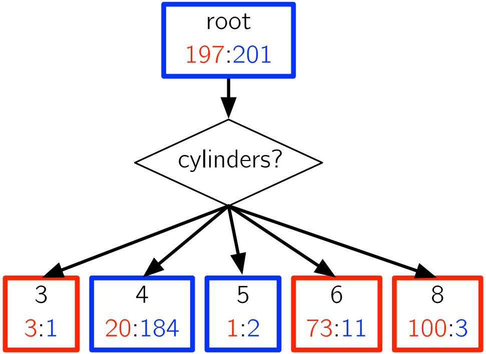
Decision Stump Example

y	maker: america	europe	asia
0	174	14	9
1	75	56	70
	$\downarrow$	$\downarrow$	$\downarrow$
	0	1	1

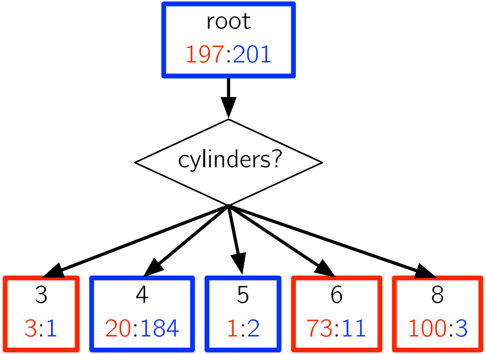


Errors: $75 + 14 + 9 = 98$ (about 25%)

Decision Stump Example



Decision Stump Example



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

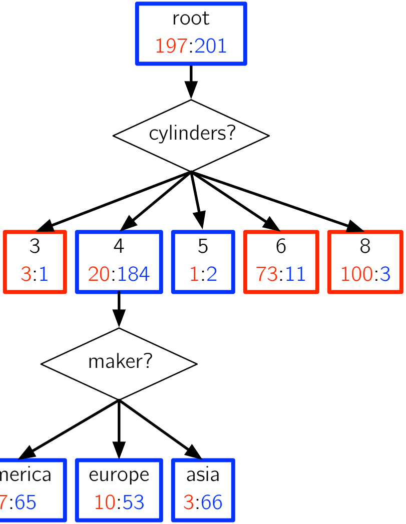
Key Idea: Recursion

A single feature **partitions** the data.

For each partition, we could choose another feature and partition further.

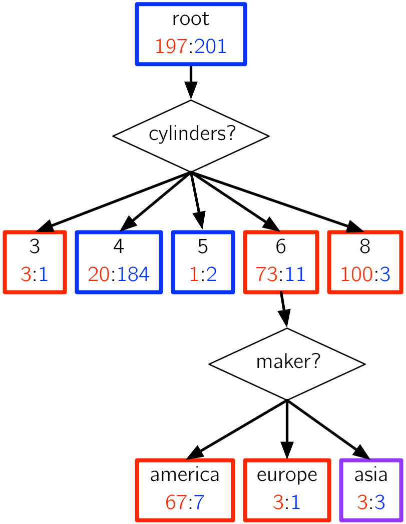
Applying this recursively, we can construct a **decision tree**.

Decision Tree Example



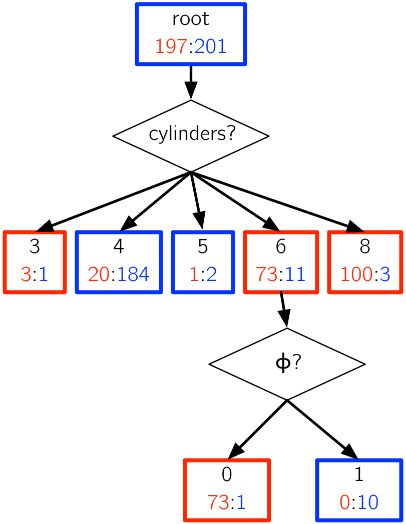
Error reduction compared to the cylinders stump?

Decision Tree Example



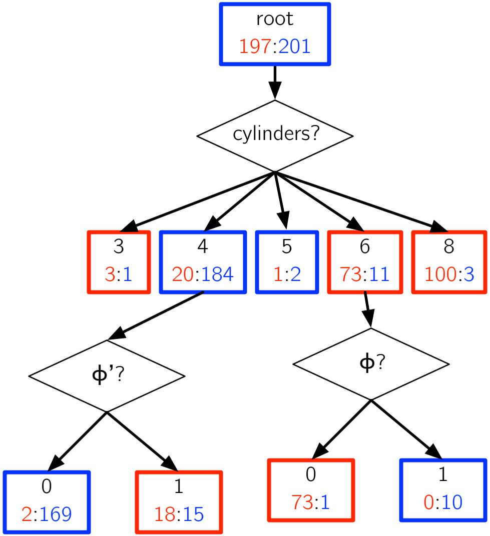
Error reduction compared to the cylinders stump?

Decision Tree Example



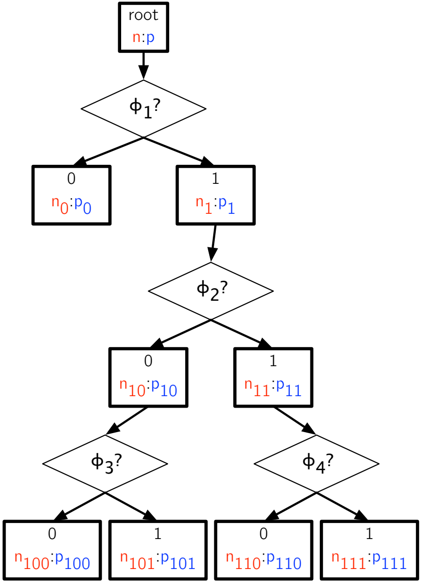
Error reduction compared to the cylinders stump?

Decision Tree Example

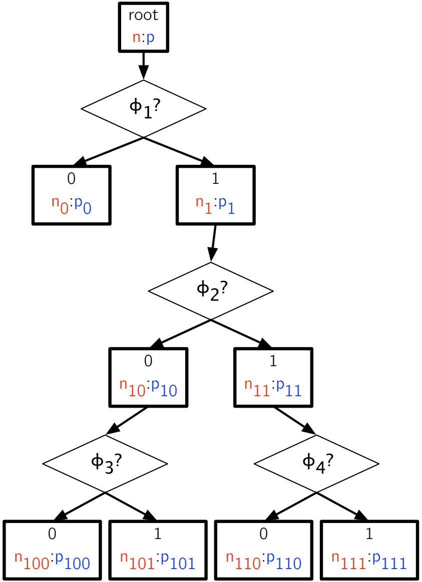


Error reduction compared to the cylinders stump?

Decision Tree: Making a Prediction



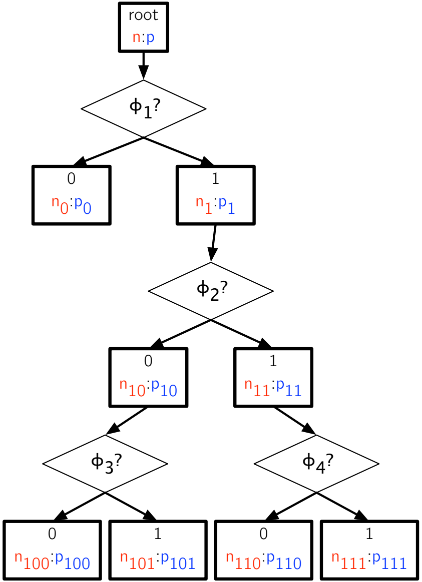
Decision Tree: Making a Prediction



Data: decision tree t, input example x
Result: predicted class
if t *has the form* $\textsc{Leaf}(y)$ **then**
    return y;
else
    $\#$ $t.\phi$ is the feature associated with t;
    $\#$ t.child(v) is the subtree for value v;
    return $\textsc{DTreeTest}(t.\text{child}(t.\phi(x)), x)$;
end

Algorithm 1: $\textsc{DTreeTest}$

Decision Tree: Making a Prediction



Equivalent boolean formulas:

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

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).

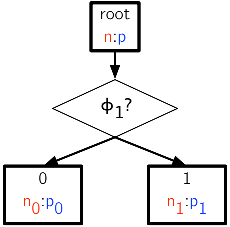
3^D formulas.

Growing a Decision Tree

root

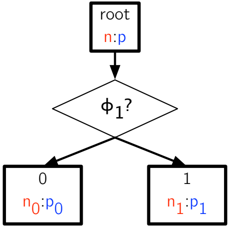
n:p

Growing a Decision Tree



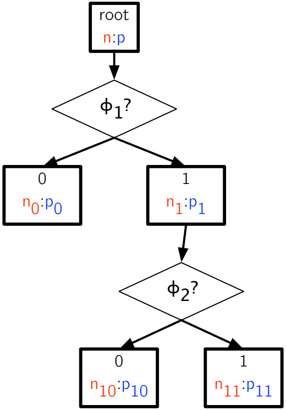
We chose feature ϕ_1. Note that $n = n_0 + n_1$ and $p = p_0 + p_1$.

Growing a Decision Tree

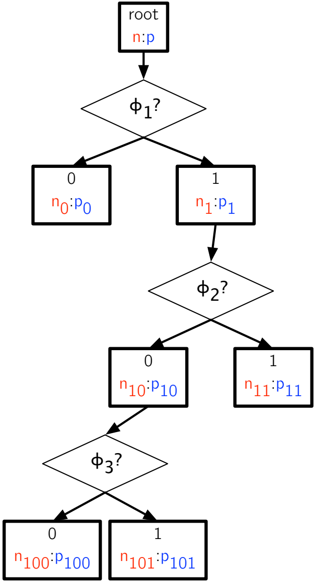


We chose not to split the left partition. Why not?

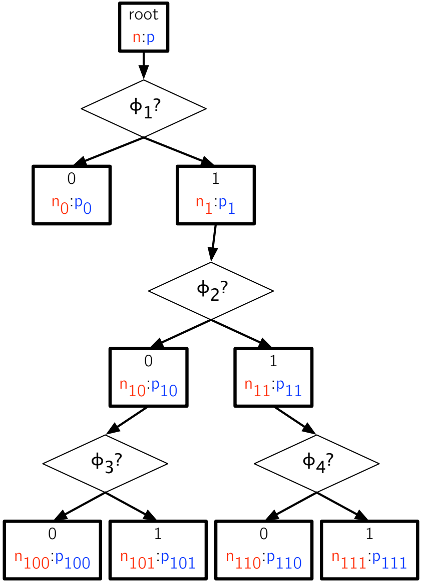
Growing a Decision Tree



Growing a Decision Tree



Growing a Decision Tree

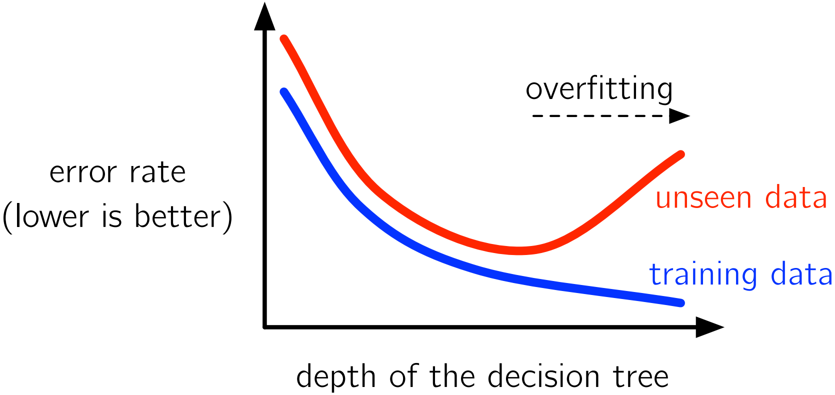


Greedily Building a Decision Tree (Binary Features)

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 $\textsc{Leaf}(y)$;
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 $\textsc{Node}(\phi^*, \{0 \rightarrow \textsc{DTreeTrain}(D_0, \Phi \setminus \{\phi^*\}), 1 \rightarrow \textsc{DTreeTrain}(D_1, \Phi \setminus \{\phi^*\})\})$;
end

Algorithm 2: $\textsc{DTreeTrain}$

Danger: Overfitting



Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Better yet, hold out two subsets, one for tuning and one for a true, honest-to-science **test**.

Detecting Overfitting

If you use all of your data to train, you won't be able to draw the red curve on the preceding slide!

Solution: hold some out. This data is called **development data**. More terms:

- Decision tree max depth is an example of a **hyperparameter**
- "I used my development data to **tune** the max-depth hyperparameter."

Better yet, hold out two subsets, one for tuning and one for a true, honest-to-science **test**.

Splitting your data into training/development/test requires careful thinking. Starting point: randomly shuffle examples with an 80%/10%/10% split.