Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 2

Slides adapted from Noah Smith

Administrivia

- Make sure that you enroll in Canvas and have access to Piazza
- Email me to introduce yourself, one of your core values, and a machine learning application you care about
- The link to lecture videos has been updated

Learning Objectives

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

Outline

Memorization vs. Generalization

Features

Decision tree

Outline

Memorization vs. Generalization

Features

Decision tree

Memorization vs. Generalization

What do you think are the differences?

Memorization vs. Generalization

Task: Given a dataset that contains transcripts at CU, predict whether a student is going to take CSCI 4622

Memorization vs. Generalization

Task: Given a dataset that contains transcripts at CU, predict whether a student is going to take CSCI 4622

- whether Michael is going to take this class?

Memorization vs. Generalization

Task: Given a dataset that contains transcripts at CU, predict whether a student is going to take CSCI 4622

- whether Michael is going to take this class?
- whether Bill Gates is going to take this class?

Memorization vs. Generalization

- training data
- test set

Memorization vs. Generalization

- training data
- test set

Formal definition in the next lecture

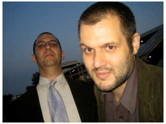
Outline

Memorization vs. Generalization

Features

Decision tree

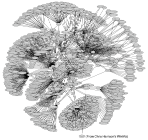
Features



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

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

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



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

Features

Let ϕ be a function that maps from inputs ($\boldsymbol{x}$) to values.

Features

Let ϕ be a function that maps from inputs ($\boldsymbol{x}$) 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 ($\boldsymbol{x}$) 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 ($\boldsymbol{x}$) 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

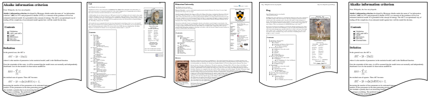
Let us have an interactive example to think through data representation!

Features

Let us have an interactive example to think through data representation!
Auto insurance quotes

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

Memorization vs. Generalization

Features

Decision tree

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

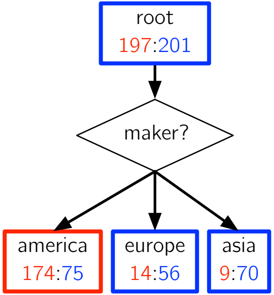
values of y		values of feature ϕ			
		v_1	v_2	$\cdots$	v_K
	0				
	1				

Decision Stump Example

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

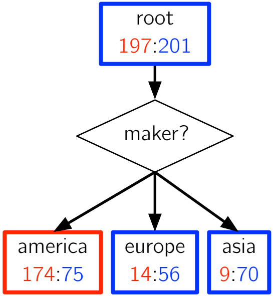
Decision Stump Example

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



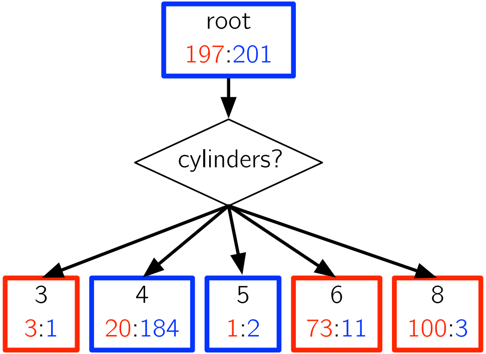
Decision Stump Example

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

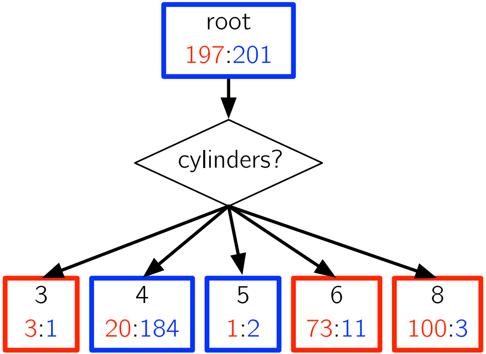


Errors: 75 + 14 + 9 = 98 (about 25%)

Decision Stump Example



Decision Stump Example



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

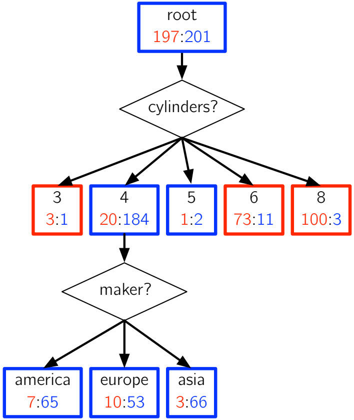
Key Idea: Recursion

A single feature **partitions** the data.

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

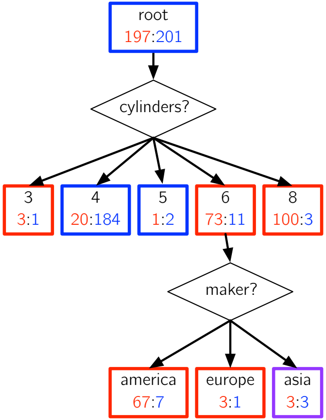
Applying this recursively, we can construct a **decision tree**.

Decision Tree Example



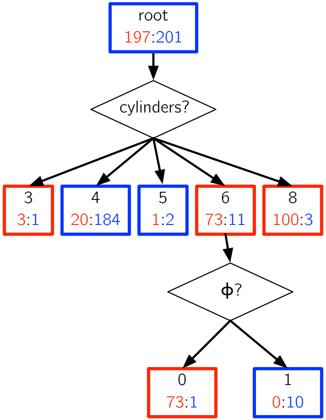
Error reduction compared to the cylinders stump?

Decision Tree Example



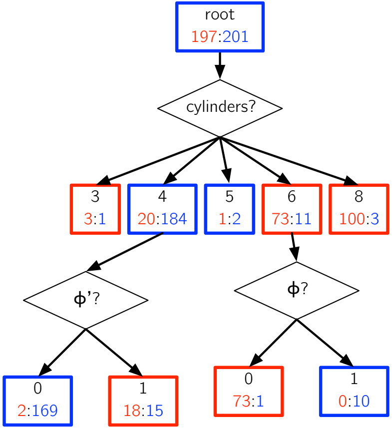
Error reduction compared to the cylinders stump?

Decision Tree Example



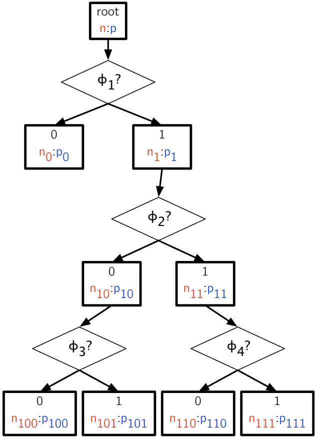
Error reduction compared to the cylinders stump?

Decision Tree Example

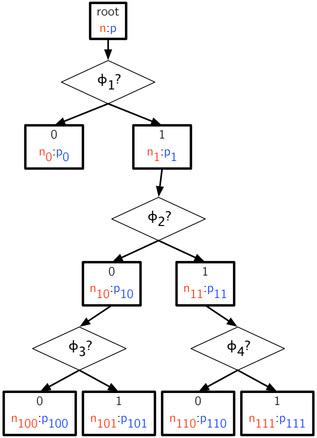


Error reduction compared to the cylinders stump?

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

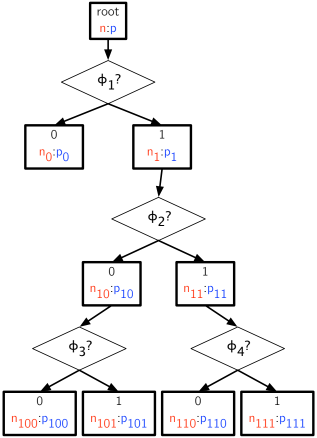
  # $t.\phi$ is the feature associated with t;

  # 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).

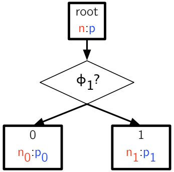
3^D formulas.

Growing a Decision Tree

root

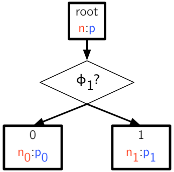
n:p

Growing a Decision Tree



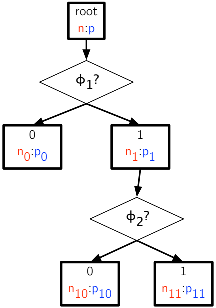
We chose feature ϕ_1. Note that $n = n_0 + n_1$ and $p = p_0 + p_1$.

Growing a Decision Tree

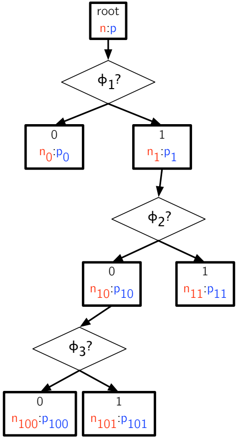


We chose not to split the left partition. Why not?

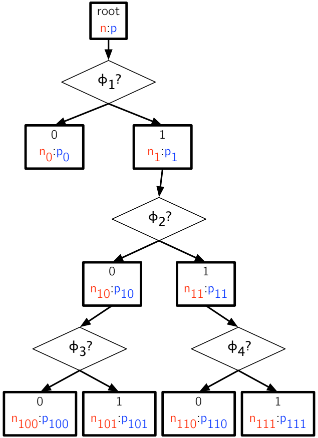
Growing a Decision Tree



Growing a Decision Tree



Growing a Decision Tree



Greedily Building a Decision Tree (Binary Features)

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?