Lecture 7: Decision Trees and Overfitting

COMP 343, Spring 2022

Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: These slides are based primarily on content from the book “Machine Learning” by Tom Mitchell, and on slides created by Vivek Srikumar (Utah), Dan Roth (Penn), and Jessica Wu (Harvey Mudd College)

Today's Topics

Homework 3 out

- Due Wednesday, February 23 by 5p

Learning decision trees (ID3 algorithm)

- Greedy heuristic (based on information gain)
  Originally developed for discrete features
- Some extensions to the basic algorithm

Avoiding overfitting

vumanfredi@wesleyan.edu

Decision Trees

RECAP

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**

   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$
      - If $S_v = \emptyset$: add leaf node with the common value of label in S

      Else: below this branch add the subtree ID3($S_v, A - \{A_b\}$)

   4. Return root node R

Set is empty

Set is not empty

Different branches may pick attributes in different orders!

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**

   1.
   2.
   3.

Can you implement this?

Everything is well defined, except for "best"

4. Return root node R

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Occam's razor:

Simpler explanations are better. Here, simpler explanations correspond to smaller trees

Why? Fewer short hypotheses than long ones

- A short hypothesis that fits the data is less likely to be a statistical coincidence
- Highly probable that a sufficiently complex hypothesis will fit the data

Entropy

Entropy (information, impurity, disorder, randomness) of a set of examples, S, with respect to binary classification is

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

$$Entropy(S) = H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

In general, for a discrete random variable with K possible values, with probabilities $\{p_1, p_2, \ldots, p_K\}$, the entropy is given by

$$H(\{p_1, p_2, \ldots, p_K\}) = -\sum_i^K p_i \log_2 p_i$$

Entropy is for a random variable: how much uncertainty is there in the values random variable takes on?

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

Over distribution of labels in data set S

Over distribution of labels just in subset S_v

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Weight by fraction of examples in subset

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

Over distribution of labels in data set S

Over distribution of labels just in subset S_v

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

If splitting on A has high gain, then removes the most uncertainty

Weight by fraction of examples in subset

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

Entropy before splitting - Entropy after splitting

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Tells us how important a given attribute of the feature vector is

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

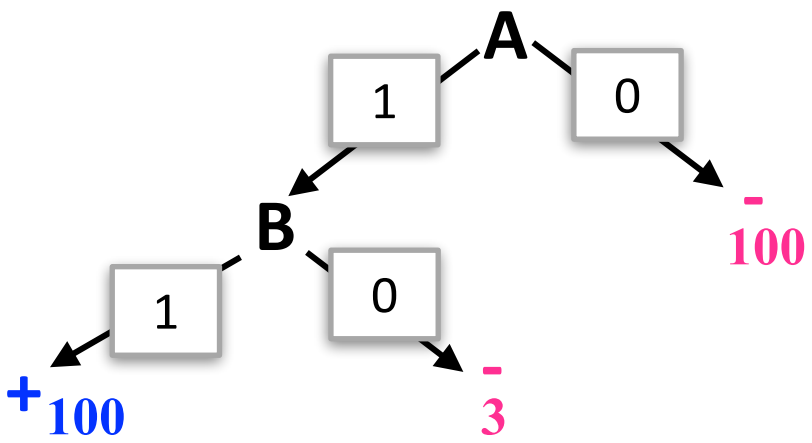
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on A:

- A=0 examples
- A=1 examples

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

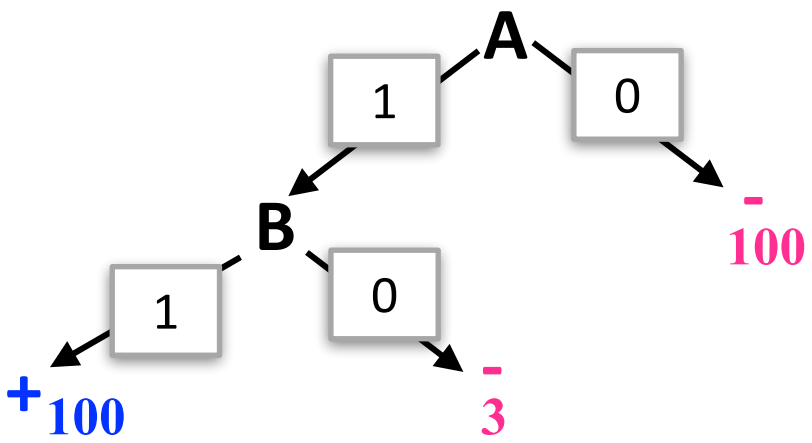
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on A:

- A=0 examples: all 100 have have same - label
- A=1 examples: all but 3 have same + label

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

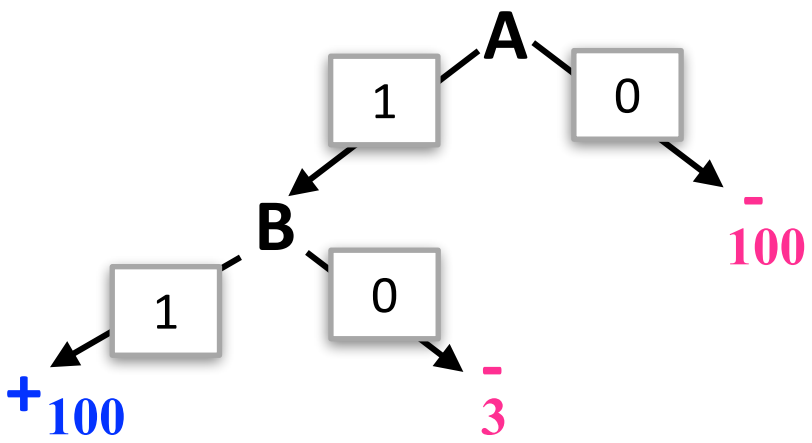
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on A:

- A=0 examples: all 100 have have same - label
- A=1 examples: all but 3 have same + label

Reduction in entropy is:

$$\frac{100}{203}H(100,0) + \frac{103}{203}H(100,3)$$

$$\approx \frac{100}{203} \times 0 + \frac{103}{203} \times 0 \approx 0$$

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

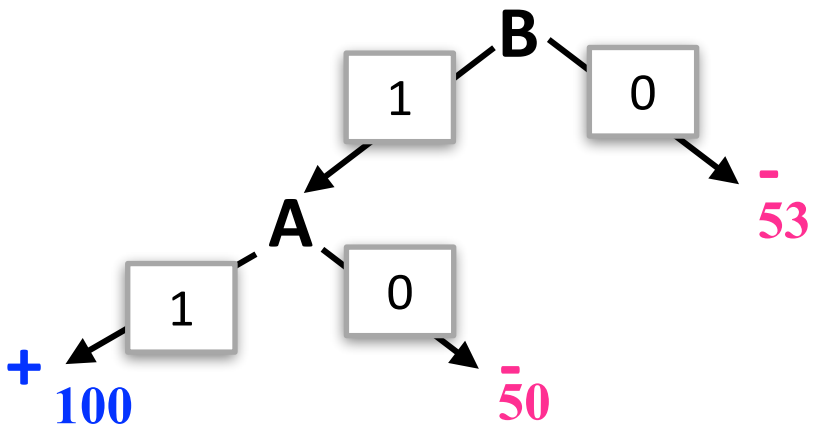
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on B:

- B=0 examples
- B=1 examples

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

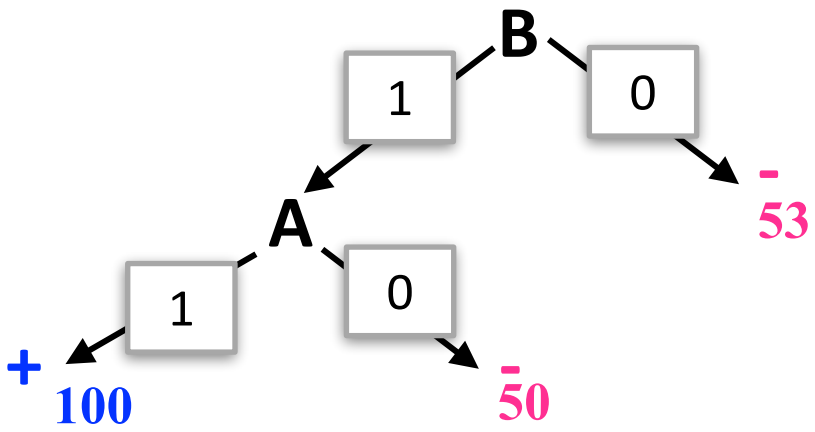
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on B:

- B=0 examples: 53 have - label, 0 have + label
- B=1 examples: 50 have - label, 100 have + label

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

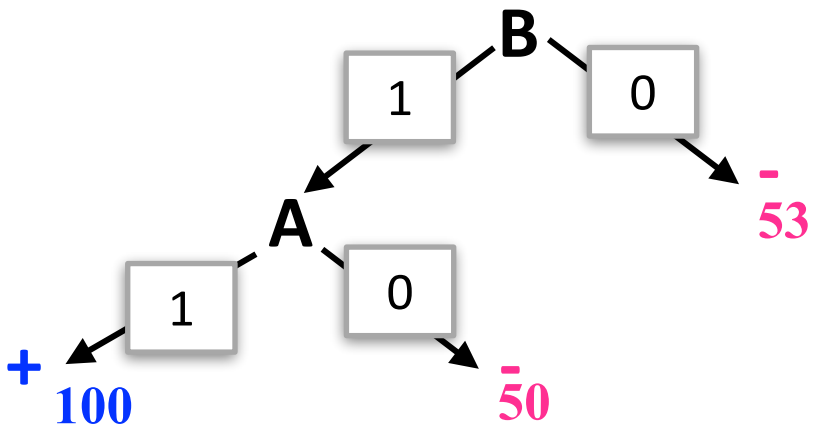
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Suppose we split on B:

- B=0 examples: 53 have - label, 0 have + label
- B=1 examples: 50 have - label, 100 have + label

Reduction in entropy is:

$$\frac{53}{203}H(53,0) + \frac{50}{203}H(50,100) > 0$$

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

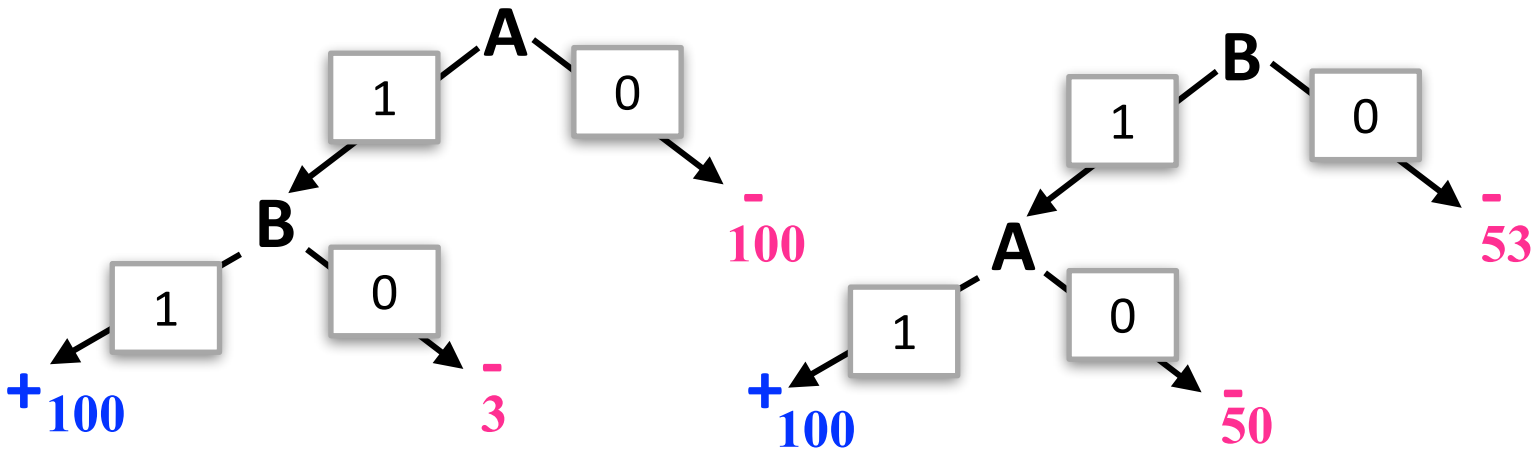
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

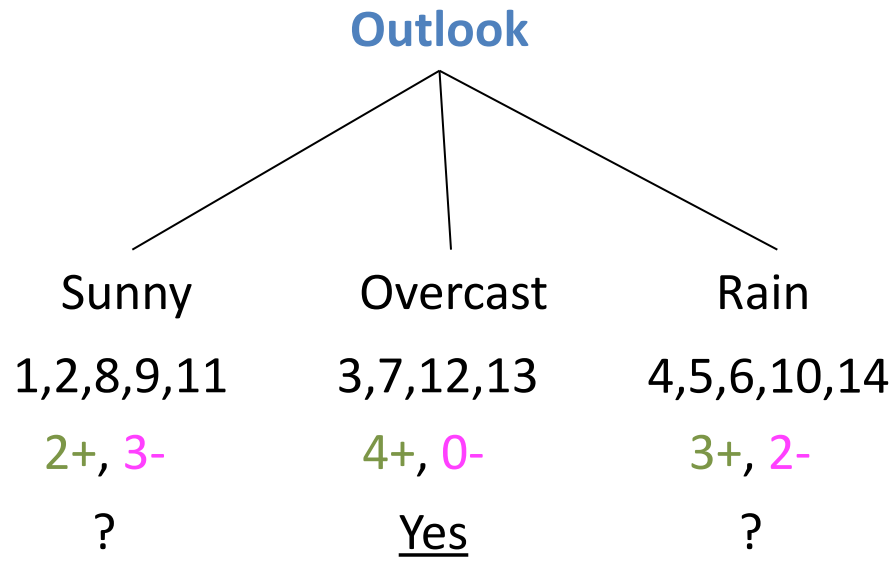
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Explains why A is split is better than B as split!

The tennis example again



	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Continue until either:

Every attribute is included in path

OR

All examples in the leaf have same label

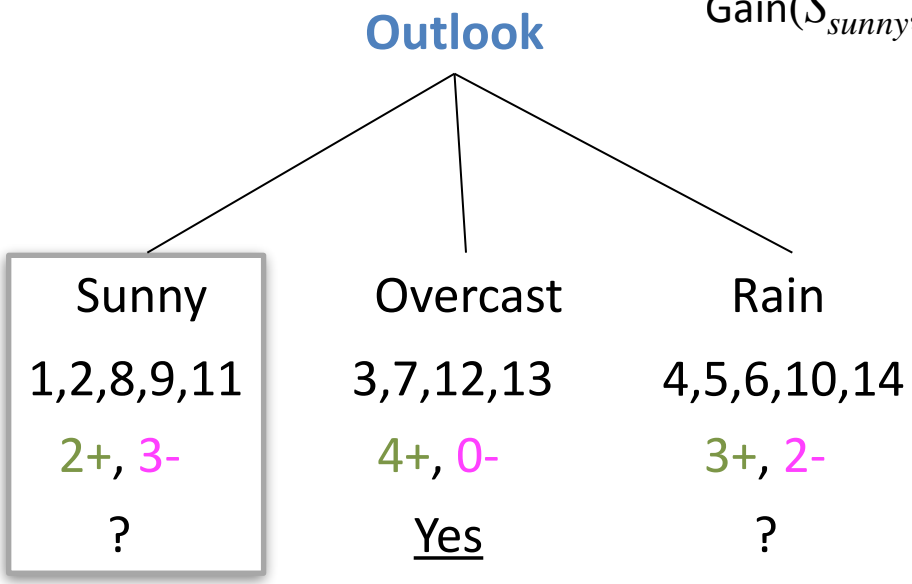
An illustrative example

Subset of data, S, for which Outlook is Sunny

Hot, Cool all same label

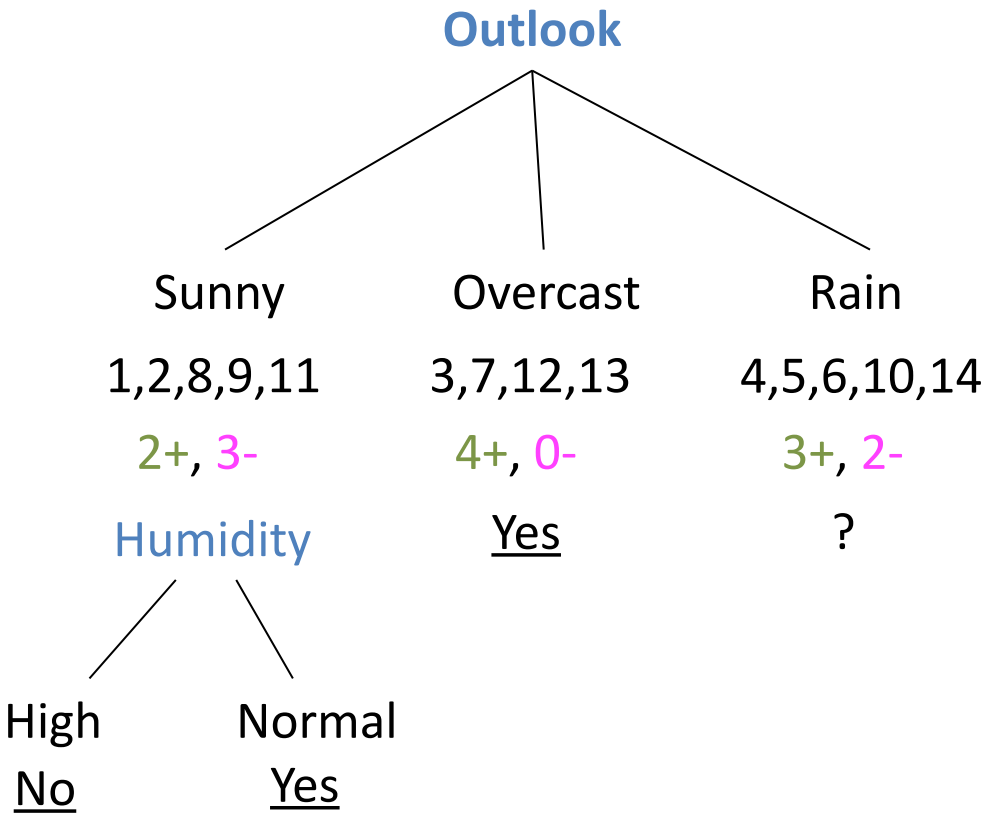
$$\text{Gain}(S_{sunny}, \text{Temp}) = .97 - 0 - (2/5)1 = .57$$
$$\text{Gain}(S_{sunny}, \text{Humidity}) = .97 - (3/5)0 - (2/5)0 = .97$$
$$\text{Gain}(S_{sunny}, \text{Wind}) = .97 - (2/5)1 - (3/5).92 = .02$$

Entropy over S

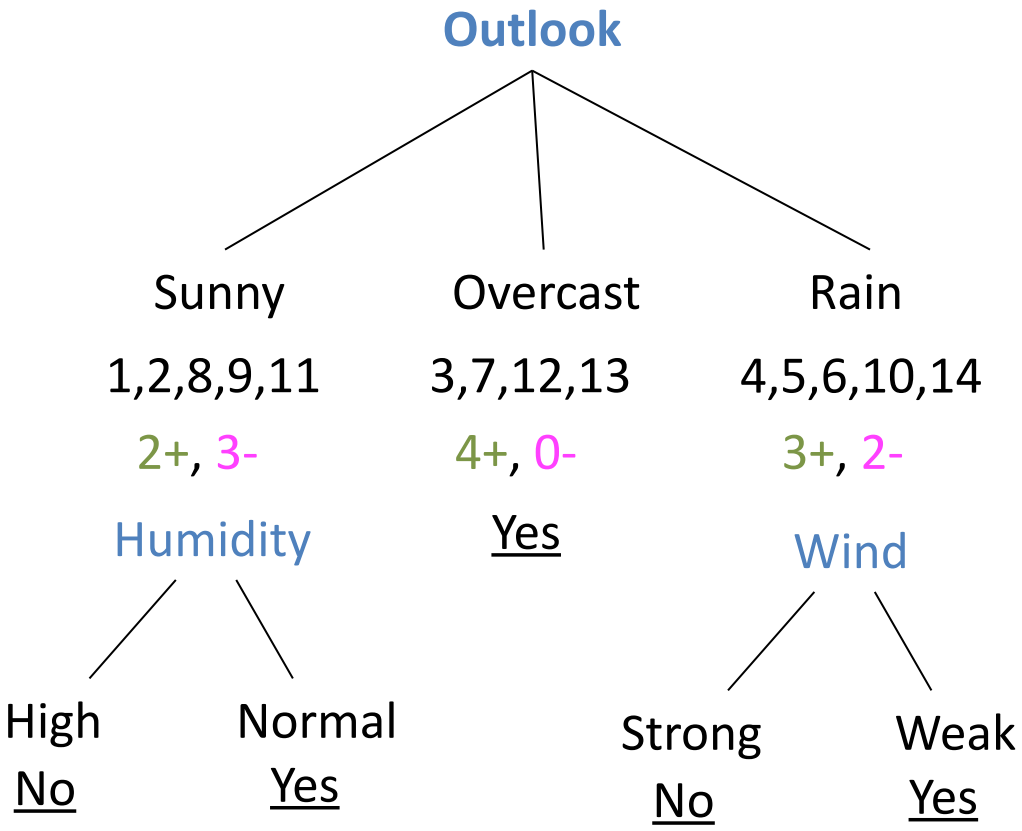


Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

An illustrative example



An illustrative example



Hypothesis space in decision tree induction

Search over decision trees, which can represent all possible discrete functions (has pros and cons)

Goal: to find the best decision tree

- Best could be “smallest depth”
- Best could be “minimizing the expected number of tests"

Finding a minimal decision tree consistent with a set of data is NP-hard

- Search over space of all possible decision trees → search over space of all possible boolean functions
- Very large space: with N boolean features then 2^N trees

How ID3 searches/learns

ID3 performs a greedy heuristic search (hill climbing without backtracking)

- Never go back and change root or parent to a different feature
- Heuristic search: no guarantee that smaller possible tree

Makes statistically based decisions using all data, i.e., ID3 is batch algorithm

- If you are given another example, you cannot incrementally change tree, you need to rerun ID3 from scratch

ID3 will find tree that represents your data

- If your data is noisy, learned tree will reflect that

Decision tree lets you visualize the learned model

- Most algorithms don't let you do this

Homework 3 discussion

Decision tree questions?

Python questions?

3 min: discussion

How are we evaluating decision tree performance?

3 min: discussion

How are we evaluating decision tree performance?

Information gain is heuristic to help us search through hypothesis space. But we ultimately care about test examples correctly labeled, or not.

Summary: learning decision trees

1. Representation: what are decision trees?
    - A hierarchical data structure that represents data

2. Algorithm: learning decision trees
    - The ID3 algorithm: a greedy heuristic
        - If all examples have the same label, create a leaf with that label
        - Otherwise, find the “most informative” attribute and split the data for different values of that attributes
        - Recurse on the splits

Summary: learning decision trees

1. Representation: what are decision trees?
    - A hierarchical data structure that represents data

2. Algorithm: learning decision trees
    - The ID3 algorithm: a greedy heuristic
        - If all examples have the same label, create a leaf with that label
        - Otherwise, find the "most informative" attribute and split the data for different values of that attributes
        - Recurse on the splits

3. Some extensions + overfitting

Decision Trees

TIPS AND TRICKS

Tips and tricks

1. Decision tree variants

2. Handling examples with missing feature values

3. Non-Boolean features

4. Avoiding overfitting

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

What if we have another way to quantify disorder?

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

What if we have another way to quantify disorder?

Can use that measure to pick best attribute instead

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

MajorityError

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

MajorityError

- Suppose the tree was not grown below this node and the most frequent label were chosen, what would be the error?

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

MajorityError

- Suppose the tree was not grown below this node and the most frequent label were chosen, what would be the error?
- Suppose at some node, there are 15+ and 5- examples. What is the MajorityError?

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

MajorityError

- Suppose the tree was not grown below this node and the most frequent label were chosen, what would be the error?
- Suppose at some node, there are 15+ and 5- examples. What is the MajorityError?
- Answer: 1/4

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

Gini Index (variation of entropy)

- how often is a randomly chosen element from the set of examples incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset?

$$Gini(S) = \sum_{i=0}^{|Labels|} p_i(1 - p_i) = 1 - \sum_{i=1}^{|Labels|} p_i^2$$

Probability of picking example with label i

Probability of ***not*** picking example with label i

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/impurity of the labels

There are other ways to measure disorder

Gini Index (variation of entropy)

- how often is a randomly chosen element from the set of examples incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset?

$$Gini(S) = \sum_{i=0}^{|Labels|} p_i(1 - p_i) = 1 - \sum_{i=1}^{|Labels|} p_i^2$$

If no mistakes, sum is 0

Split on feature with lowest Gini Index

Information gain vs. Gini index

Information gain

- favors smaller partitions with distinct values

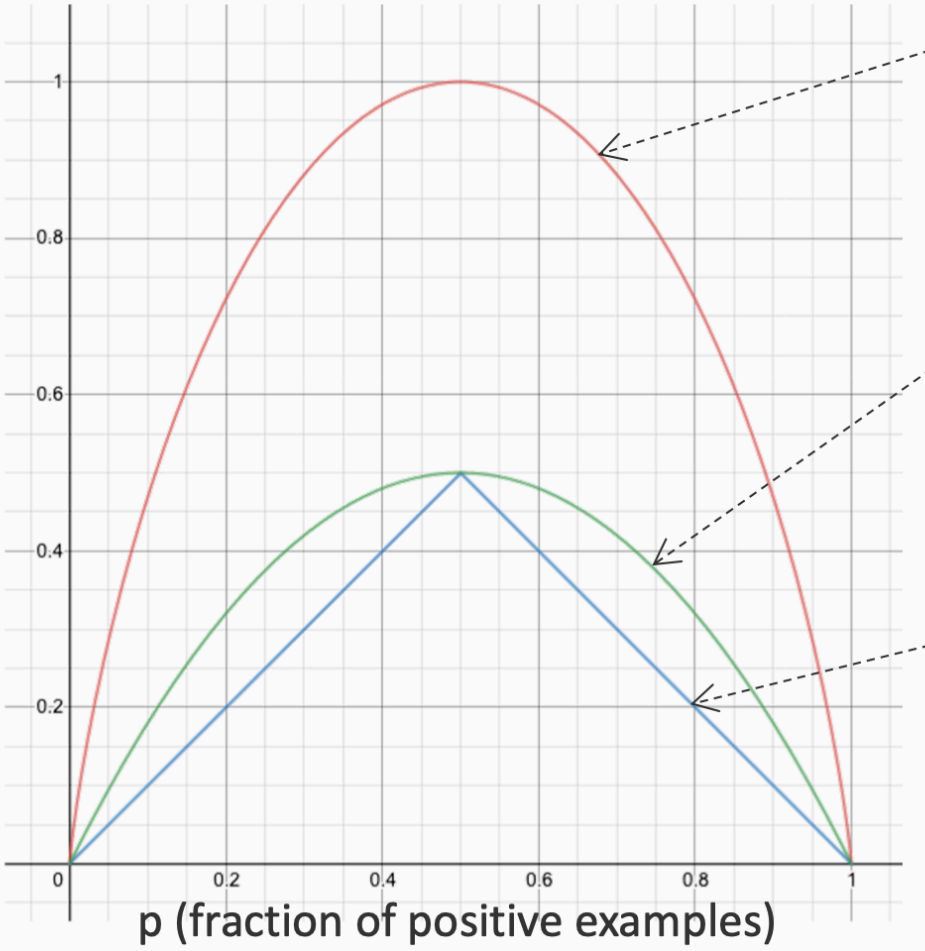
Gini Index

- favors larger partitions and easy to implement (no logarithms to compute)

For most data, results will be the same

1. Variants of information gain

Let p be the fraction of positive examples. Then $1 - p$ is the fraction of negative examples



Entropy:

$$-(p \log_2 p + (1-p) \log_2 (1-p))$$

Gini Index:

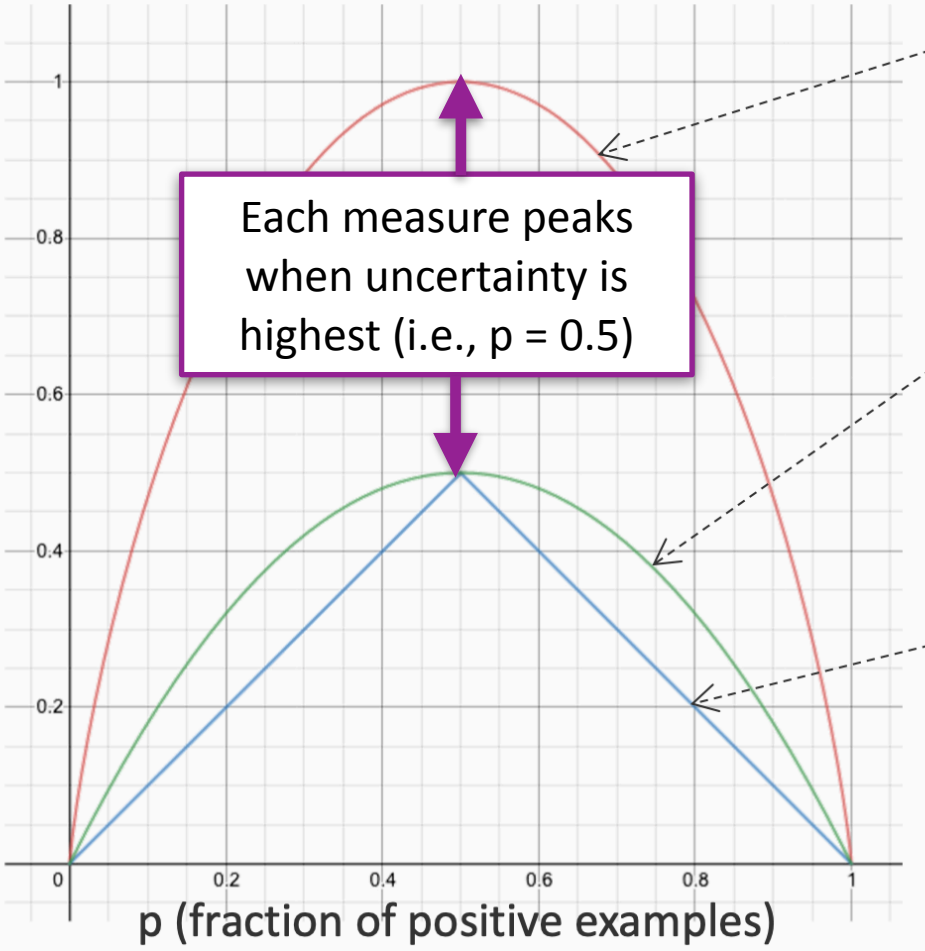
$$1 - (p^2 + (1-p)^2)$$

MajorityError:

$$\min(p, 1-p)$$

1. Variants of information gain

Let p be the fraction of positive examples. Then $1 - p$ is the fraction of negative examples



Entropy:

$$-(p \log_2 p + (1-p) \log_2 (1-p))$$

Gini Index:

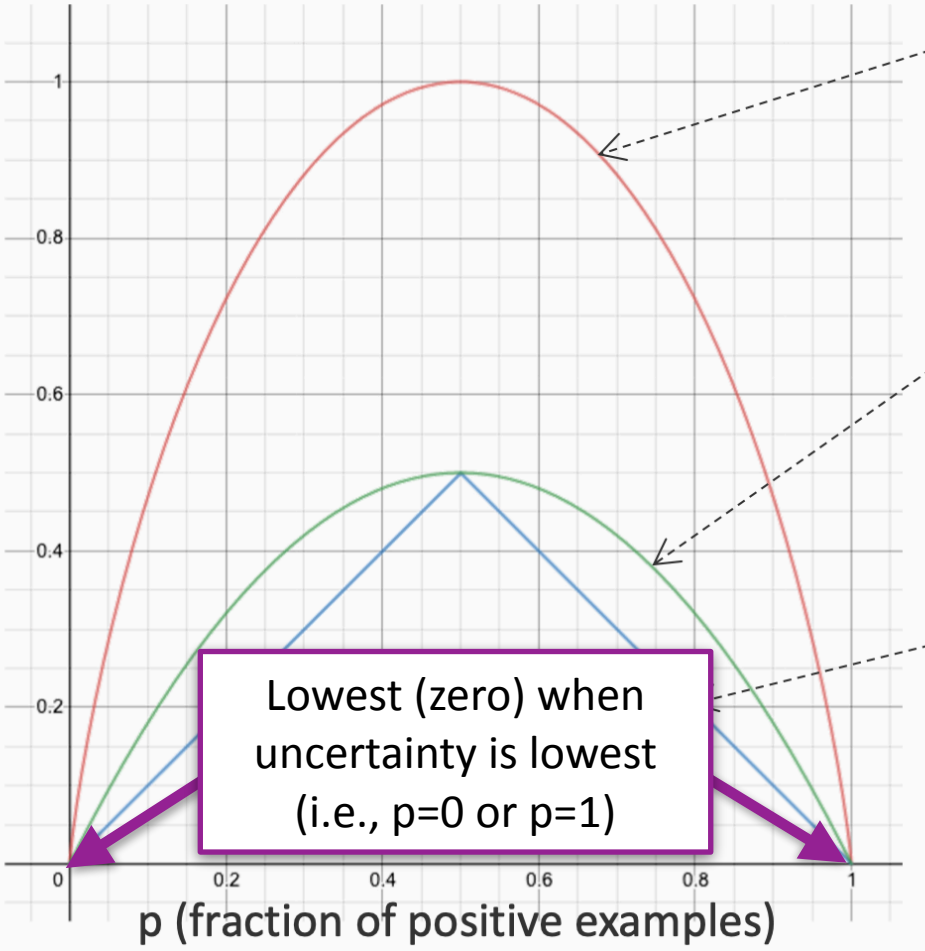
$$1 - (p^2 + (1-p)^2)$$

MajorityError:

$$\min(p, 1-p)$$

1. Variants of information gain

Let p be the fraction of positive examples. Then $1 - p$ is the fraction of negative examples



Entropy:

$$-(p \log_2 p + (1-p) \log_2 (1-p))$$

Gini Index:

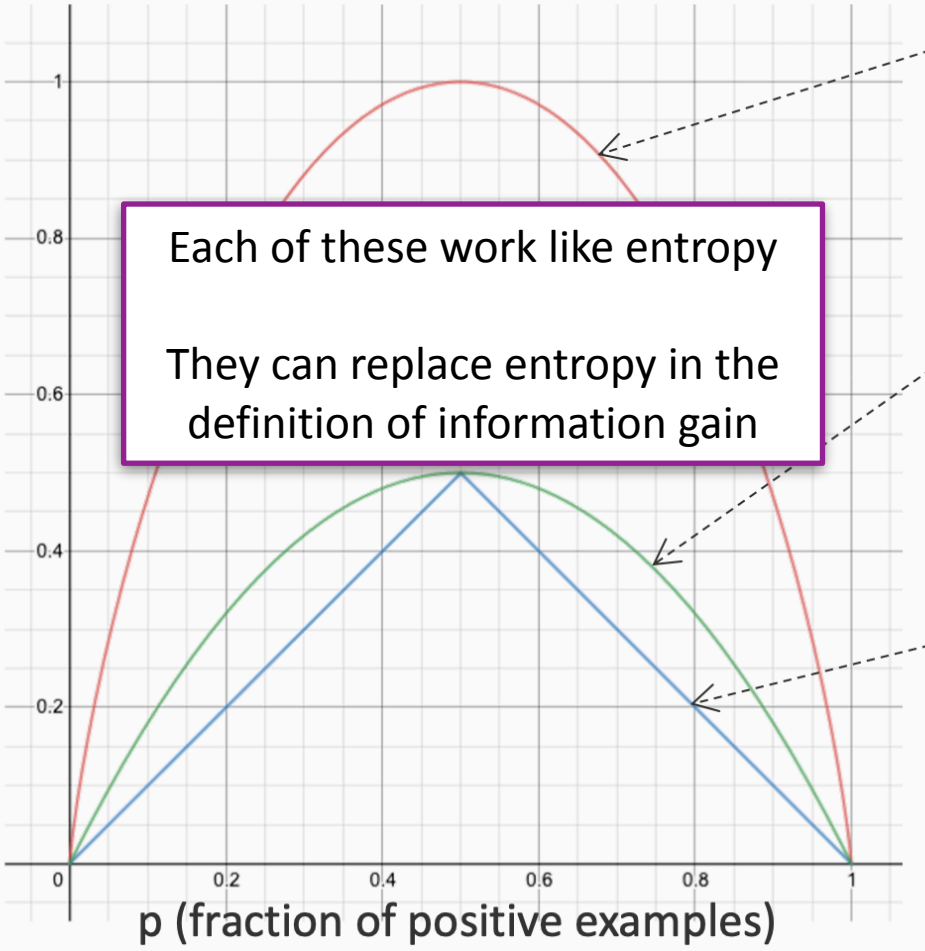
$$1 - (p^2 + (1-p)^2)$$

MajorityError:

$$\min(p, 1-p)$$

1. Variants of information gain

Let p be the fraction of positive examples. Then $1 - p$ is the fraction of negative examples



Each of these work like entropy

They can replace entropy in the definition of information gain

Entropy:

$$-(p \log_2 p + (1-p) \log_2 (1-p))$$

Gini Index:

$$1 - (p^2 + (1-p)^2)$$

MajorityError:

$$\min(p, 1-p)$$

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at training time?

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at training time?

Option 1: Ignore the problem!

Downside?

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at training time?

Option 1: Ignore the problem!

Downside? Data is hard to get

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	??? High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at training time?

Option 2: Complete the example

Use most common value of attribute in data, or use most common value of attribute among all examples with same output

Downside?

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	??? High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at training time?

Option 2: Complete the example

Use most common value of attribute in data, or use most common value of attribute among all examples with same output

Downside? We don't care about predicting the value, we care about predicting the label

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

e.g., Outlook = {5/14 Sunny, 4/14 Overcast, 5/14 Rain}

What can we do at training time?

Option 3: Use fractional counts of attribute values

Downside? Need to update entropy calculation to handle fractional counts, makes things more complicated

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at test time if attribute is missing value?

2. Missing feature values

Suppose an example is missing the value of an attribute.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

What can we do at test time if attribute is missing value?

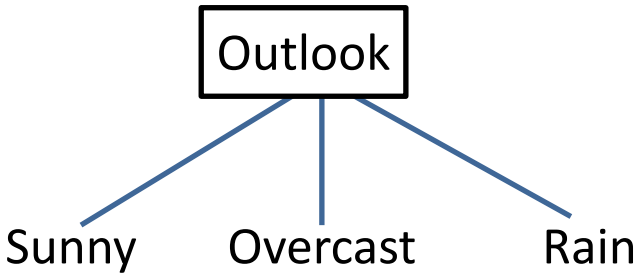
Use the same method: choose most common label or any of the other options

3. Non-Boolean feature values

Suppose features can take multiple values …

We have seen one edge per split

- i.e., a multi-way split



3. Non-Boolean feature values

Suppose features can take multiple values …

Another option

- Make the attributes Boolean by testing for each value
- E.g., outlook is sunny can be expressed as combination of 3 booleans

Convert Outlook=Sunny →

{
Outlook:Sunny=True
Outlook:Overcast=False
Outlook:Rain=False
}

3. Non-Boolean feature values

Suppose features can take multiple values …

Another option

- Make the attributes Boolean by testing for each value
- E.g., outlook is sunny can be expressed as combination of 3 booleans

Convert Outlook=Sunny → {
Outlook:Sunny=True
Outlook:Overcast=False
Outlook:Rain=False
}

Allows reuse of binary tree code, for example

3. Non-Boolean feature values

Suppose features can take multiple values …

Another option

- Make the attributes Boolean by testing for each value
- E.g., outlook is sunny can be expressed as combination of 3 booleans

Convert Outlook=Sunny → {
Outlook:Sunny=True
Outlook:Overcast=False
Outlook:Rain=False
}

For numeric features, use thresholds or ranges to get Boolean/ discrete alternatives

Decision Trees

OVERFITTING

Overfitting

Goal for today

- What is overfitting?
- How we can combat overfitting for decision trees?

Keep in mind

- Overfitting is an issue not just to decision trees but pretty much every single machine learning algorithm

The “First Bit” function

A Boolean function with n inputs

Simply returns the value of the first input, all others irrelevant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

If $n = 2$: returns value of X_0

If $n = 100$: returns value of X_0

$\vdots$

$$Y = X_0$$

X_1 is irrelevant

The "First Bit" function

A Boolean function with n inputs

Simply returns the value of the first input, all others irrelevant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?

$$Y = X_0$$

X_1 is irrelevant

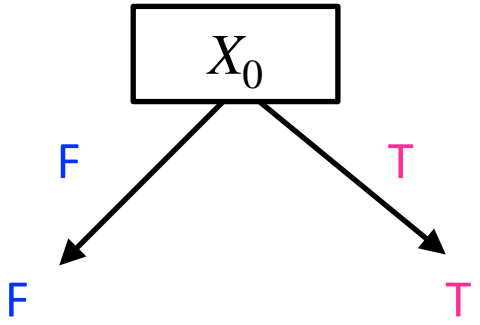
The "First Bit" function

A Boolean function with n inputs

Simply returns the value of the first input, all others irrelevant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?



$$Y = X_0$$

X_1 is irrelevant

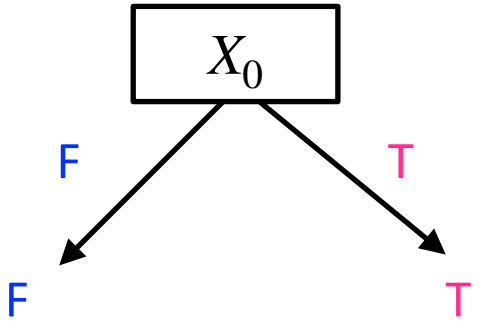
The "First Bit" function

A Boolean function with n inputs

Simply returns the value of the first input, all others irrelevant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?



$$Y = X_0$$

X_1 is irrelevant

Convince yourself that ID3 will generate this tree

The best case scenario: perfect data

What if instead of 2 bits we have n bits

Suppose we have all 2^n examples for training. What will the error be on any future examples?

The best case scenario: perfect data

What if instead of 2 bits we have n bits

Suppose we have all 2^n examples for training. What will the error be on any future examples?

Zero! Because we have seen every possible input!

The best case scenario: perfect data

What if instead of 2 bits we have n bits

Suppose we have all 2^n examples for training. What will the error be on any future examples?

Zero! Because we have seen every possible input!

And the decision tree can represent the function and ID3 will build a tree consistent with training set, regardless of noise in the data

Whatever label your training set assigns to example will be label that ID3 assigns

The best case scenario: perfect data

What if instead of 2 bits we have n bits

Suppose we have all 2^n examples for training. What will the error be on any future examples?

Zero! Because we have seen every possible input!

And the decision tree can represent the function and ID3 will build a tree consistent with training set, regardless of noise in the data

Perfect data: training error is zero

Future generalization error is also zero: because you've seen all of the examples because the data is perfect

Problem

What if data is noisy? After all, nature is noisy!

Error when collecting data (e.g., faulty sensor)

Error labeling data (e.g., label as cat but not a cat)

Error introduced later (e.g., spreadsheet messes up the data)

Only one way of being right but infinite ways of being wrong

Noisy data

What if the data is noisy? And we have all 2^n examples

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

Noisy data

What if the data is noisy? And we have all 2^n examples

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	F T
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T F
T	T	F	T
T	T	T	T

Suppose, the outputs of both training and test sets are randomly corrupted

Train and test sets are no longer identical

Both have noise, possibly different

Noisy data

What if the data is noisy? And we have all 2^n examples

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	F T
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T F
T	T	F	T
T	T	T	T

Suppose, the outputs of both training and test sets are randomly corrupted

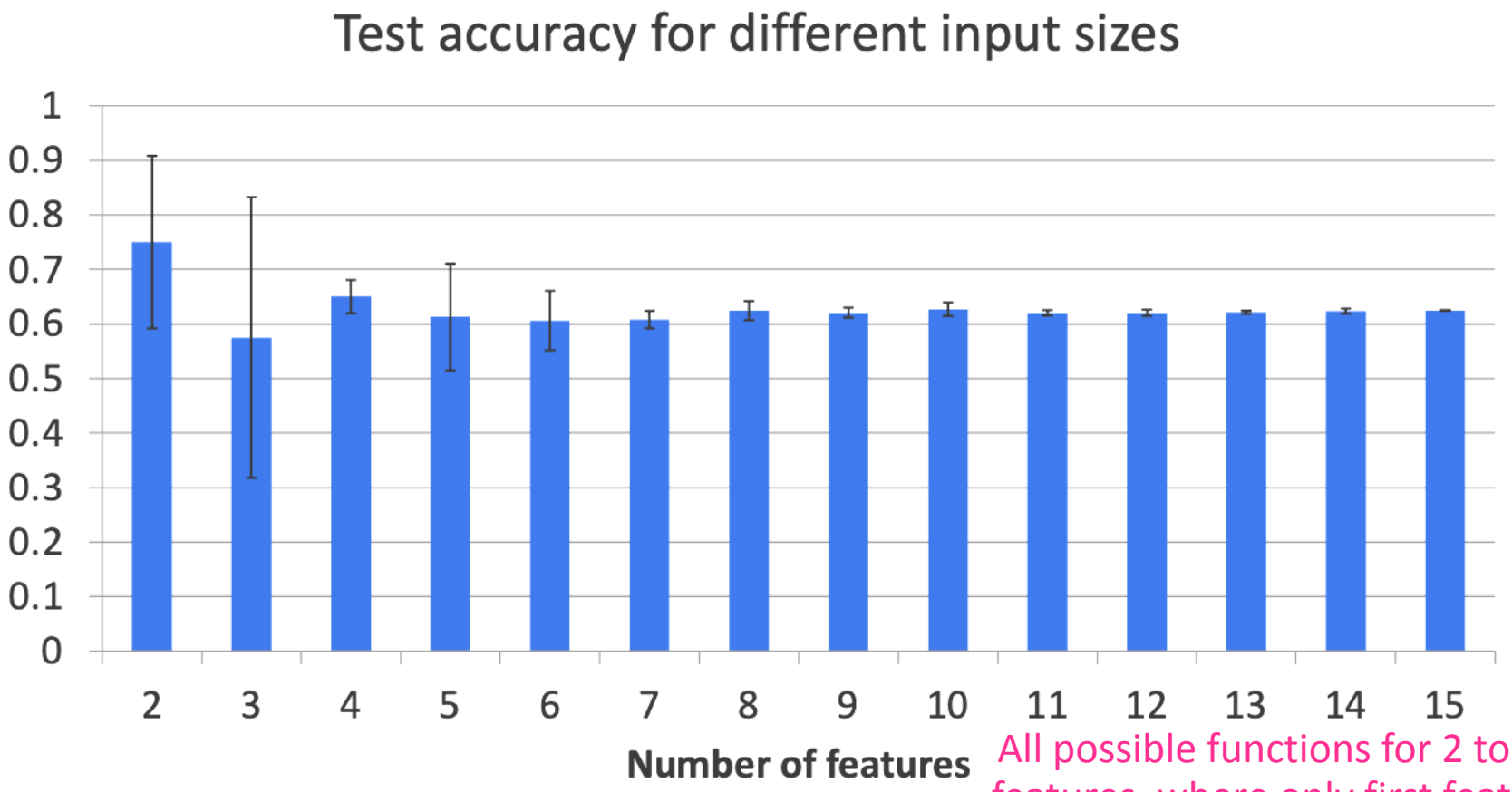
Train and test sets are no longer identical

Both have noise, possibly different

Prediction accuracy drops because there is noise!

Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples

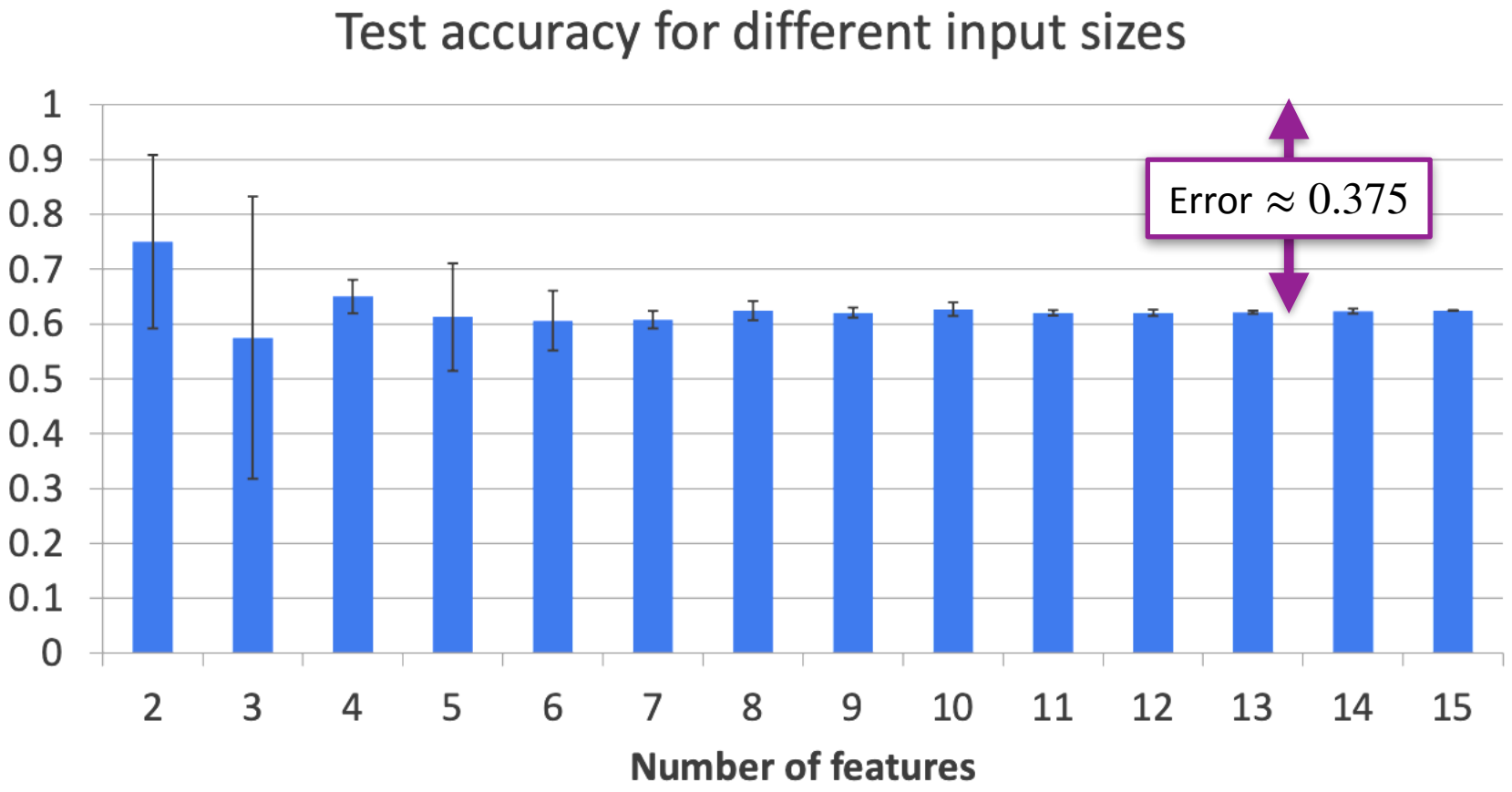


All possible functions for 2 to 15 features, where only first feature is relevant (i.e., indicates label)

Error bars are generated by running the same experiment multiple times for the same setting

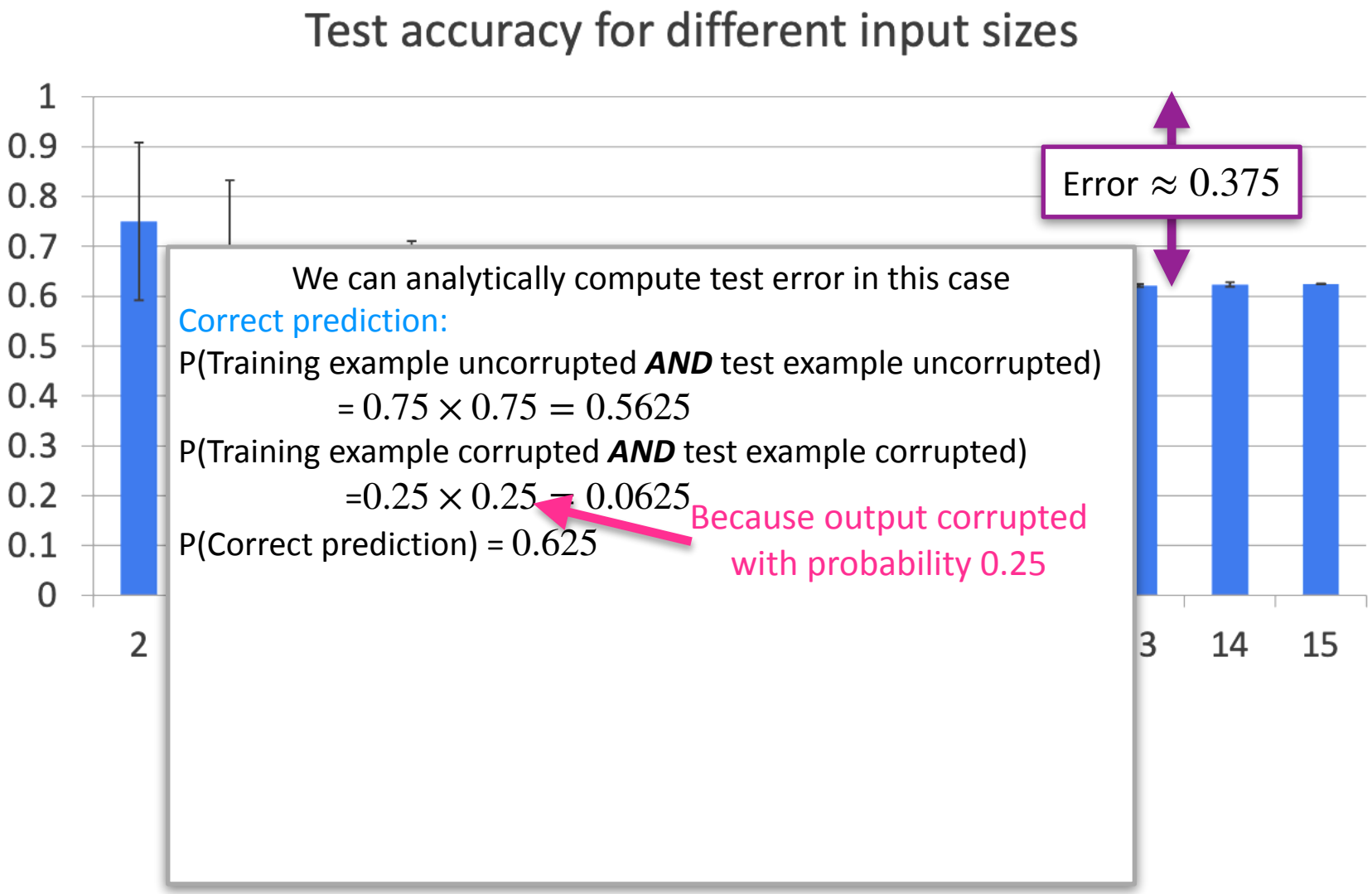
Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



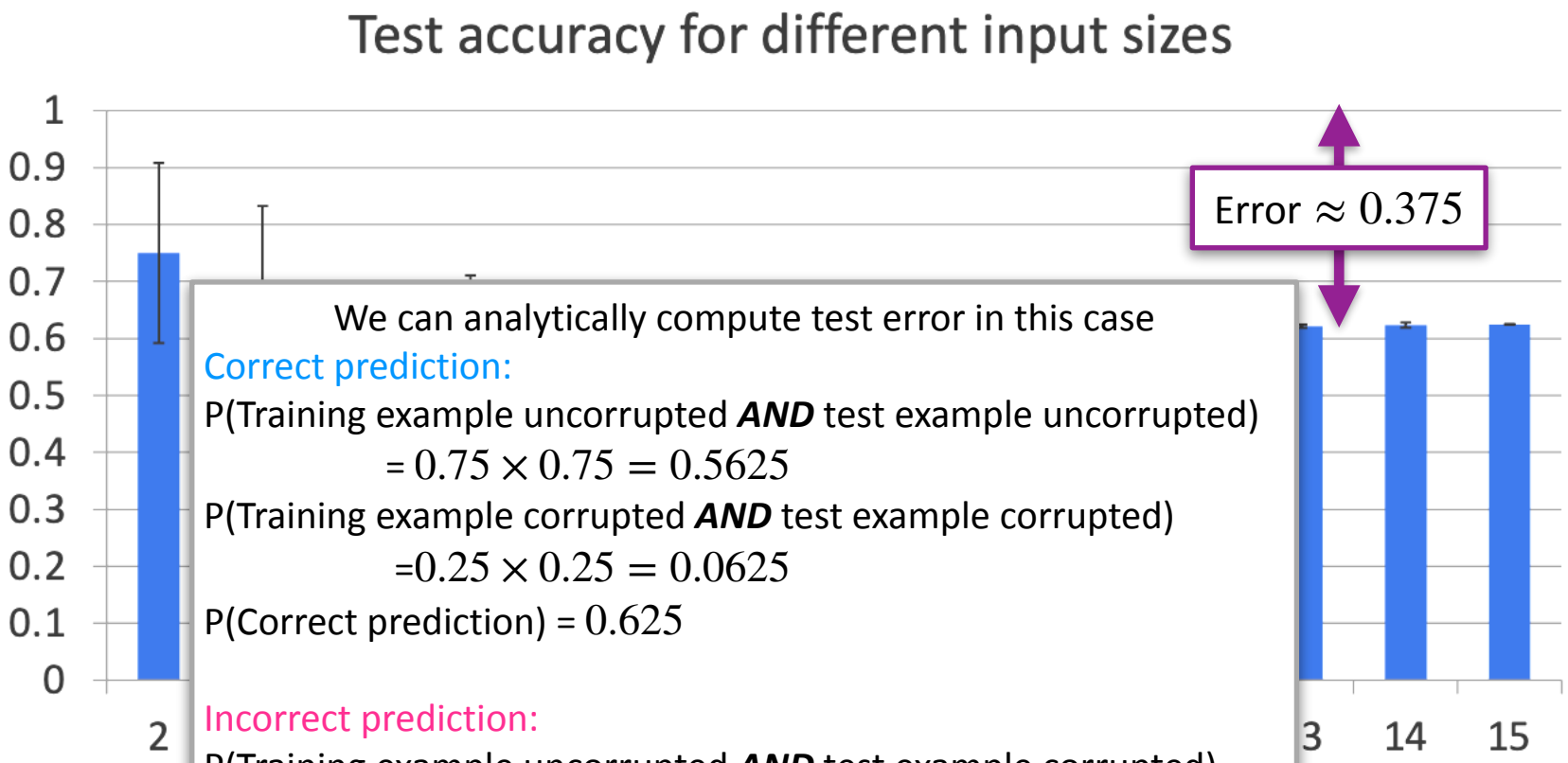
Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



We can analytically compute test error in this case

Correct prediction:

P(Training example uncorrupted ***AND*** test example uncorrupted)

$= 0.75 \times 0.75 = 0.5625$

P(Training example corrupted ***AND*** test example corrupted)

$= 0.25 \times 0.25 = 0.0625$

P(Correct prediction) = 0.625

Incorrect prediction:

P(Training example uncorrupted ***AND*** test example corrupted)

$= 0.75 \times 0.25 = 0.1875$

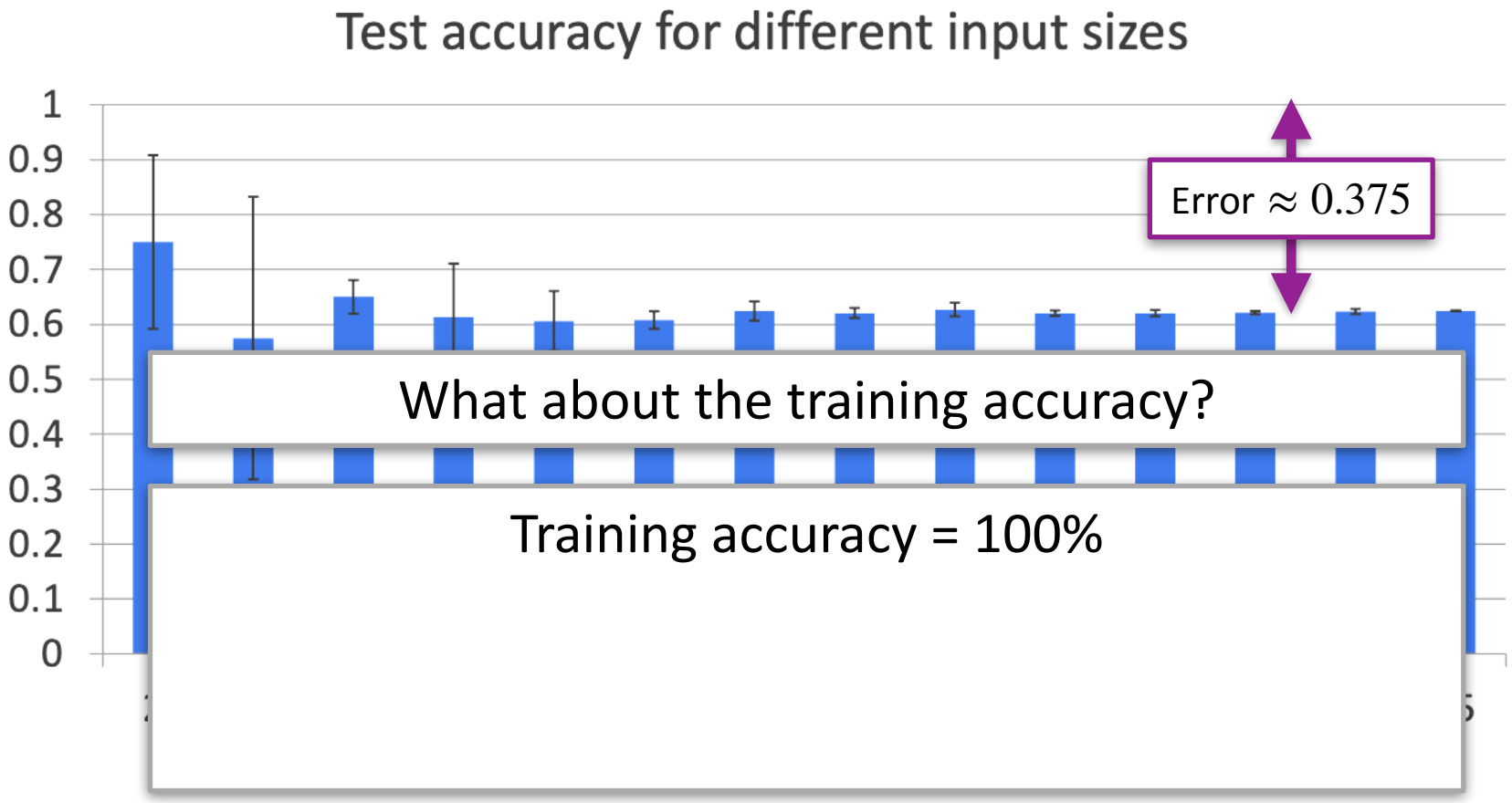
P(Training example corrupted ***AND*** test example uncorrupted)

$= 0.25 \times 0.75 = 0.1875$

P(Incorrect prediction) = 0.375

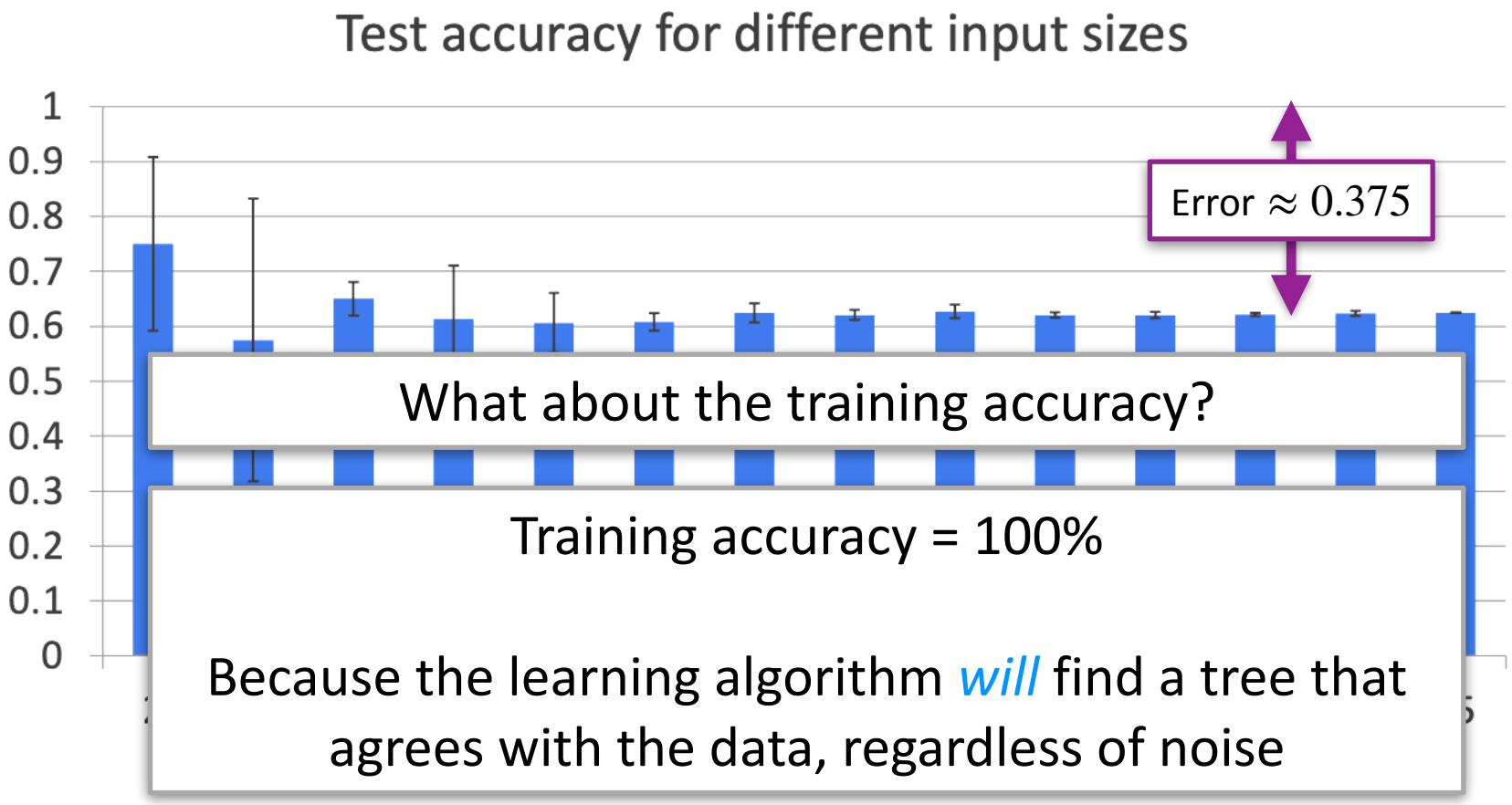
Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



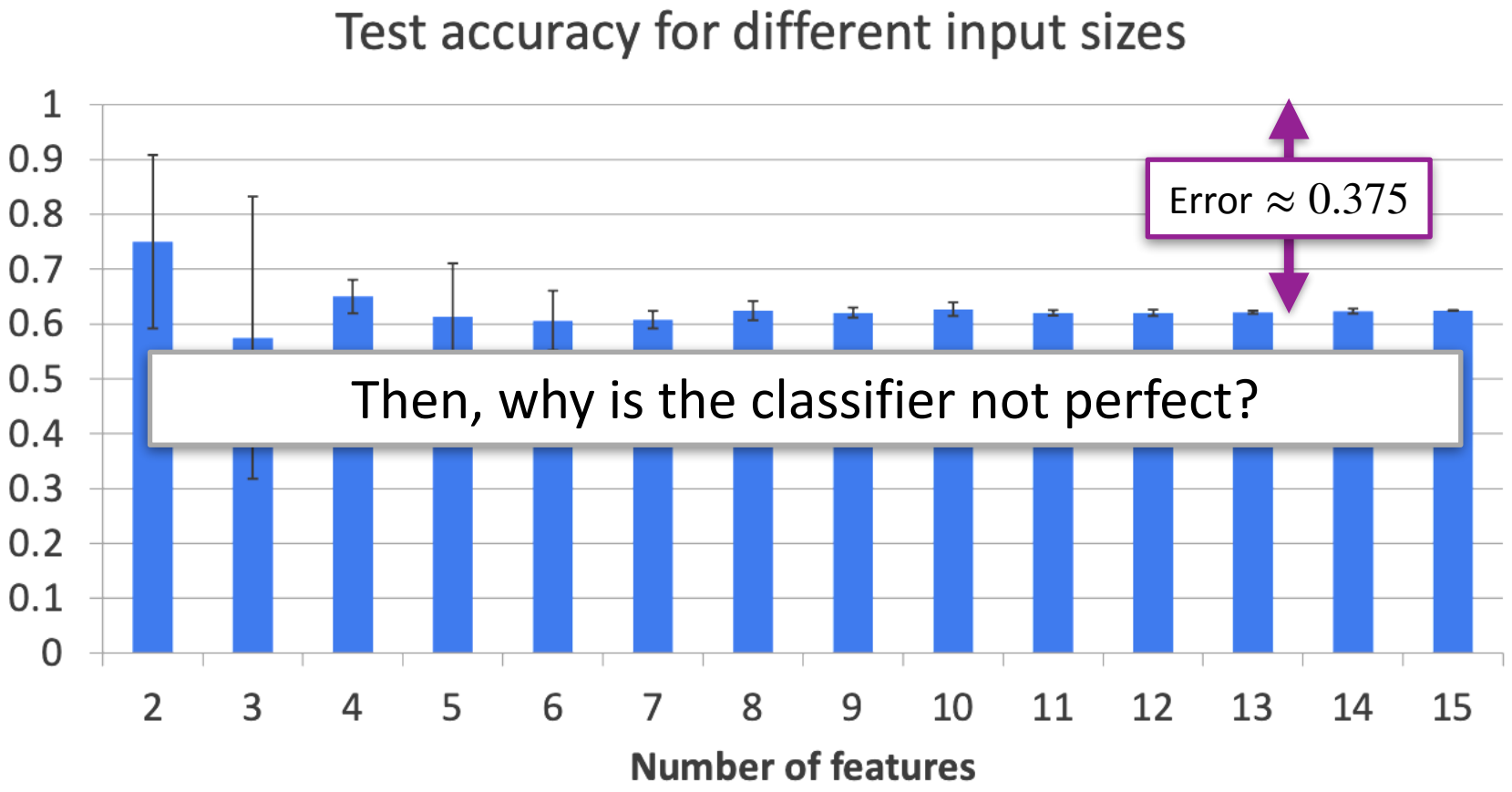
What about the training accuracy?

Training accuracy = 100%

Because the learning algorithm *will* find a tree that agrees with the data, regardless of noise

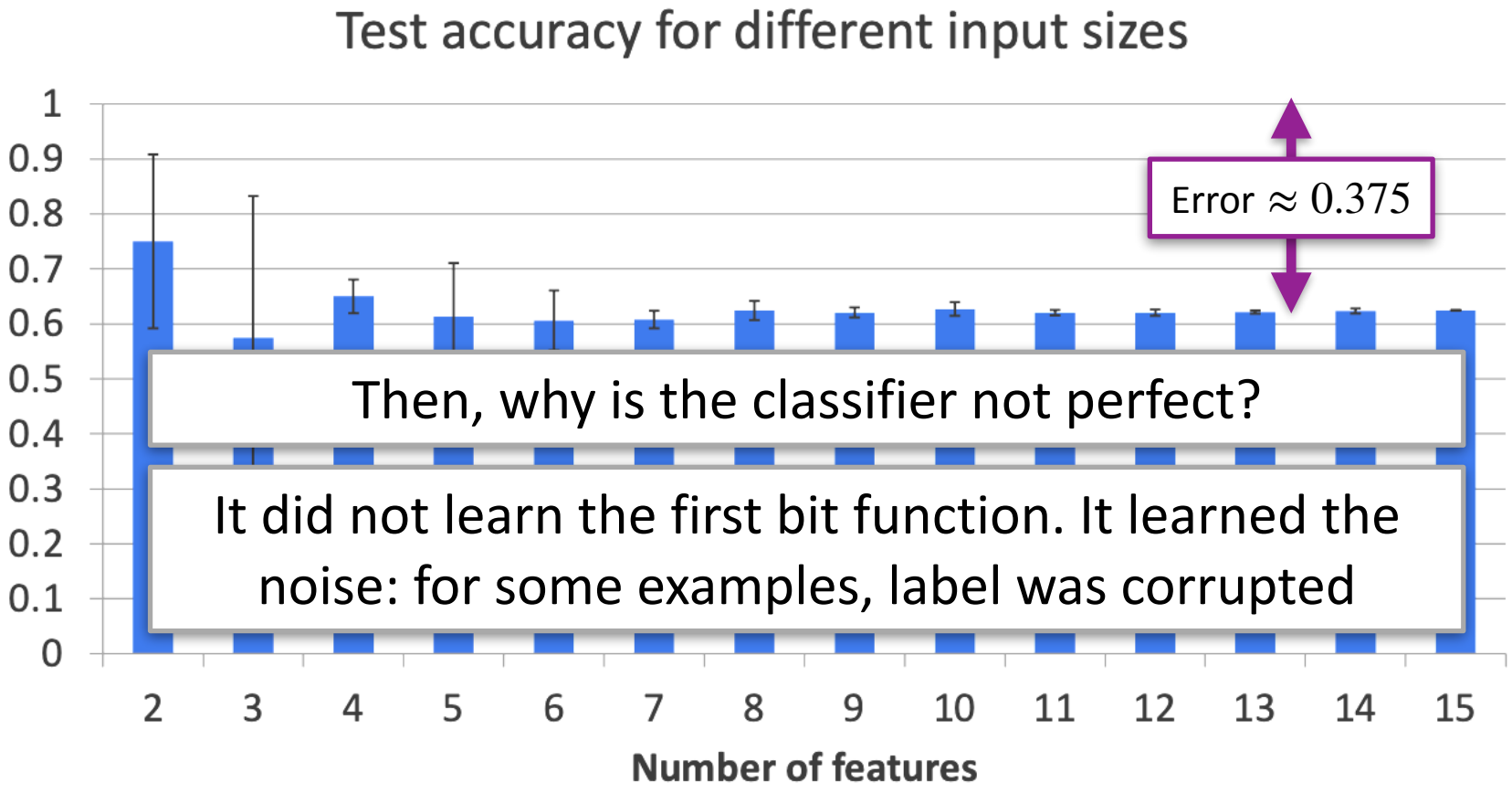
Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



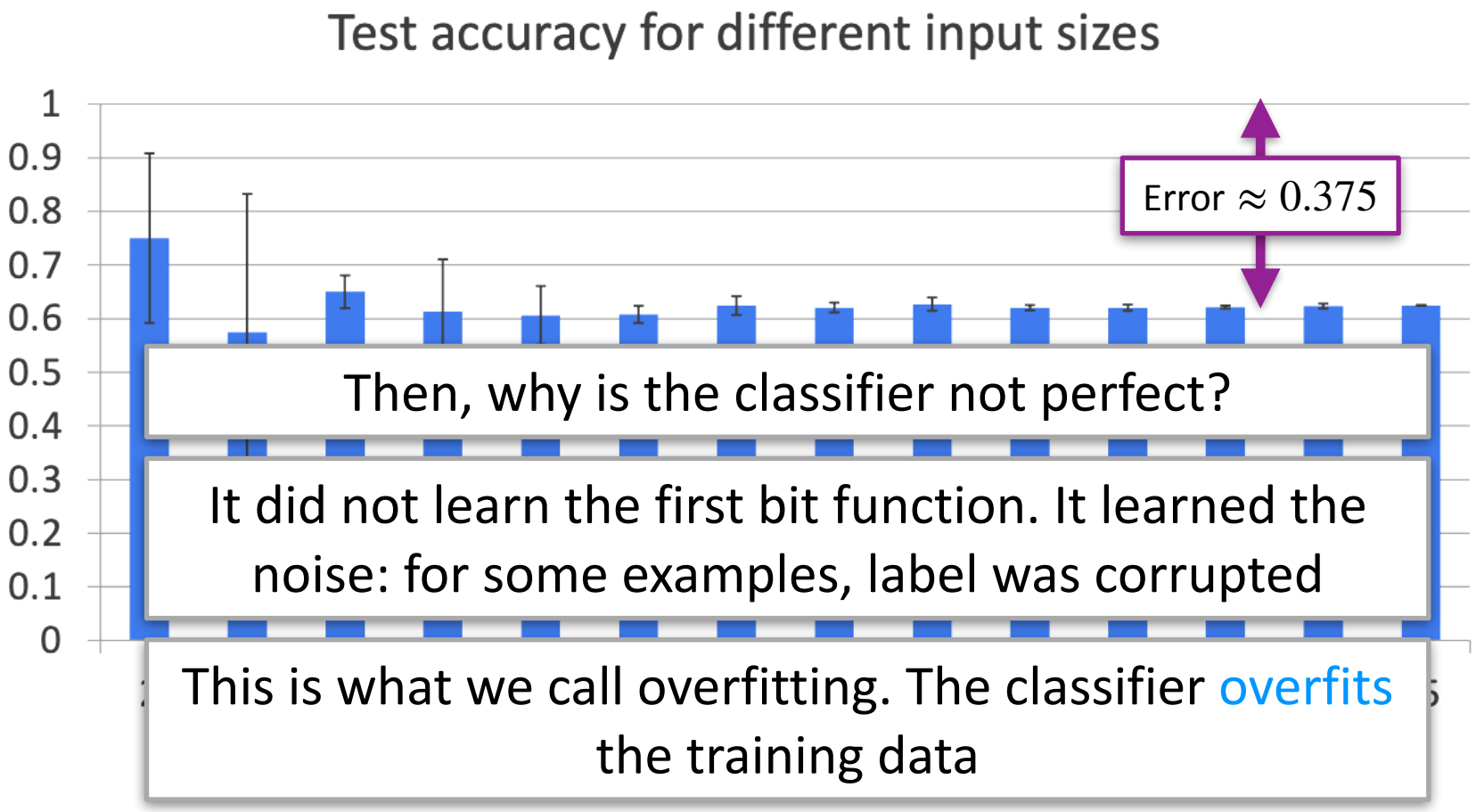
Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



Overfitting

You can think of overfitting as when the learning algorithm finds a hypothesis that fits the noise in the data

- Irrelevant attributes or noisy examples influence the choice of the hypothesis

Overfitting

You can think of overfitting as when the learning algorithm finds a hypothesis that fits the noise in the data

- Irrelevant attributes or noisy examples influence the choice of the hypothesis

Why is this bad?

- May lead to poor performance on future examples

Every learning algorithm needs to have a way of combatting overfitting

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$. Suppose we are using a hypothesis space H

Two kinds of errors:

Training error for hypothesis $h \in H$

True error for $h \in H$

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$. Suppose we are using a hypothesis space H

Two kinds of errors:

Training error for hypothesis $h \in H$

True error for $h \in H$

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$. Suppose we are using a hypothesis space H

Two kinds of errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

- Fraction of training examples on which hypothesis h makes a mistake

True error for $h \in H$

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$. Suppose we are using a hypothesis space H

Two kinds of errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

- Fraction of training examples on which hypothesis h makes a mistake

True error for $h \in H$: $error_D(h)$

- Expected error that hypothesis h makes on entire set of examples that exist … even examples we have not seen
- Mathematical concept: we cannot calculate

One definition of overfitting

Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_D(h)$

A hypothesis h overfits the training data if there is another hypothesis h' such that

1. h has lower training error than the competing hypothesis h' but
   - $error_{train}(h) < error_{train}(h')$

One definition of overfitting

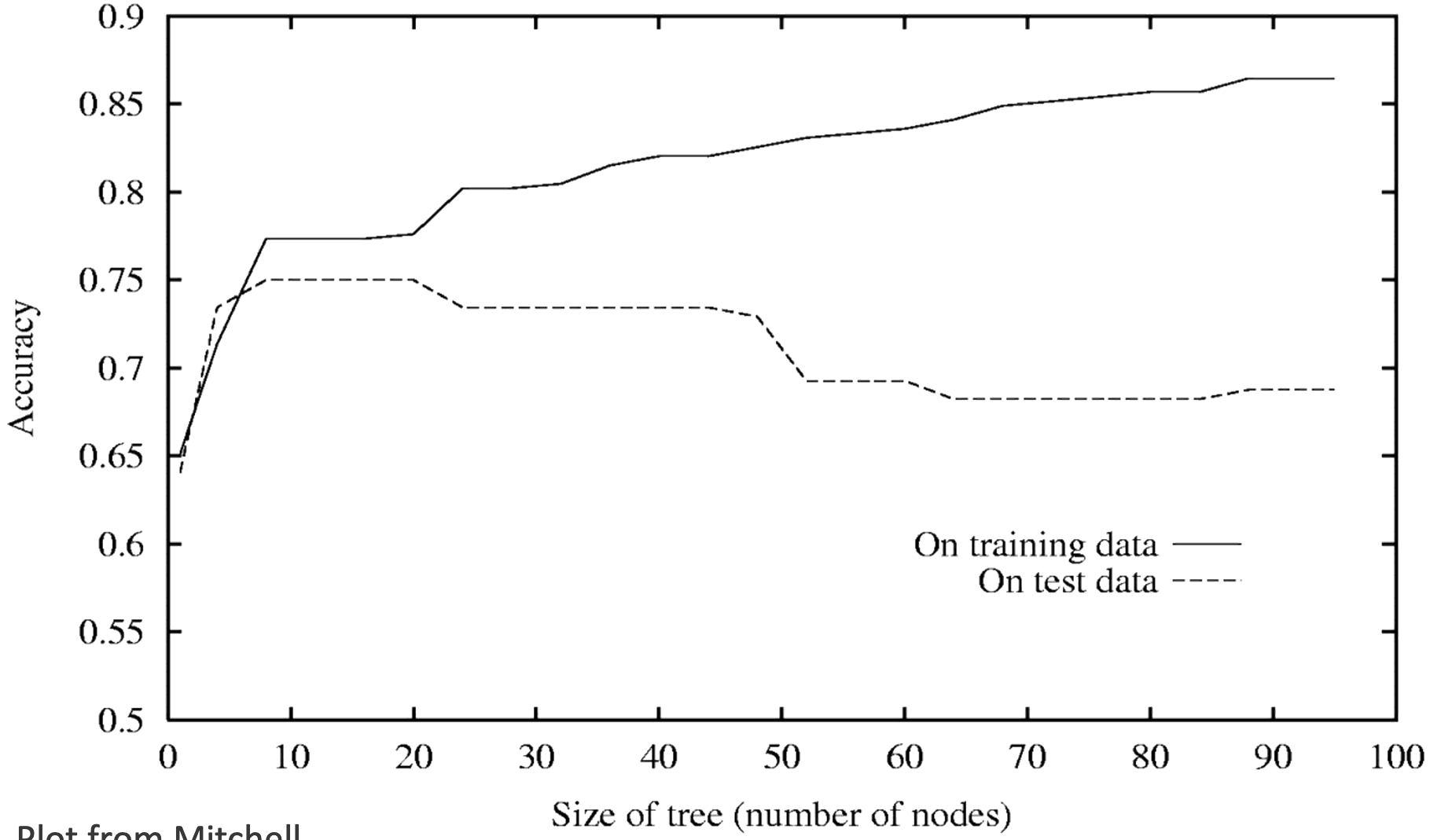
Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_D(h)$

A hypothesis h overfits the training data if there is another hypothesis h' such that

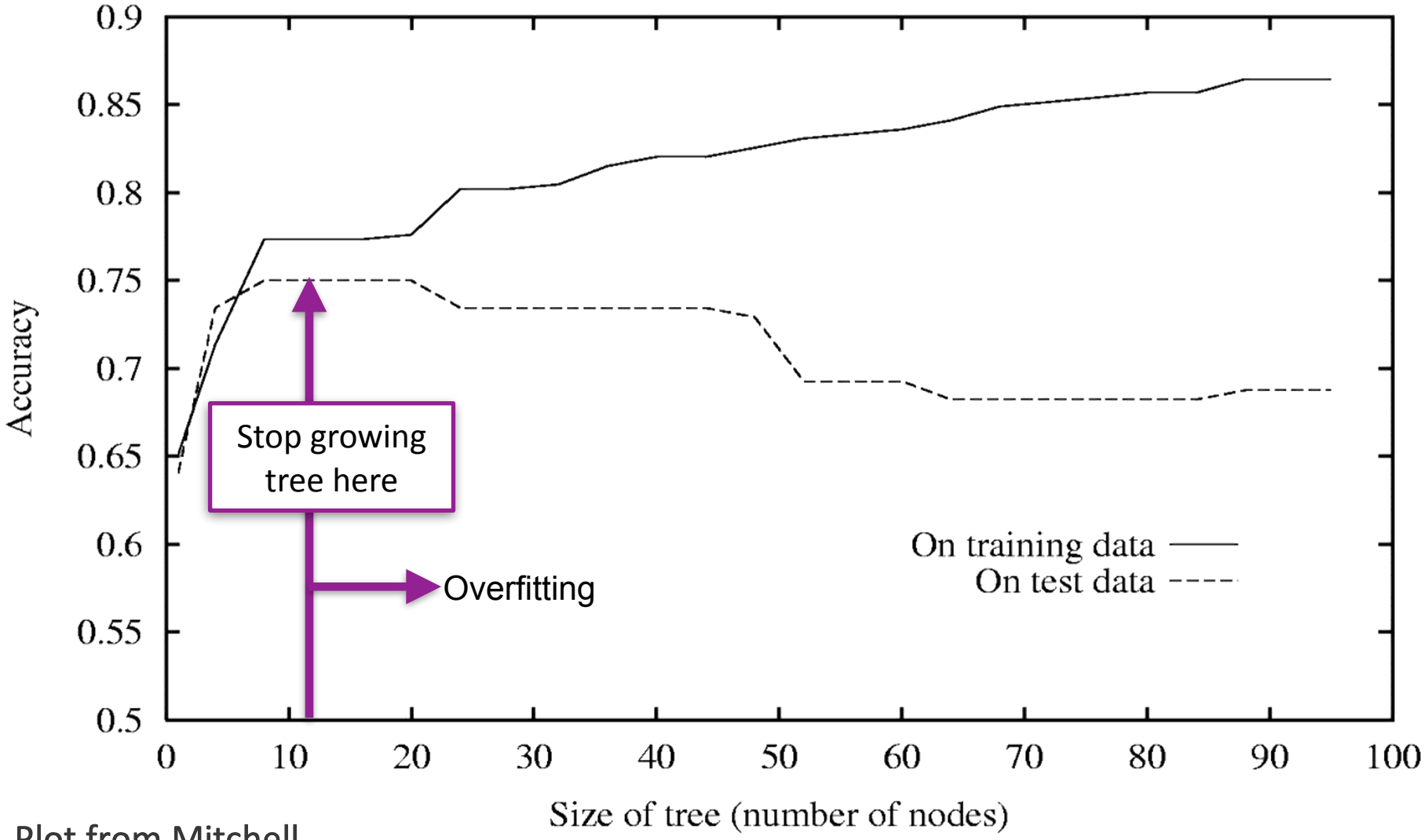
1. h has lower training error than the competing hypothesis h' but
   - $error_{train}(h) < error_{train}(h')$
2. h' generalizes better than h
   - $error_D(h) > error_D(h')$

Decision trees will overfit



Plot from Mitchell

Decision trees will overfit



Plot from Mitchell