

Decision Trees: Discussion

Machine Learning



This lecture: Learning Decision Trees

- 1. Representation:** What are decision trees?
- 2. Algorithm:** Learning decision trees
 - The ID3 algorithm: A greedy heuristic
- 3. Some extensions**

This lecture: Learning Decision Trees

1. **Representation:** What are decision trees?
2. **Algorithm:** Learning decision trees
 - The ID3 algorithm: A greedy heuristic
3. **Some extensions**

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.

Other ways to measure disorder

Example: The *MajorityError*, which computes:

“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.

Other ways to measure disorder

Example: The *MajorityError*, which computes:

“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: $\frac{1}{4}$

Works like entropy

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do [at training time](#)?

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

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do **at training time**?

Different methods to “Complete the example”:

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
 - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
 - **Exercise:** Will this change probability computations?

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do **at training time**?

Different methods to “Complete the example”:

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
 - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
 - **Exercise:** Will this change probability computations?

At test time?

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do **at training time**?

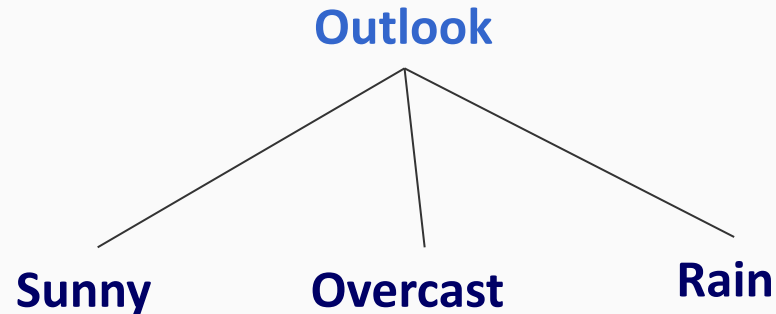
Different methods to “Complete the example”:

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
 - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
 - **Exercise:** Will this change probability computations?

At test time? Use the same method

3. Non-Boolean features

- If the features can take multiple values
 - We have seen one edge per value (i.e a multi-way split)



3. Non-Boolean features

- If the features can take multiple values
 - We have seen one edge per value (i.e a multi-way split)
 - Another option: Make the attributes Boolean by testing for each value
 - Convert `Outlook=Sunny` → {`Outlook:Sunny=True`,
`Outlook:Overcast=False`,
`Outlook:Rain=False`}
 - Or, perhaps group values into disjoint sets

3. Non-Boolean features

- If the features can take multiple values
 - We have seen one edge per value (i.e a multi-way split)
 - Another option: Make the attributes Boolean by testing for each value
 - Convert `Outlook=Sunny` → {`Outlook:Sunny=True`,
`Outlook:Overcast=False`,
`Outlook:Rain=False`}
 - Or, perhaps group values into disjoint sets
- For numeric features, use thresholds or ranges to get Boolean/discrete alternatives

4. *Overfitting*

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

$$Y = X_0$$

X_1 is irrelevant

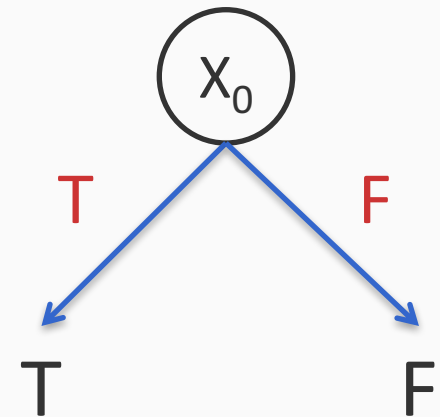
What is the decision tree for this function?

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?



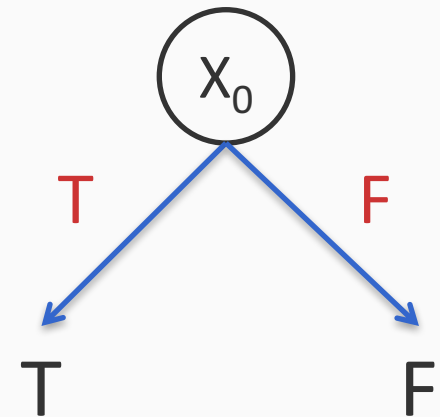
The “First Bit” function

- 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

Exercise: Convince yourself that ID3 will generate this tree

What is the decision tree for this function?



The best case scenario: Perfect data

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

The best case scenario: Perfect data

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 consistent tree

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

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

E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

Test accuracy for different input sizes

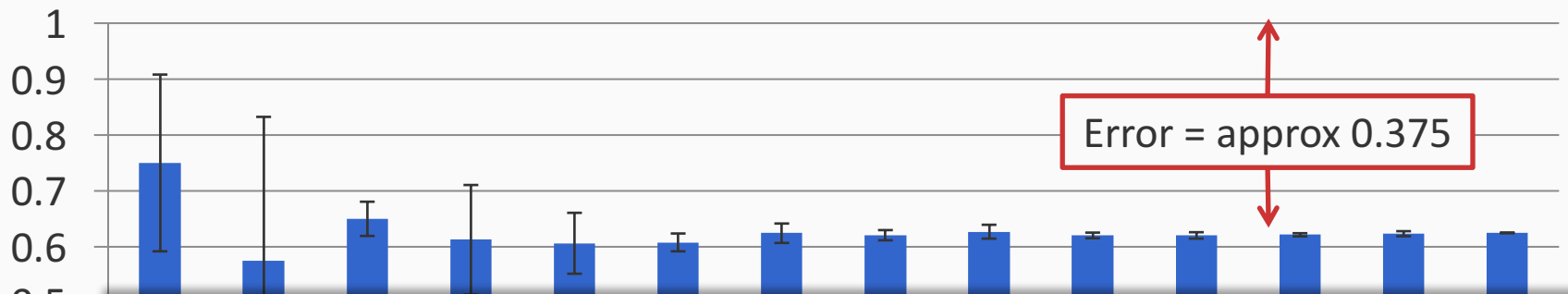


The error bars are generated by running the same experiment multiple times for the same setting

E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

Test accuracy for different input sizes



We can analytically compute test error in this case

Correct prediction:

$P(\text{Training example uncorrupted AND test example uncorrupted}) = 0.75 \times 0.75$

$P(\text{Training example corrupted AND test example corrupted}) = 0.25 \times 0.25$

$P(\text{Correct prediction}) = 0.625$

Incorrect prediction:

$P(\text{Training example uncorrupted AND test example corrupted}) = 0.75 \times 0.25$

$P(\text{Training example corrupted and AND example uncorrupted}) = 0.25 \times 0.75$

$P(\text{incorrect prediction}) = 0.375$

E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

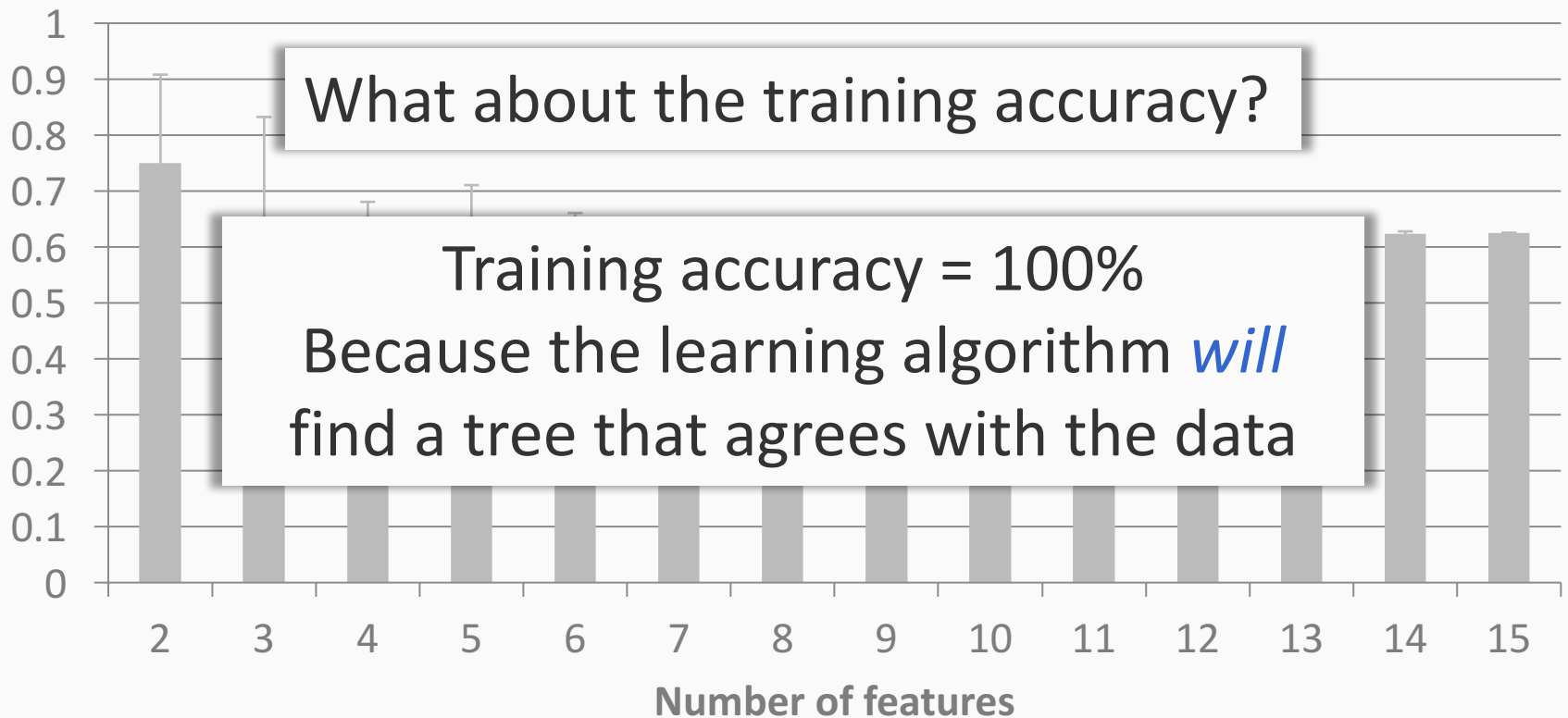
Test accuracy for different input sizes



E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

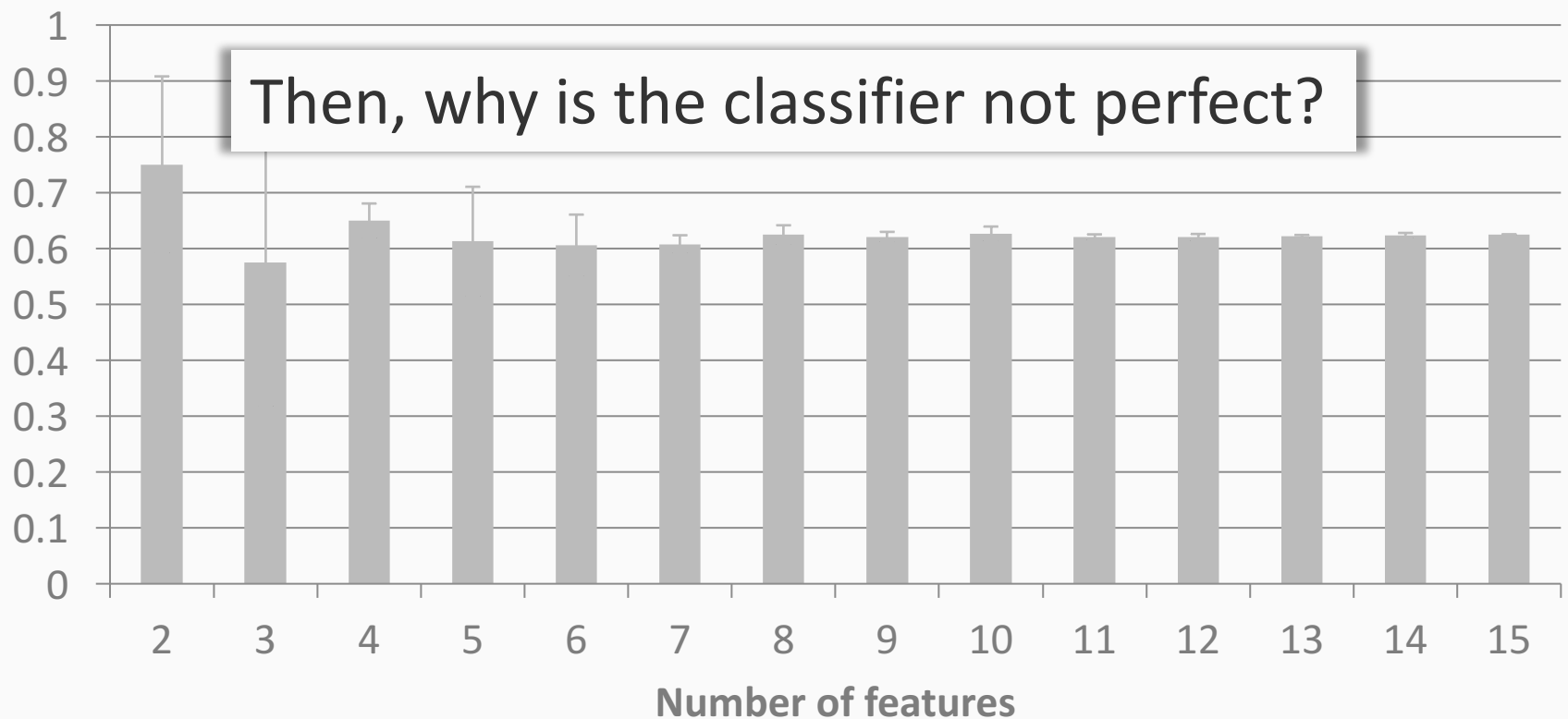
Test accuracy for different input sizes



E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

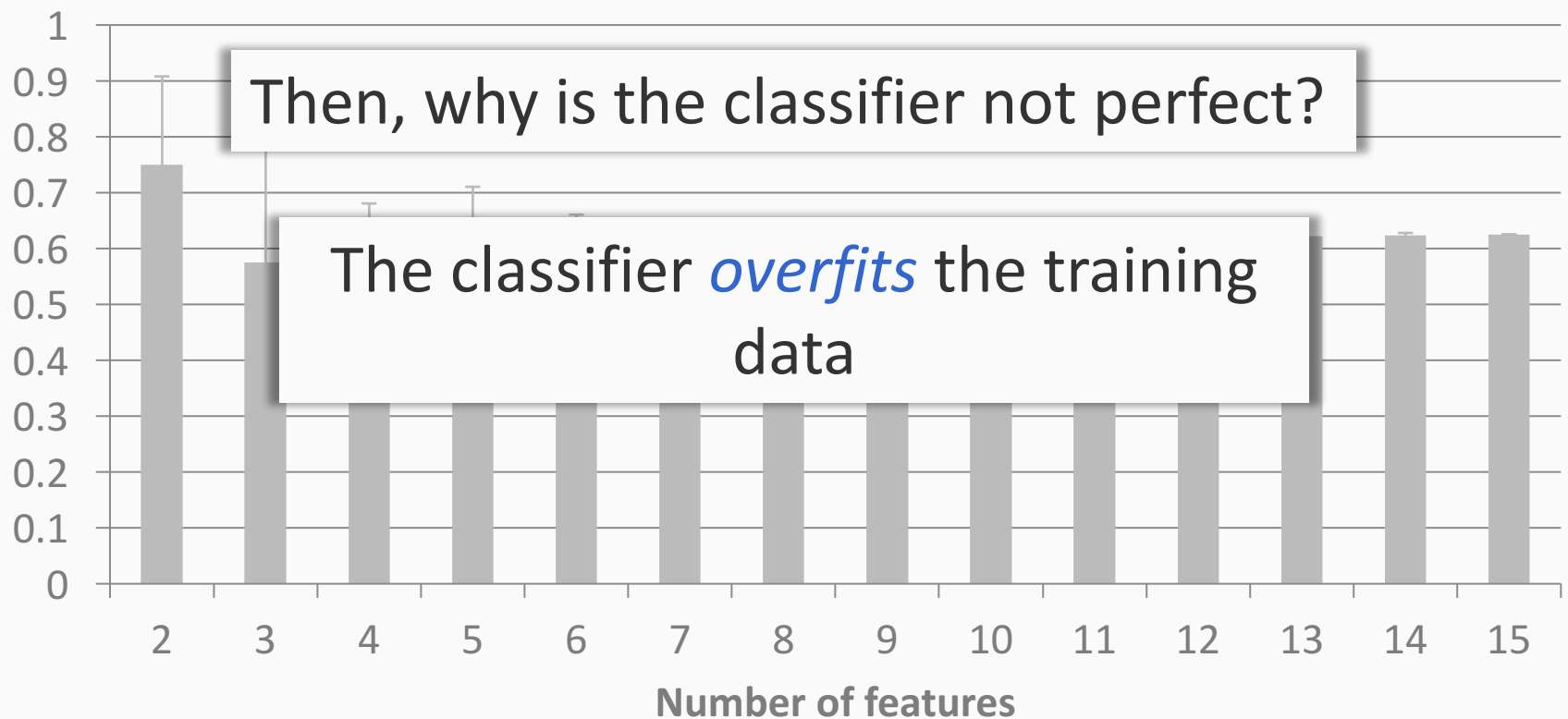
Test accuracy for different input sizes



E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.

Test accuracy for different input sizes



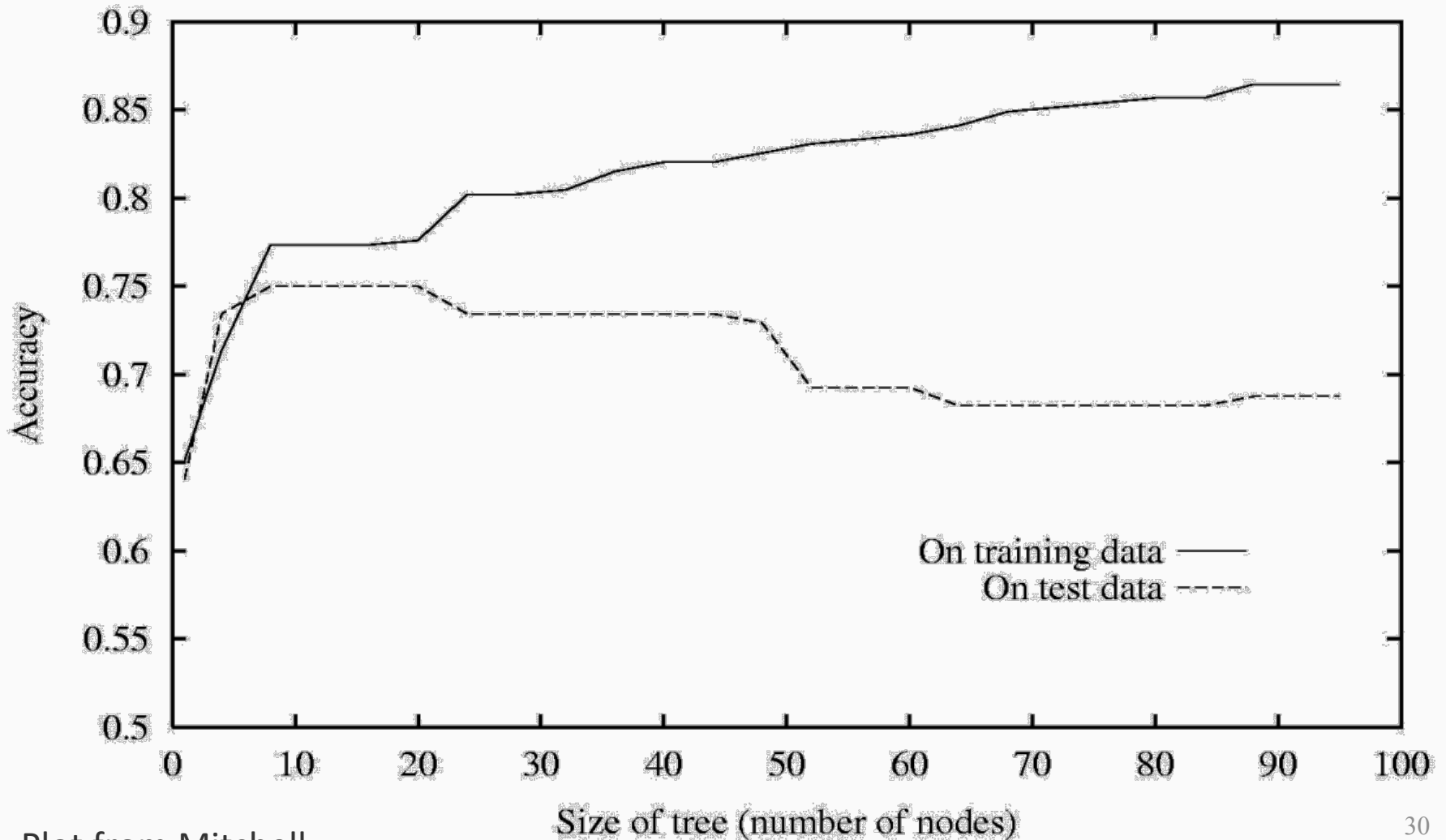
Overfitting

- The learning algorithm finds a hypothesis that fits the noise in the data
 - Irrelevant attributes or noisy examples influence the choice of the hypothesis
- May lead to poor performance on future examples

Overfitting: One definition

- Data comes from a probability distribution $D(X, Y)$
- We are using a hypothesis space H
- Errors:
 - **Training error** for hypothesis $h \in H$: $\text{error}_{\text{train}}(h)$
 - **True error** for $h \in H$: $\text{error}_D(h)$
- A hypothesis h **overfits** the training data if there is another hypothesis h' such that
 1. $\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$
 2. $\text{error}_D(h) > \text{error}_D(h')$

Decision trees *will* overfit



Avoiding overfitting with decision trees

Occam's Razor

Favor simpler (in this case, shorter) hypotheses

Why? Fewer shorter trees, less likely to fit better by coincidence

- Some approaches:
 1. Fix the depth of the tree
 - **Decision stump** = a decision tree with only one level
 - Typically will not be very good by itself
 - But, we will revisit decision stumps later (short decision trees can make very good features for a second layer of learning)

Avoiding overfitting with decision trees

Occam's Razor

Favor simpler (in this case, shorter) hypotheses

Why? Fewer shorter trees, less likely to fit better by coincidence

- Some approaches:
 2. Optimize on a *held-out set* (also called *development set* or *validation set*) while growing the trees
 - Split your data into two parts –training set and held-out set
 - Grow your tree on the training split and check the performance on the held-out set after every new node is added
 - If growing the tree hurts validation set performance, stop growing

Avoiding overfitting with decision trees

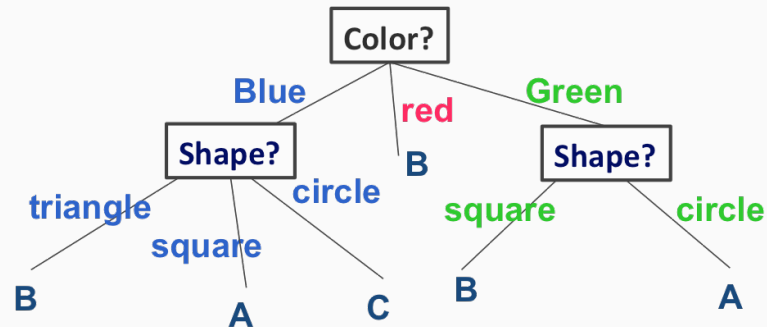
Occam's Razor

Favor simpler (in this case, shorter) hypotheses

Why? Fewer shorter trees, less likely to fit better by coincidence

- Some approaches:
 3. Grow full tree and then prune as a post-processing step in one of several ways:
 1. Use a validation set for pruning from bottom up greedily
 2. Convert the tree into a set of rules (one rule per path from root to leaf) and prune each rule independently

Reminder: Decision trees are rules



- If Color=Blue and Shape=triangle, then output = B
- If Color=Red, then output = B

...

Summary: Decision trees

- A popular machine learning tool
 - Prediction is easy
 - If we have Boolean features and binary classification, decision trees can represent any Boolean function
- Greedy heuristics for learning
 - ID3 algorithm (using information gain)
 - Robust implementations of some variants (eg. C4.5 algorithm) exist
- (Can be used for regression too)
- Decision trees are prone to overfitting unless you take care to avoid it