Decision Trees

CMSC 422

SOHEIL FEIZI
sfeizi@cs.umd.edu

Slides adapted from MARINE CARPUAT

Last lecture: introducing machine learning

What does "learning by example" mean?

- Classification tasks

- Learning requires examples + inductive bias
- Generalization vs. memorization

- Formalizing the learning problem
  - Function approximation
  - Learning as minimizing expected loss

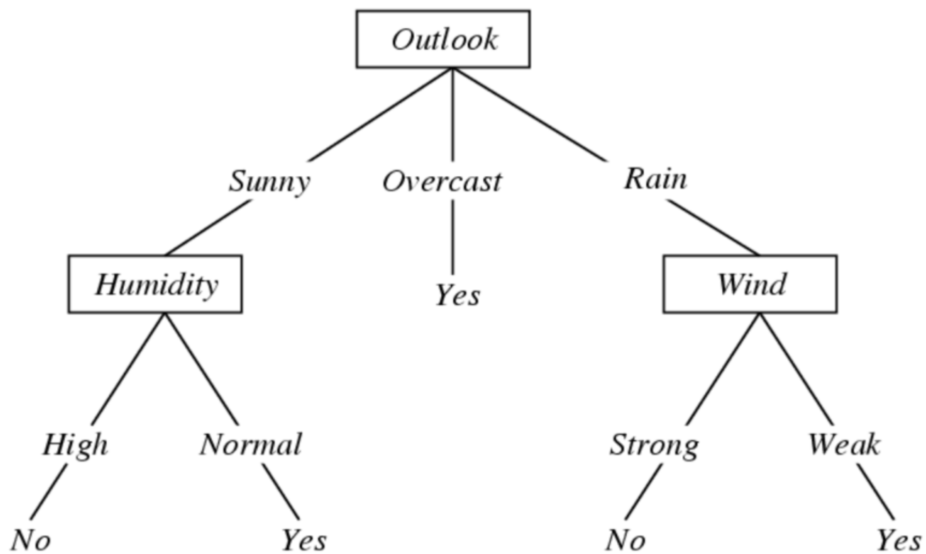
Today: Decision Trees

- **What is a decision tree?**
- How to learn a decision tree from data?
- What is the inductive bias?
- Generalization?

An example training set

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

A decision tree to decide whether to play tennis



Decision Trees

- Representation
  - Each internal node tests a feature
  - Each branch corresponds to a feature value
  - Each leaf node assigns a classification
    - or a probability distribution over classifications

- Decision trees represent functions that map examples in X to classes in Y

f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?

Exercise

- How would you represent the following Boolean functions with decision trees?
  - AND
  - OR
  - XOR

Today: Decision Trees

- What is a decision tree?
- **How to learn a decision tree from data?**
- What is the inductive bias?
- Generalization?

Function Approximation with Decision Trees

Problem setting

- Set of possible instances X
  - Each instance $x \in X$ is a feature vector $x = [x_1, \dots, x_D]$
- Unknown target function $f: X \to Y$
  - Y is discrete valued
- Set of function hypotheses $H = \{h \mid h: X \to Y\}$
  - Each hypothesis h is a decision tree

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \dots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Decision Trees Learning

- Finding the hypothesis $h \in H$
  - That minimizes training error
  - Or maximizes training accuracy

- How?
  - H is too large for exhaustive search!
  - We will use a heuristic search algorithm which
    - Picks questions to ask, in order
    - Such that classification accuracy is maximized

Top-down Induction of Decision Trees

```
CurrentNode = Root

DTtrain(examples for CurrentNode,features at CurrentNode):
    1. Find F, the “best” decision feature for next node
    2. For each value of F, create new descendant of node
    3. Sort training examples to leaf nodes
    4. If training examples perfectly classified
         Stop
       Else
         Recursively apply DTtrain over new leaf nodes
```

How to select the “best” feature?

- A good feature is a feature that lets us make correct classification decision

- One way to do this:
  - select features based on their classification accuracy

- Let’s try it on the PlayTennis dataset

Let's build a decision tree using features W, H, T

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Partitioning examples according to Humidity feature

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Partitioning examples:
H = Normal

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

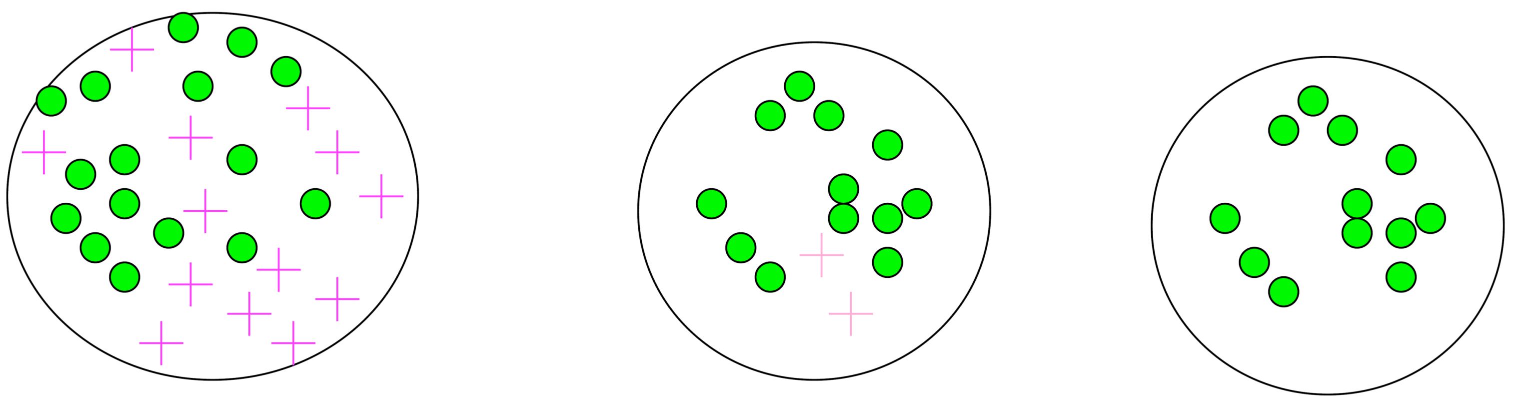
Partitioning examples:
H = Normal and W = Strong

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

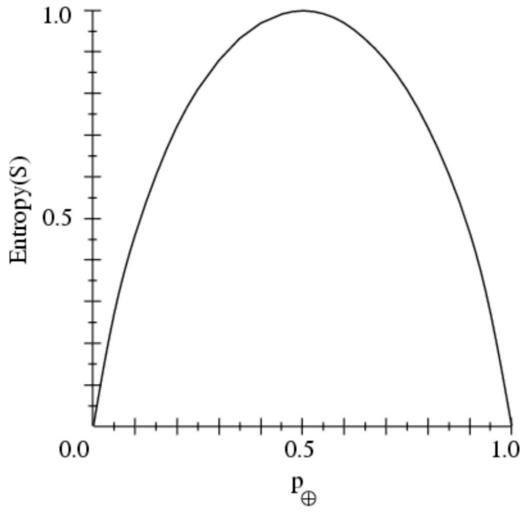
Another feature selection criterion: Entropy

- Used in the ID3 algorithm [Quinlan, 1963]
  - pick feature with smallest entropy to split the examples at current iteration

- Entropy measures impurity of a sample of examples



Sample Entropy



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$H(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy

Entropy $H(X)$ of a random variable X

# of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient possible code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Conditional Entropy

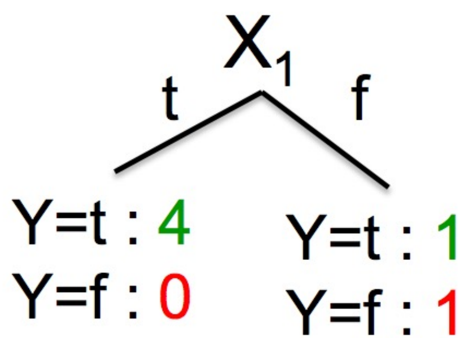
Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$



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

$$H(Y|X_1) = -\,4/6\,(1 \log_2 1 + 0 \log_2 0) - 2/6\,(1/2 \log_2 1/2 + 1/2 \log_2 1/2) = 2/6$$

slide from David Sontag, NYU

Information gain

- Decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y \mid X)$$

In our running example:

$IG(X_1) = H(Y) – H(Y|X_1)$
$= 0.65 – 0.33$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

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

slide from David Sontag, NYU

Today: Decision Trees

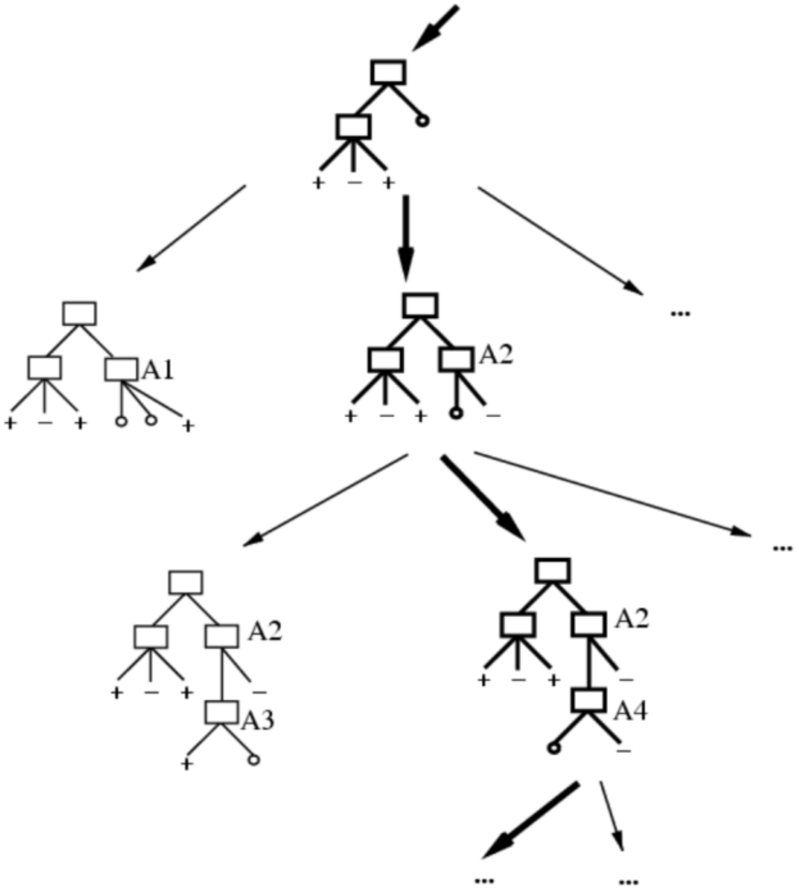
- What is a decision tree?
- How to learn a decision tree from data?
- **What is the inductive bias?**
- Generalization?

Inductive bias in decision tree learning

```
CurrentNode = Root

DTtrain(examples for CurrentNode,features at CurrentNode):
    1. Find F, the “best” decision feature for next node
    2. For each value of F, create new descendant of node
    3. Sort training examples to leaf nodes
    4. If training examples perfectly classified
         Stop
       Else
         Recursively apply DTtrain over new leaf nodes
```

Inductive bias in decision tree learning



- Our learning algorithm performs heuristic search through space of decision trees
- It stops at smallest acceptable tree

- Occam's razor: prefer the simplest hypothesis that fits the data

Why prefer short hypotheses?

- Pros
  - Fewer short hypotheses than long ones
    - A short hypothesis that fits the data is less likely to be a statistical coincidence
- Cons
  - What's so special about short hypotheses?

Evaluating the learned hypothesis h

- Assume
  - we've learned a tree h using the top-down induction algorithm
  - It fits the training data perfectly
- Are we done? Can we guarantee we have found a good hypothesis?

Recall: Formalizing Induction

- Given
  - a loss function l
  - a sample from some unknown data distribution D

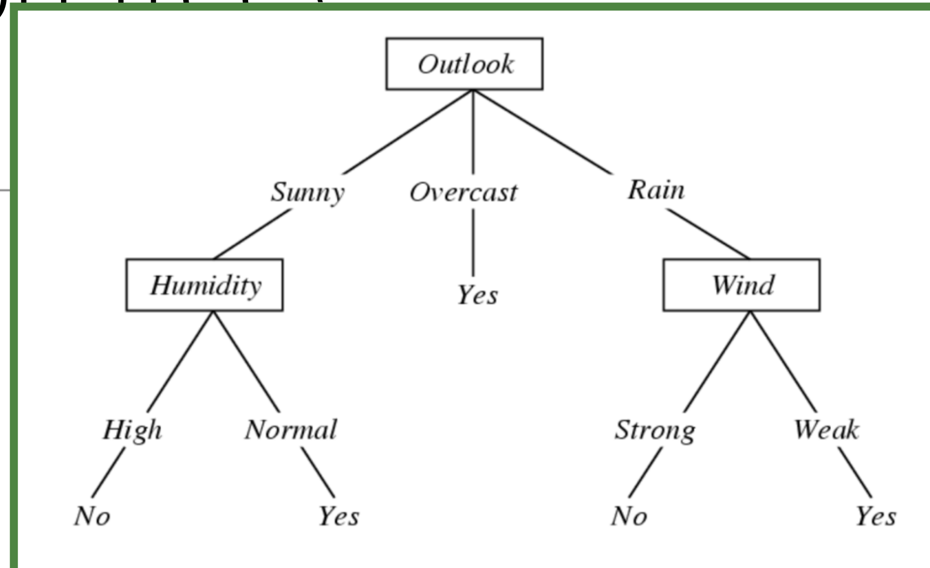
- Our task is to compute a function f that has low expected error over D with respect to l.

$$\mathbb{E}_{(x,y)\sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x,y)\, l(y, f(x))$$

Training error is not sufficient

- We care about **generalization** to new examples
- A tree can classify training data perfectly, yet classify new examples incorrectly
  - Because training examples are only a sample of data distribution
    - a feature might correlate with class by coincidence
  - Because training examples could be noisy
    - e.g., accident in labeling

Let’s add a noisy training example. How does this affect the learned decision tree?



Day	Outlook	Temperature	Humidity	Wind	
D1	Sunny	Hot	High	Weak	
D2	Sunny	Hot	High	Strong	
D3	Overcast	Hot	High	Weak	
D4	Rain	Mild	High	Weak	
D5	Rain	Cool	Normal	Weak	
D6	Rain	Cool	Normal	Strong	
D7	Overcast	Cool	Normal	Strong	
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No
D15	Sunny	Hot	Normal	Strong	No

Overfitting

- Consider a hypothesis h and its:
  - Error rate over training data $error_{train}(h)$
  - True error rate over all data $error_{true}(h)$

- We say h overfits the training data if

$$error_{train}(h) < error_{true}(h)$$

- Amount of overfitting =

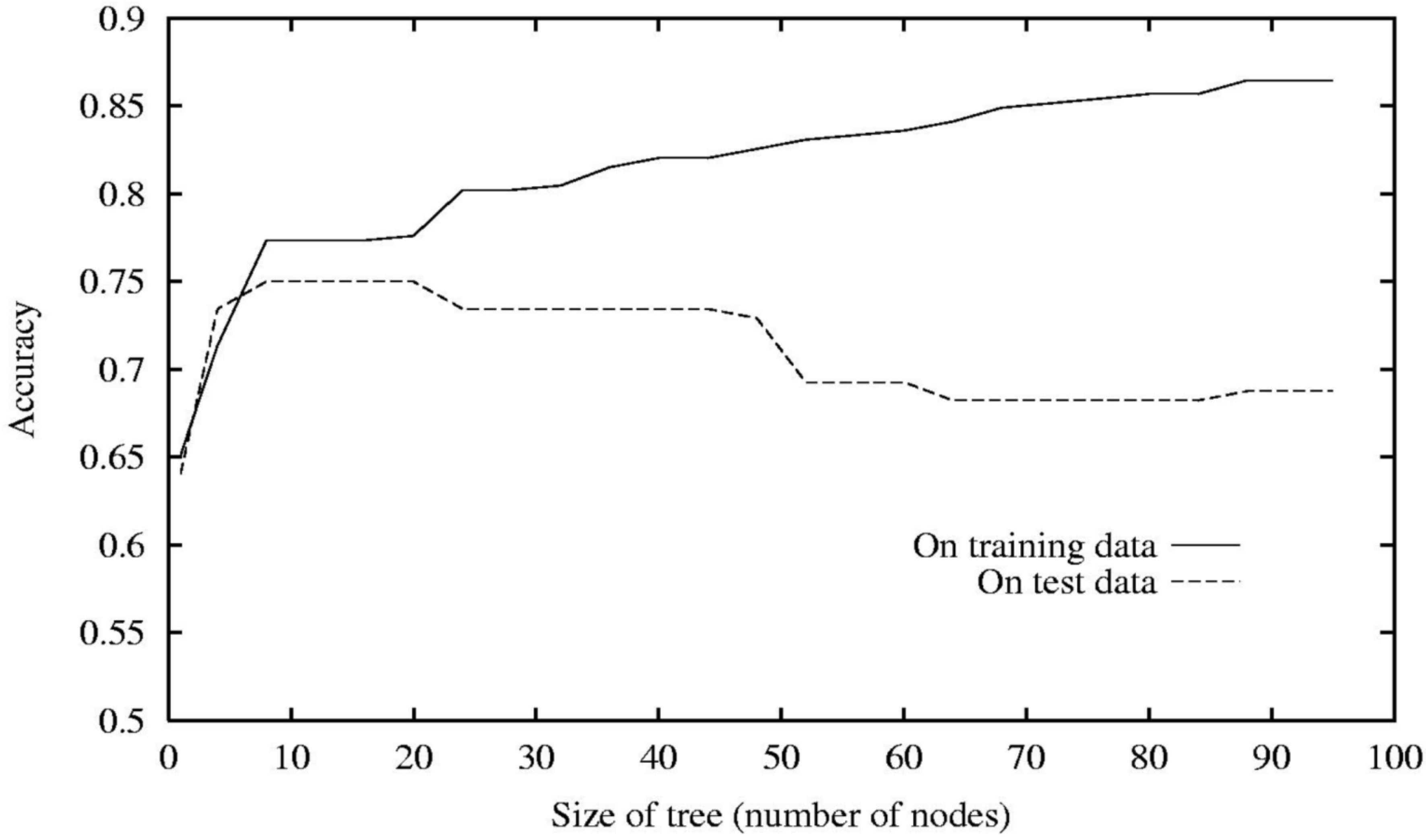
$$error_{true}(h) - error_{train}(h)$$

Evaluating on test data

- Problem: we don't know $error_{true}(h)$!

- Solution:
  - we set aside a test set
    - some examples that will be used for evaluation
  - we don't look at them during training!
  - after learning a decision tree, we calculate
    $$error_{test}(h)$$

Measuring effect of overfitting in decision trees



Underfitting/Overfitting

- Underfitting
  - Learning algorithm had the opportunity to learn more from training data, but didn’t
- Overfitting
  - Learning algorithm paid too much attention to learn noisy part of the training data; the resulting tree doesn’t generalize

- What we want:
  - A decision tree that neither underfits nor overfits
  - Because it is is expected to do best in the future

Today: Decision Trees

- What is a decision tree?
- How to learn a decision tree from data?
  - Top-down induction to minimize classification error
- What is the inductive bias?
  - Occam's razor: preference for short trees
- Generalization?
  - Overfitting can be an issue