CMSC 422 Introduction to Machine Learning

Lecture 3 Decision Trees & Limits of Learning

Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

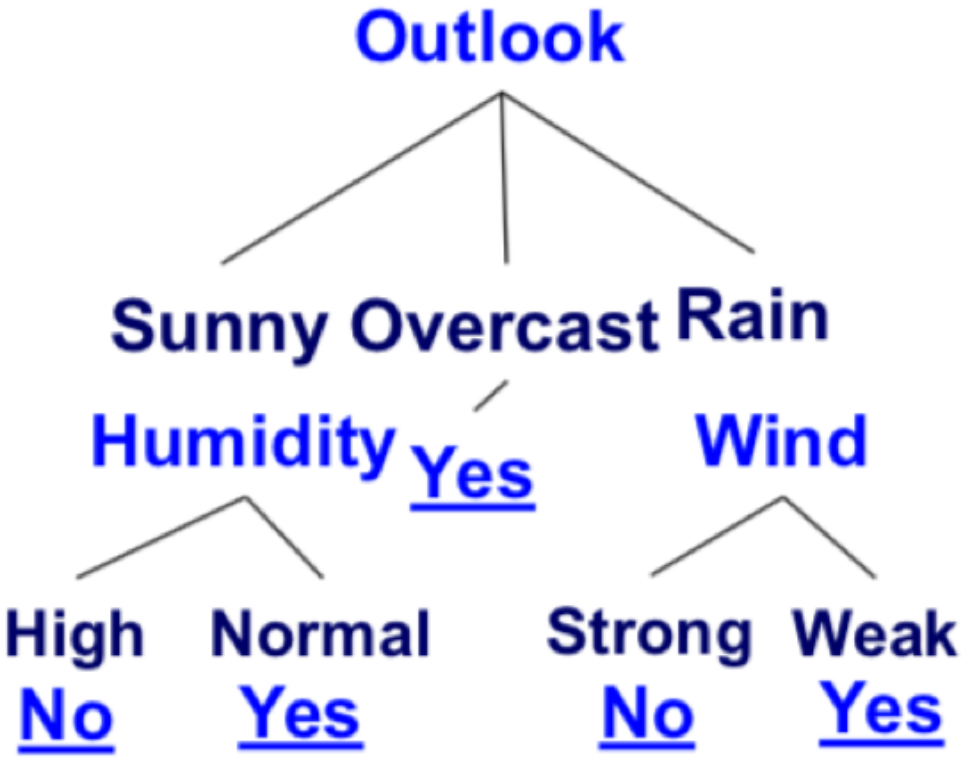
Last Lecture: Decision Trees

- **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



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

Today: Decision Trees

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

A decision tree to distinguish homes in New York from homes in San Francisco

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

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
```

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - However, finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on. (**Sub optimality**)

Information Gain

- The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S, a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

  - Original/current **set of examples** is S.
  - S_v is the subset of S for which attribute a has value v.
  - The entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set.
    - Partitions of low entropy (imbalanced splits) lead to high gain
  - **Select the attribute with largest information gain.**

induceDecisionTree(S)

1. Does S uniquely define a class?

   if all $s \in S$ have the same label y: **return** S;

2. Find the feature with the most information gain:

   $i = \text{argmax}_i Gain(S, X_i)$

3. Add children to S :

   for k **in** Values(X_i):

   $S_k = \{s \in S | X_i = k\}$

   addChild(S , S_k)

   induceDecisionTree(S_k)

   return S;

Today: Decision Trees

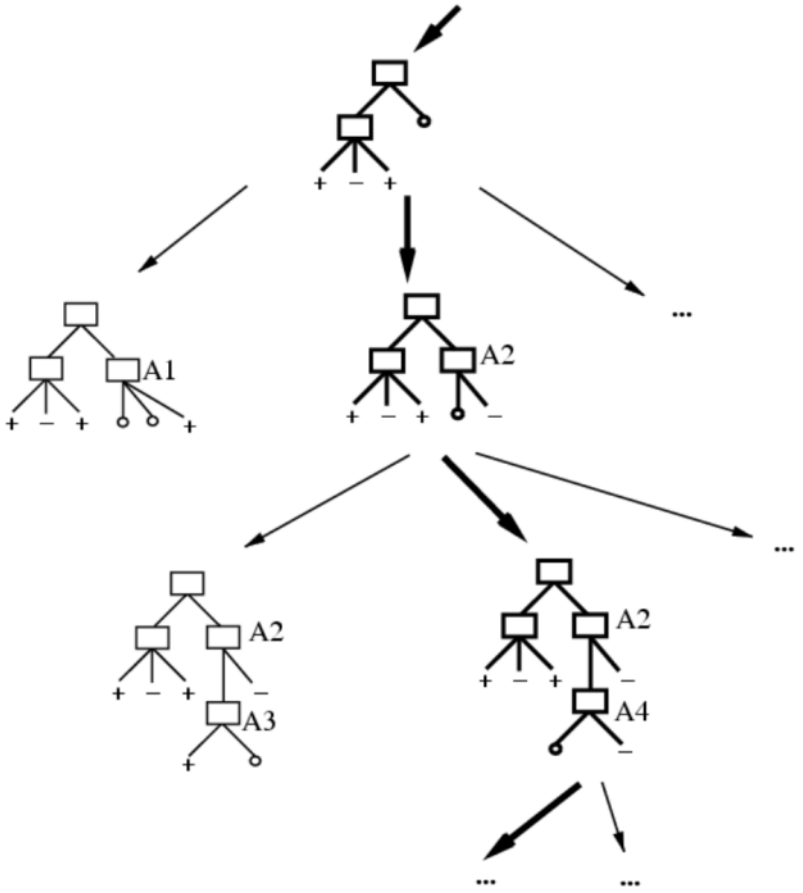
What is a decision tree?

How to learn a decision tree from data?

What is the inductive bias?

Generalization?

Inductive bias in decision tree learning



- Our learning algorithm performs heuristic search through space of decision trees
- It stops at smallest acceptable tree
- Why do we prefer small trees?
  - 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.

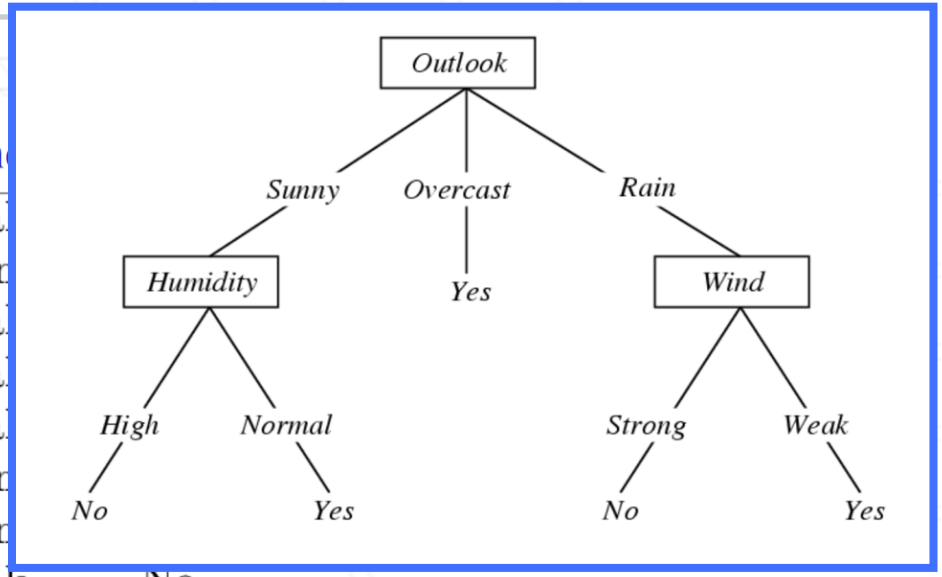
$$\mathbb{E}_{(x,y)\sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x,y)\, l(y, f(x))$$

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	Win	
D1	Sunny	Hot	High	Wea	
D2	Sunny	Hot	High	Stron	
D3	Overcast	Hot	High	Wea	
D4	Rain	Mild	High	Wea	
D5	Rain	Cool	Normal	Wea	
D6	Rain	Cool	Normal	Stron	
D7	Overcast	Cool	Normal	Stron	
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 $error_{train}(h)$:

$$error_{train}(h) = \sum_{n=1}^{N} \frac{1}{N} l(y^{(n)}, h(x^{(n)}))$$

  - True error rate over all data $error_{true}(h)$:

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

- We say h overfits the training data if

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

- Amount of overfitting =

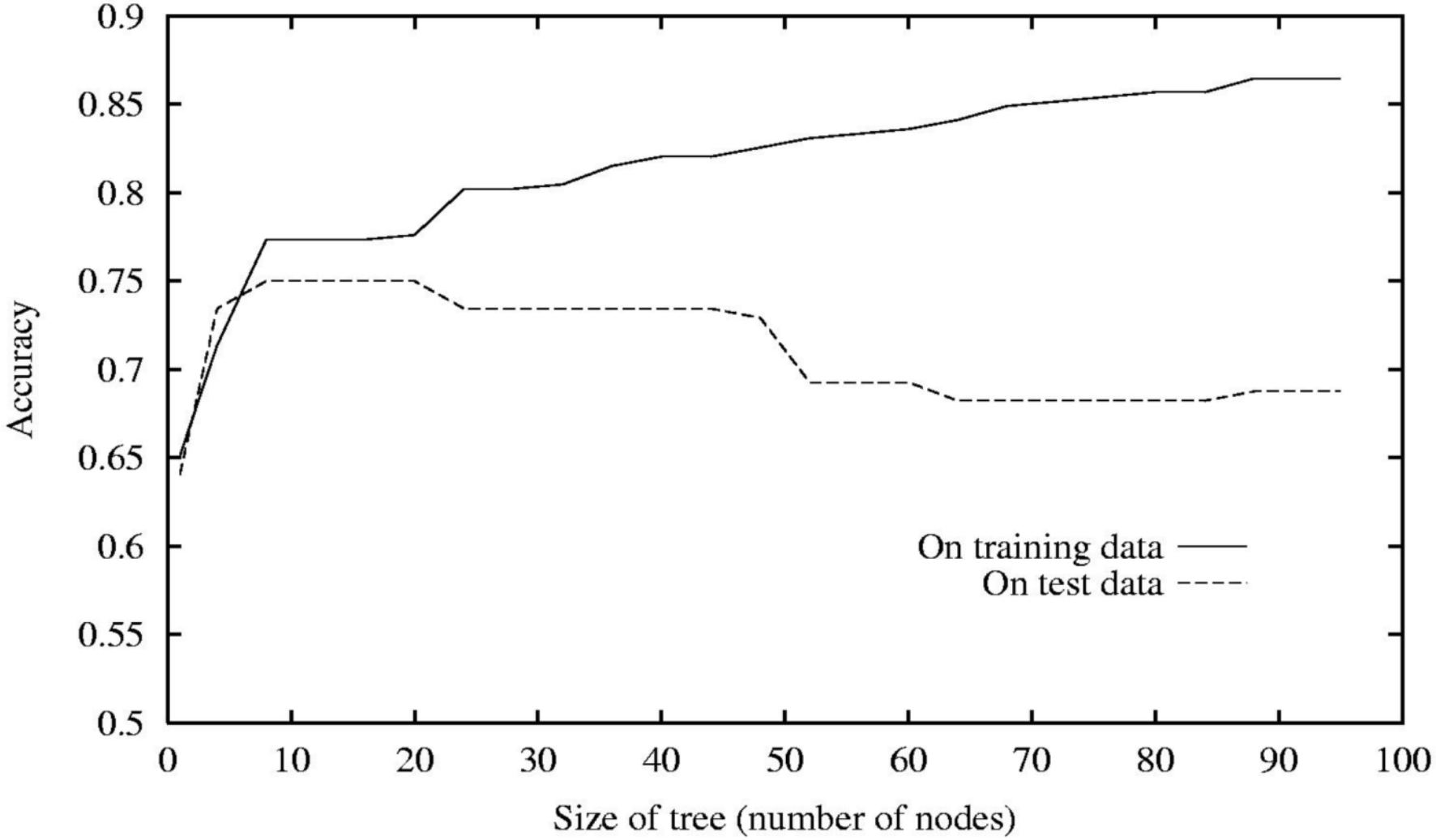
$$error_{true}(h) - error_{train}(h)$$

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 $error_{test}(h)$

$$error_{test}(h) = \sum_{n=1}^{N} \frac{1}{N} l({y_{test}}^{(n)}, h({x_{test}}^{(n)}))$$

Effect of overfitting in decision trees



Overfitting

- Another way of putting it

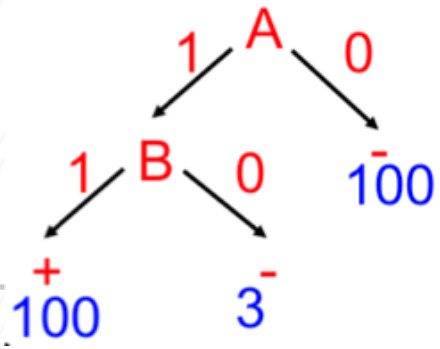
- A hypothesis h is said to overfit the training data, if there is another hypothesis h', such that
  - h has a smaller error than h' on the training data
  - but h has larger error on the test data than h'.

Underfitting/Overfitting

- Underfitting
  - Learning algorithm had the opportunity to learn more from training data, but didn't
  - Or didn't have sufficient data to learn from
- Overfitting
  - Learning algorithm paid too much attention to idiosyncracies of the training data; the resulting tree doesn't generalize

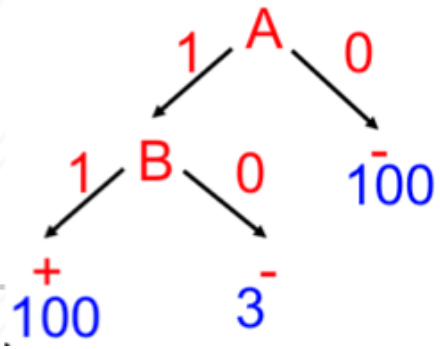
- What we want:
  - A decision tree that neither underfits nor overfits
  - Because it is expected to do best in the future

Pruning A Decision Tree



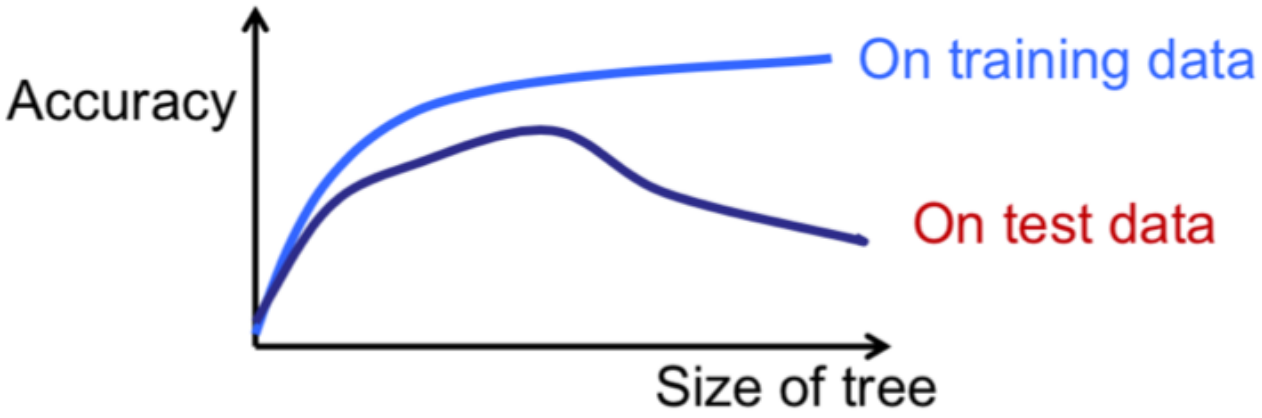
- Prune = remove leaves and assign majority label of the parent to all items
- Prune the children of S if:
  - All children are leaves, and
  - The accuracy on the validation set does not decrease if we assign the most frequent class label to all items at S.

Avoid Overfitting



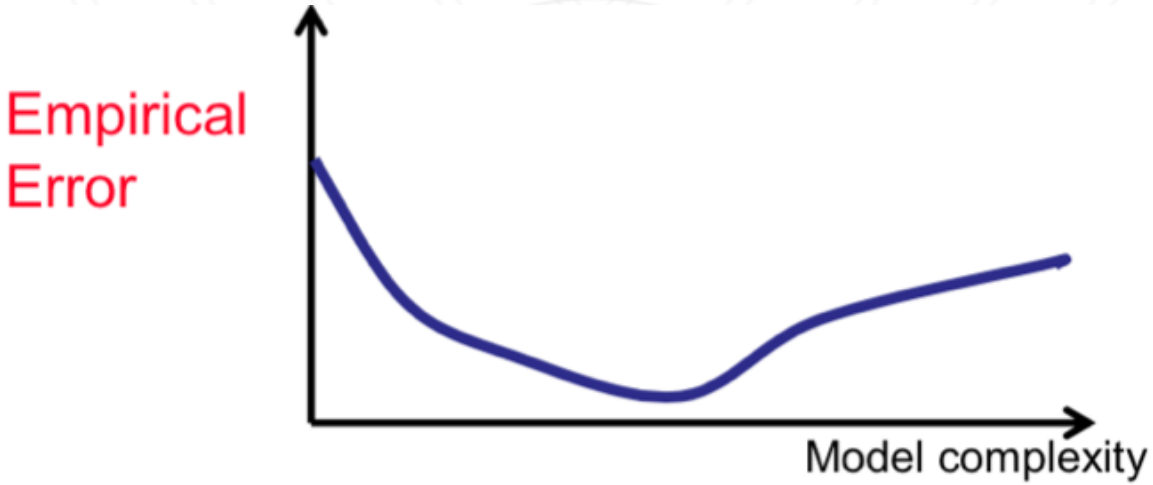
- Two basic approaches
  - Pre-pruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Post-pruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Methods for evaluating subtrees to prune
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions?
- This is related to the notion of regularization that we will see in other contexts–keep the hypothesis simple.

Overfitting



- A decision tree overfits the training data when its accuracy on the training data goes up but its accuracy on unseen data goes down.

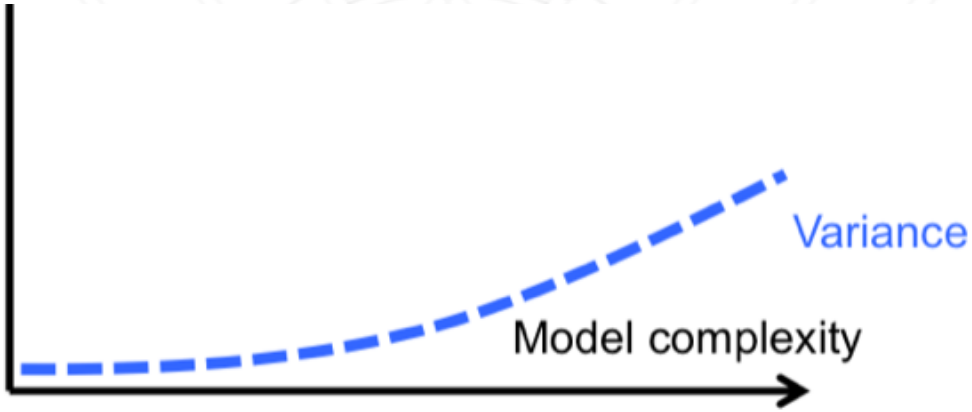
Overfitting



- Empirical error (= on a given data set):

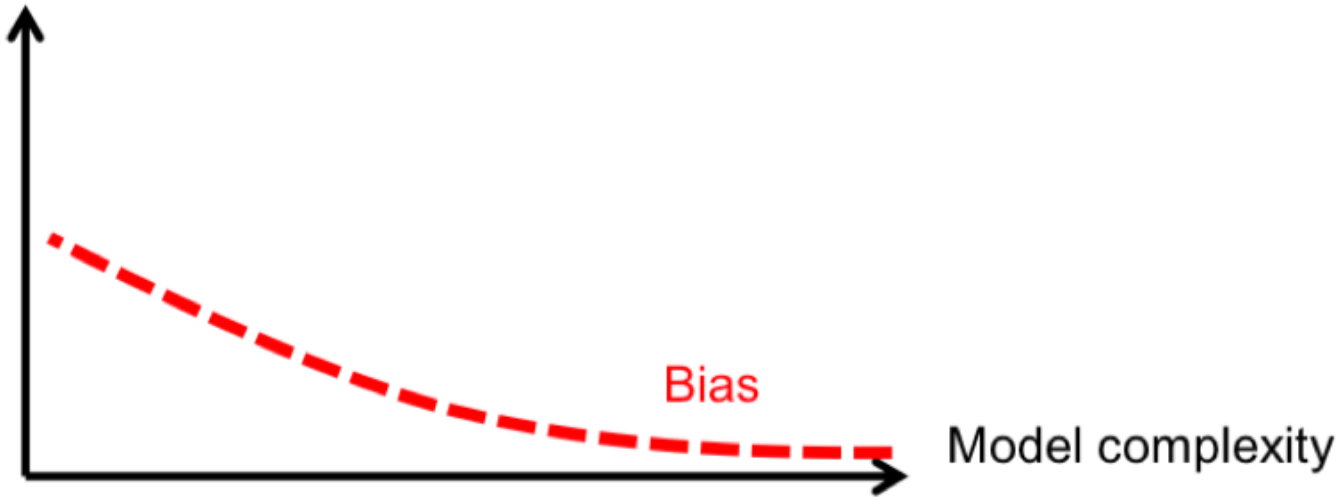
The percentage of items in this data set are misclassified by the classifier *f*.

Variance of A Learner (informally)



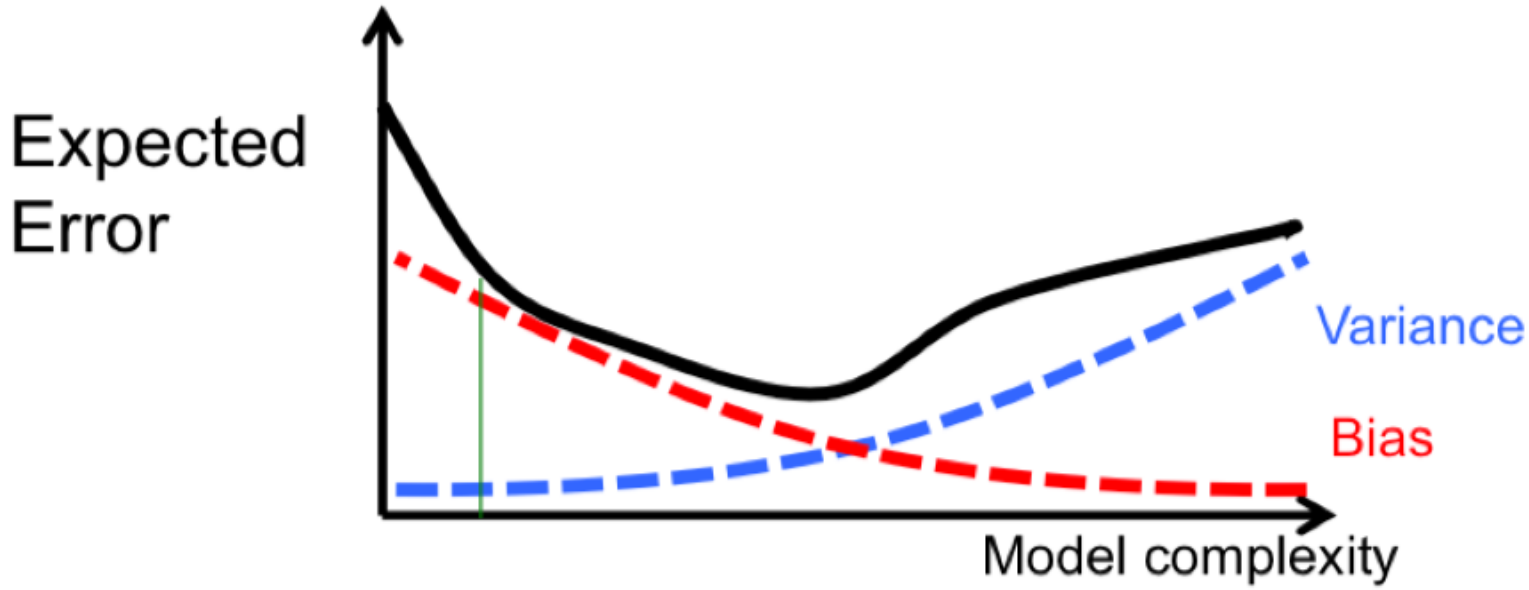
- How susceptible is the learner to minor changes in the training data?
  - (i.e. to different samples from P(X, Y))
- Variance increases with model complexity

Bias of A Learner (informally)



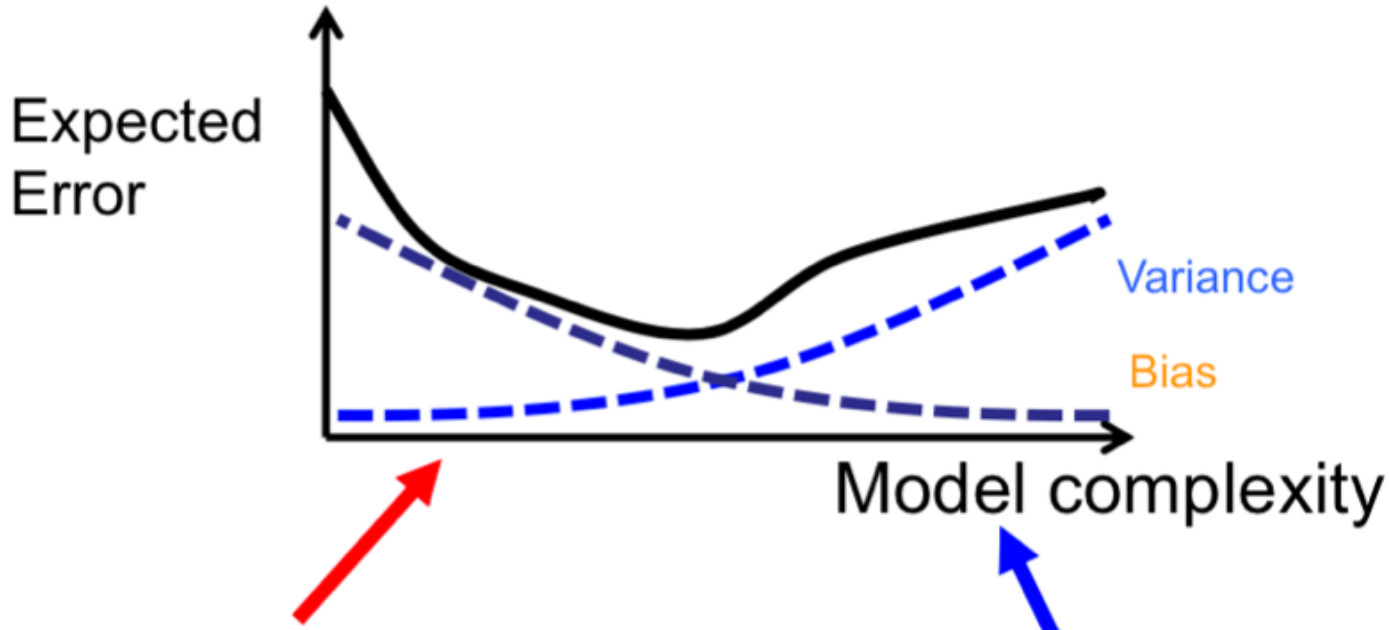
- How likely is the learner to identify the target hypothesis?
- Bias is low when the model is expressive (low empirical error)
- Bias is high when the model is (too) simple
  - The larger the hypothesis space is, the easier it is to be close to the true hypothesis.

Impact of Bias and Variance



- Expected error ≈ bias + variance

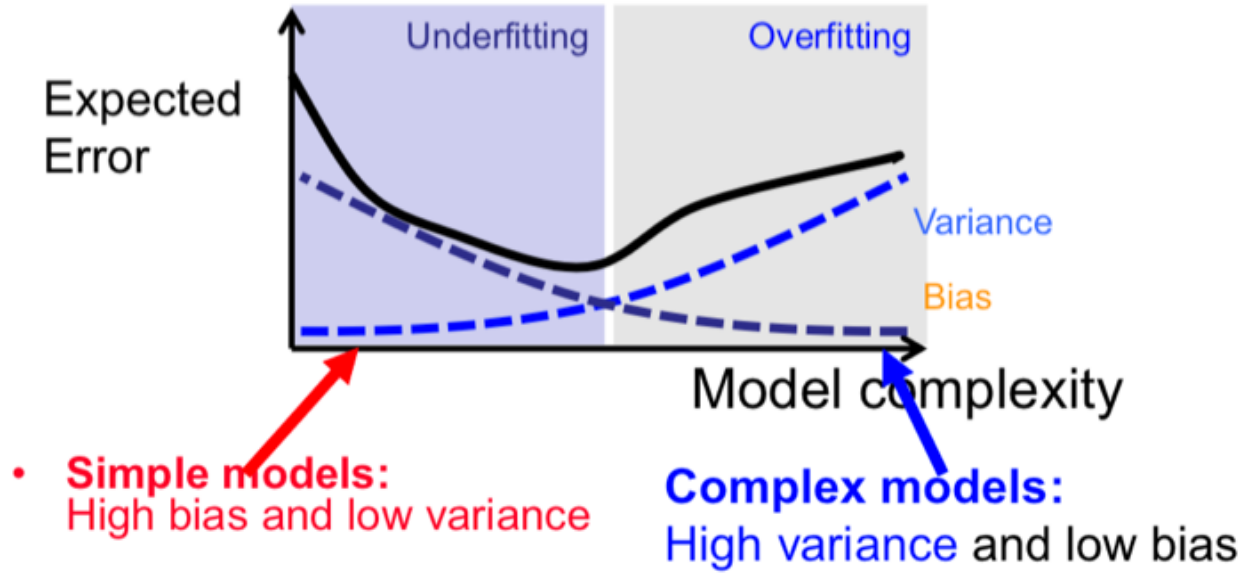
Model Complexity



- **Simple models:** High bias and low variance

Complex models: High variance and low bias

Underfitting and Overfitting



- This can be made more accurate for some loss functions.
- We will develop a more precise and general theory that trades expressivity of models with empirical error

Expectation

- X is a discrete random variable with distribution $P(X)$:
- Expectation of X ($\mathbb{E}[X]$), aka. the mean of X (μ_X)

$$\mathbb{E}[X] = \sum_x P(X = x)x := \mu_X$$

- Expectation of a function of X ($\mathbb{E}[f(X)]$)

$$\mathbb{E}[f(X)] = \sum_x P(X = x)f(X = x)$$

- If X is continuous, replace sums with integrals

Variance and Standard Deviation

- Squared difference between X and its mean:

$$(X - \mathbb{E}[X])^2 = (x - \mu_x)^2$$

- Variance of X:

$$\mathrm{V}ar[X] = \mathbb{E}[(X - \mu_x)^2] = \sigma_X^2$$

- The expected value of the square difference between X and its mean
- Standard deviation of X:

$$\sigma_X = \sqrt{\sigma_X^2} = \sqrt{Var(X)}$$

- (= the square root of the variance)

Variance (continued)

- The variance of X is equal to the expected value of X^2 minus the square of its mean

$$Var[X] = \mathbb{E}[X] - (\mathbb{E}[X])^2$$
$$= \mathbb{E}[X^2] - \mu_X^2$$

- Proof:

$$Var[X] = \mathbb{E}[(X - \mu_X)^2]$$
$$= \mathbb{E}[X^2 - 2\mu_X X + \mu_X^2]$$
$$= \mathbb{E}[X^2] - 2\mu_X \mathbb{E}[X] + \mu_X^2$$
$$= \mathbb{E}[X^2] - \mu_X^2$$

Practical impact on decision tree learning

What we want:

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

How can we encourage that behavior?

- Set a maximum tree depth D

Your thoughts?

What are the pros and cons of decision trees?

DEALING WITH DATA

What real data looks like…

Class y

Example

1 robocop is an intelligent science fiction thriller and social satire , one with class and style . the film , set in old detroit in the year 1991 , stars peter weller as murphy , a lieutenant on the city's police force . 1991's detroit su department run by a p inc .) whose empl strike . to make matte ers has been terrorizin

0 do the f ? they have resurrecte nto a live action hoc embarrassi . wasn't mr . magoo enough , people ? obviously not . inspector gadget is not what i would call ideal family entertainment . […]

How would you define input vectors x to represent each example? What features would you use?

Train/Dev/Test Sets

In practice, we always split examples into 3 distinct sets

- **Training set**
  - Used to learn the **parameters** of the ML model
  - e.g., what are the nodes and branches of the decision tree
- **Development set**
  - aka tuning set, aka validation set, aka held-out data
  - Used to learn **hyperparameters**
    - Parameter that controls other parameters of the model
    - e.g., max depth of decision tree
- **Test set**
  - Used to evaluate how well we're doing on new unseen examples

Cardinal rule of machine learning:

Never *ever* touch your test data!

Summary: what you should know

- **Decision Trees**
  - What is a decision tree, and how to induce it from data
- **Fundamental Machine Learning Concepts**
  - Difference between memorization and generalization
  - What inductive bias is, and what is its role in learning.
  - What underfitting and overfitting means
  - How to take a task and cast it as a learning problem

Why you should never ever touch your test data!!

UNIVERSITY OF
MARYLAND

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu