Machine Learning

CS 539

Worcester Polytechnic Institute

Department of Computer Science

Instructor: Prof. Kyumin Lee

Project Teams

- Yu-Chi Liang, William Ryan, Riley Blair, and Stephen Fanning
- Chris Lee, Andrew Kerekon, Amulya Mohan, Alex Siracusa, and Sulaiman Moukheiber
- Vagmi Bhagavathula, Deepti Gosukonda, Adina Palayoor, Bishoy Soliman Hanna, and Jared Chan

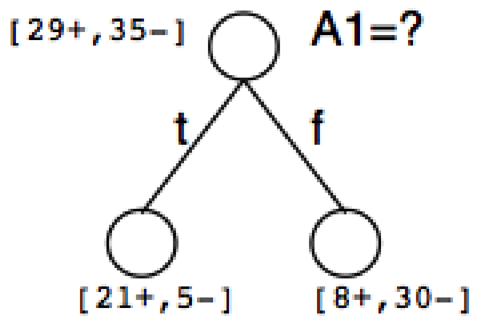
So far, 14 students formed teams.

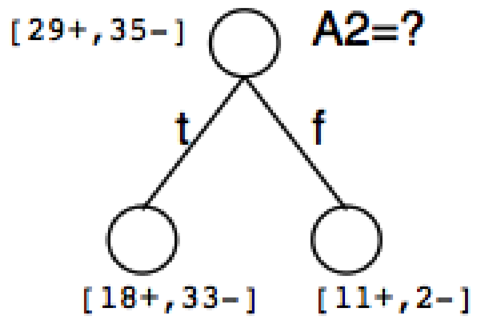
Email me names of your team members

Form a team by Jan 25

Decision Trees

Which attribute is the best classifier?





- A statistical property called *information gain*, measures how well a given attribute separates the training examples
- Information gain uses the notion of *entropy*, commonly used in information theory
- *Information gain = expected reduction of entropy*

Entropy

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *Y.*

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

- In a binary attribute,
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

$Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$ $[0 \log_2 0 = 0]$

$Entropy\ ([14+, 0-]) =$

$Entropy\ ([9+, 5-]) =$

$Entropy\ ([7+, 7-]) =$

$[\log_2 1/2 = -1]$

Entropy

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *Y.*

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

- In a binary attribute,
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

$Entropy\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$ $[0 \log_2 0 = 0]$

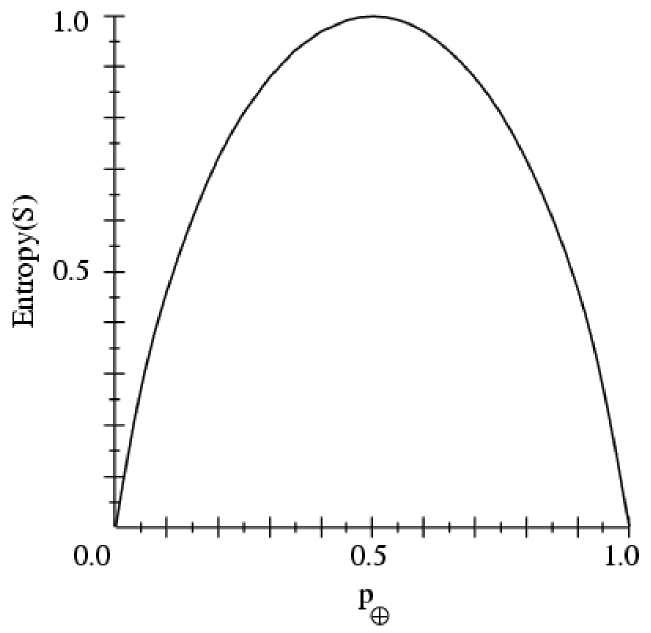
$Entropy\ ([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$

$Entropy\ ([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$

$Entropy\ ([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) =$

$= 1/2 + 1/2 = 1$ $[\log_2 1/2 = -1]$

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

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

Conditional entropy H(Y|A)

- Given attribute A, measure Y's conditional entropy:

  = The average entropy(children nodes)

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

$Values(A)$: possible values for A

S_v: subset of S for which A has value v

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing an attribute/feature A

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

$Values(A)$: possible values for A

S_v: subset of S for which A has value v

Information Gain = entropy(parent) – [average entropy(children)]

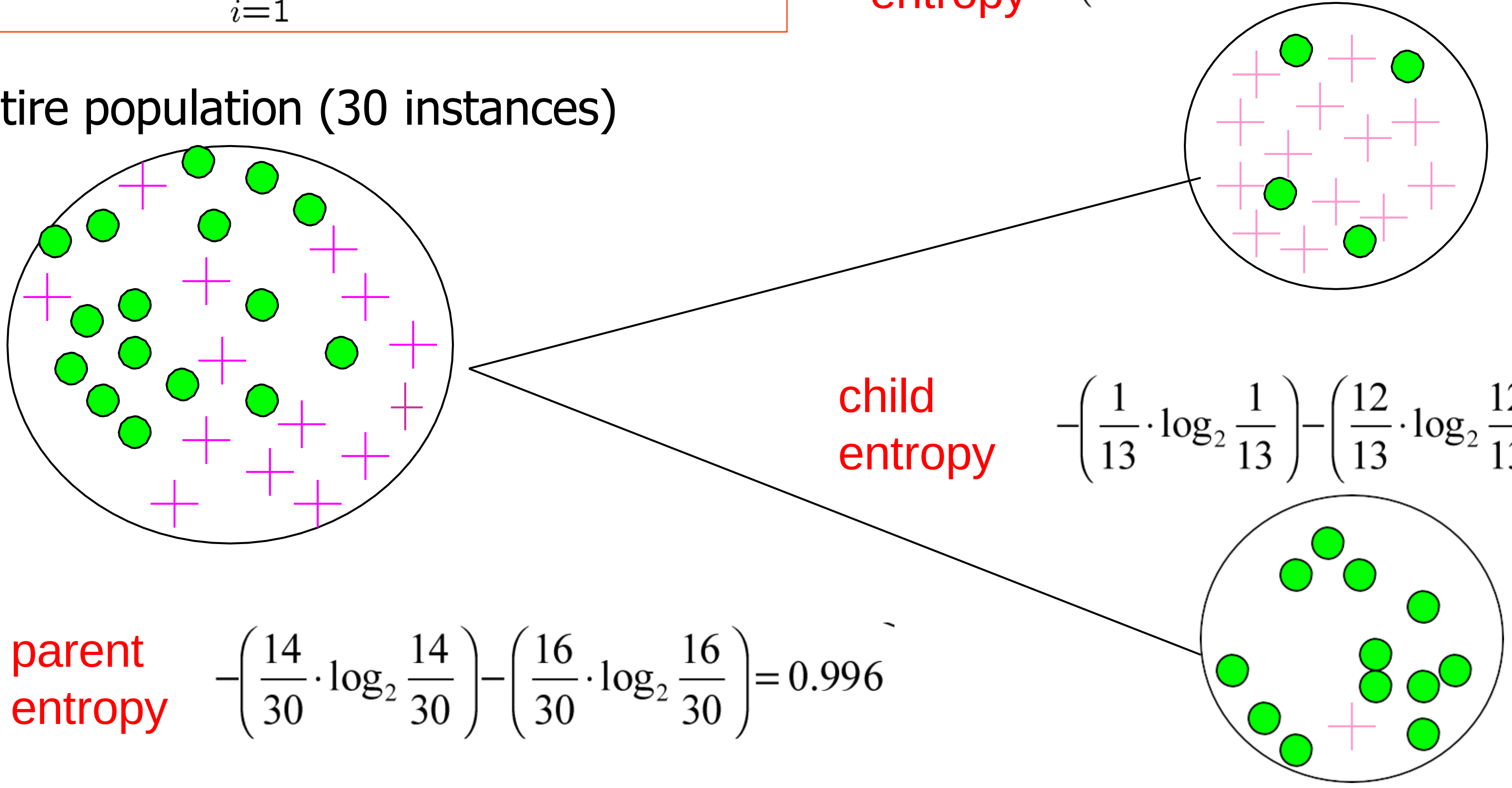
Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

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

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

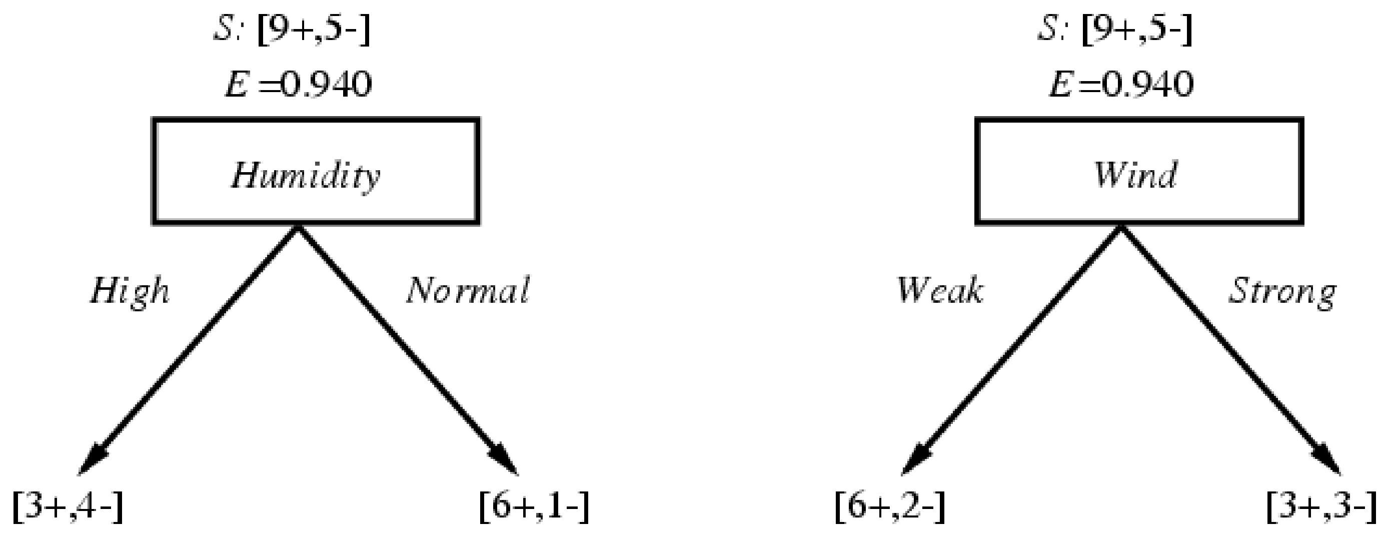
Information Gain= 0.996 - 0.615 = 0.38

Training Example

Day	Outlook	Temperature	Humidity	Wind	PlayTenr
Dl	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
DlO	Rain	Mild	Normal	Weak	Yes
Dll	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

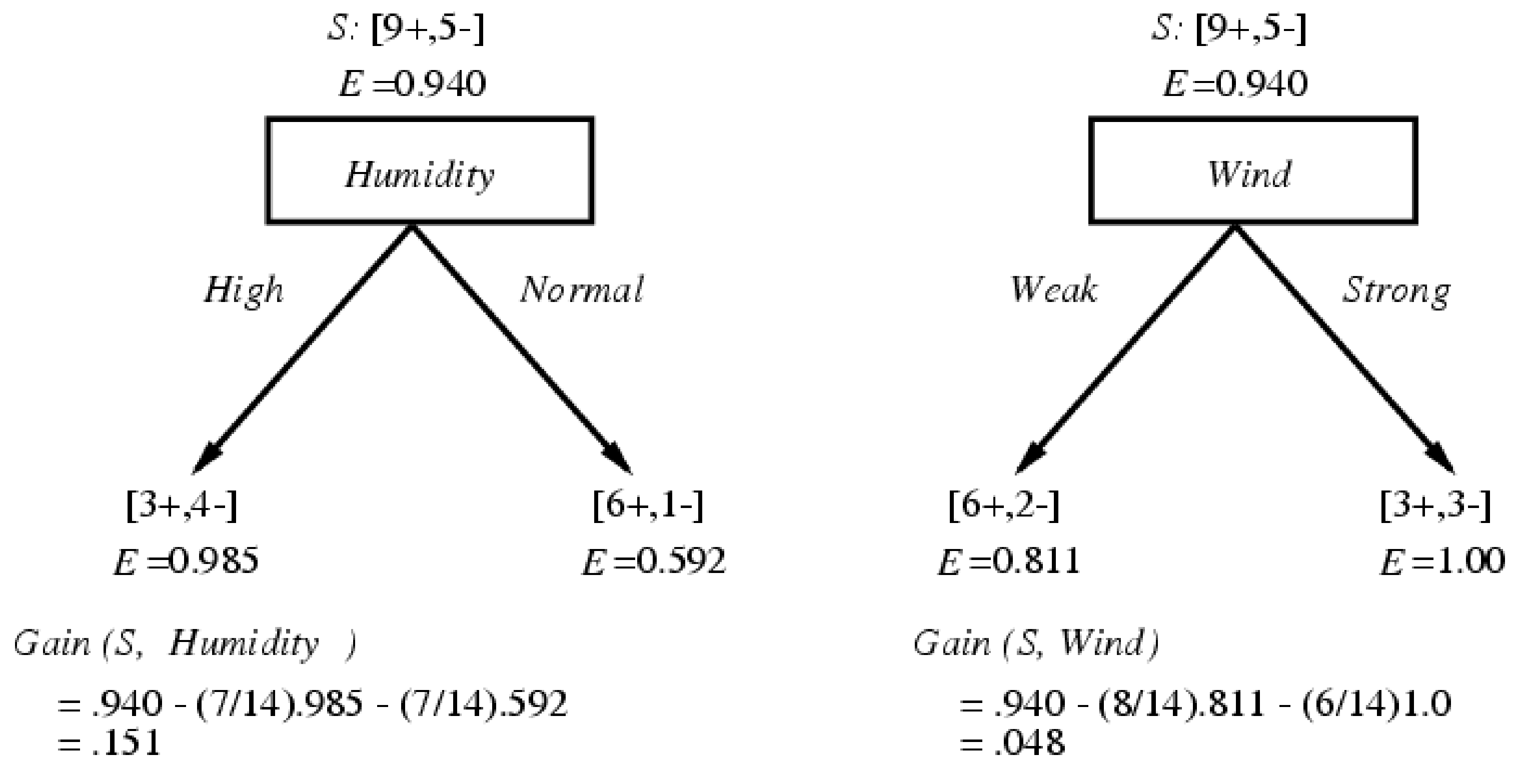
Selecting the Next Attribute

Which attribute is the best classifier?



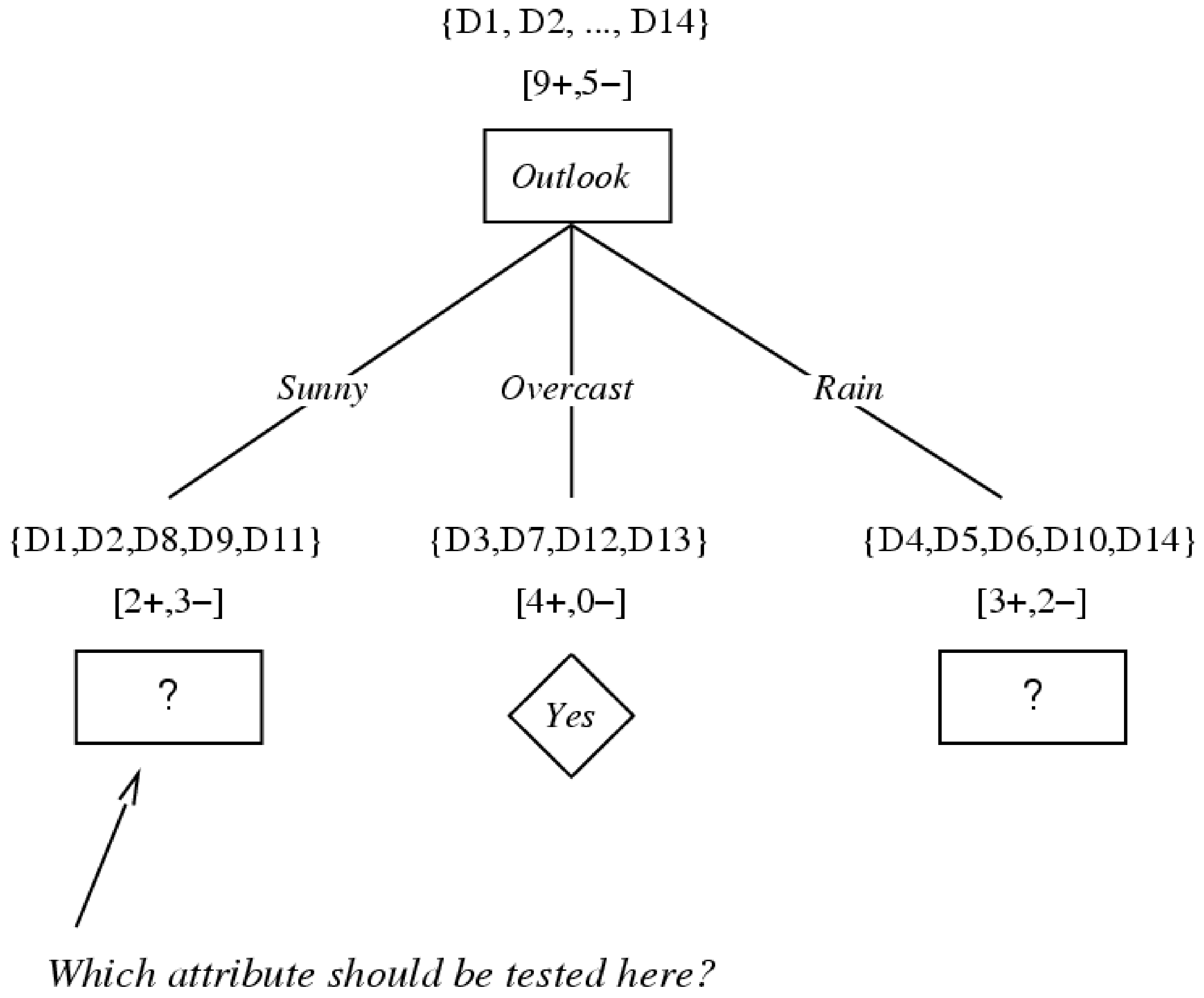
Selecting the Next Attribute

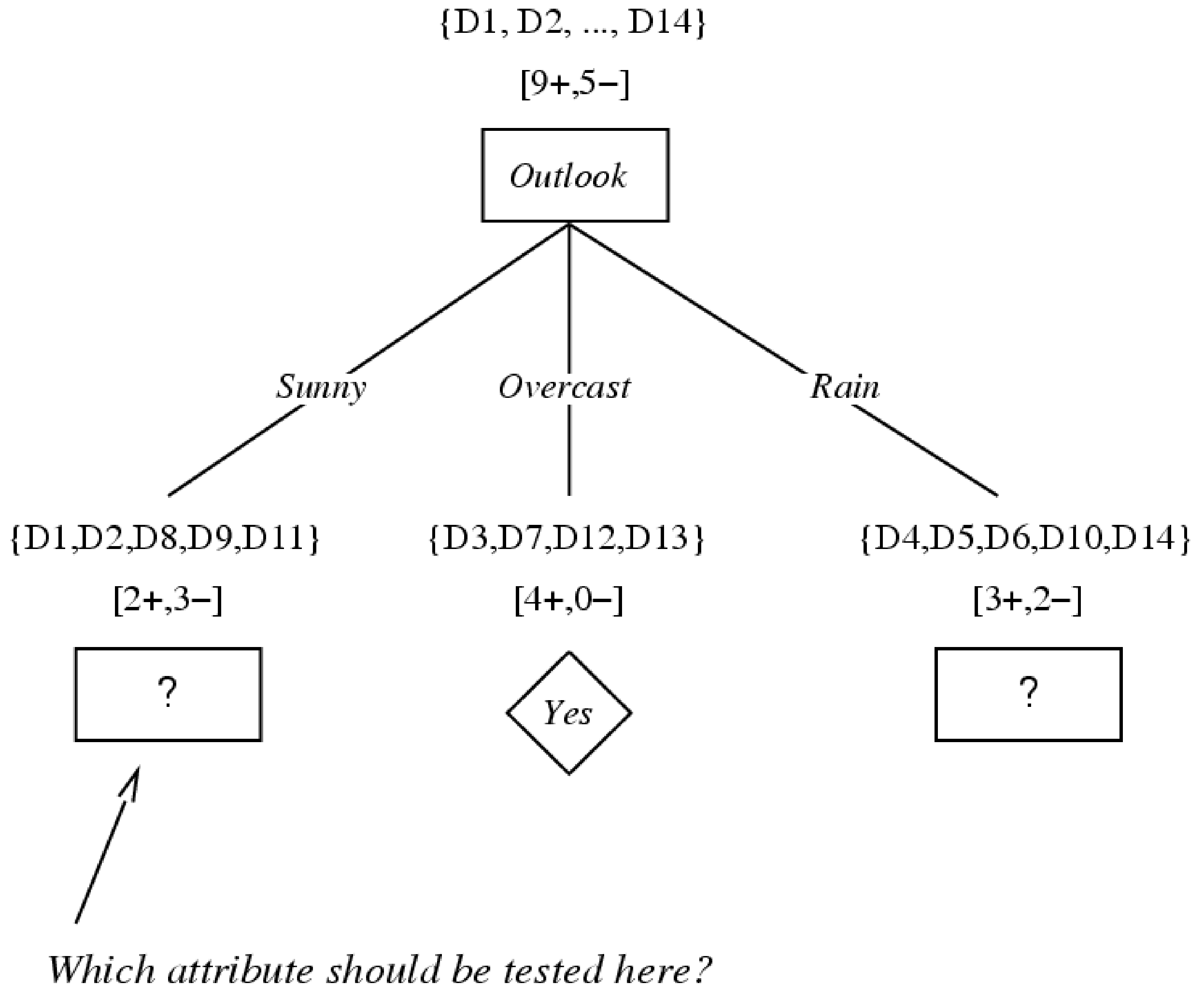
Which attribute is the best classifier?



First step: which attribute to test at the root?

- Which attribute should be tested at the root?
  - $Gain(S, Outlook) = 0.246$
  - $Gain(S, Humidity) = 0.151$
  - $Gain(S, Wind) = 0.084$
  - $Gain(S, Temperature) = 0.029$
- *Outlook* provides the best prediction for the target


{D1, D2, ..., D14}
[9+,5−]
Outlook
Sunny
Overcast
Rain
{D1,D2,D8,D9,D11}
[2+,3−]
{D3,D7,D12,D13}
[4+,0−]
{D4,D5,D6,D10,D14}
[3+,2−]
?
Yes
?
Which attribute should be tested here?



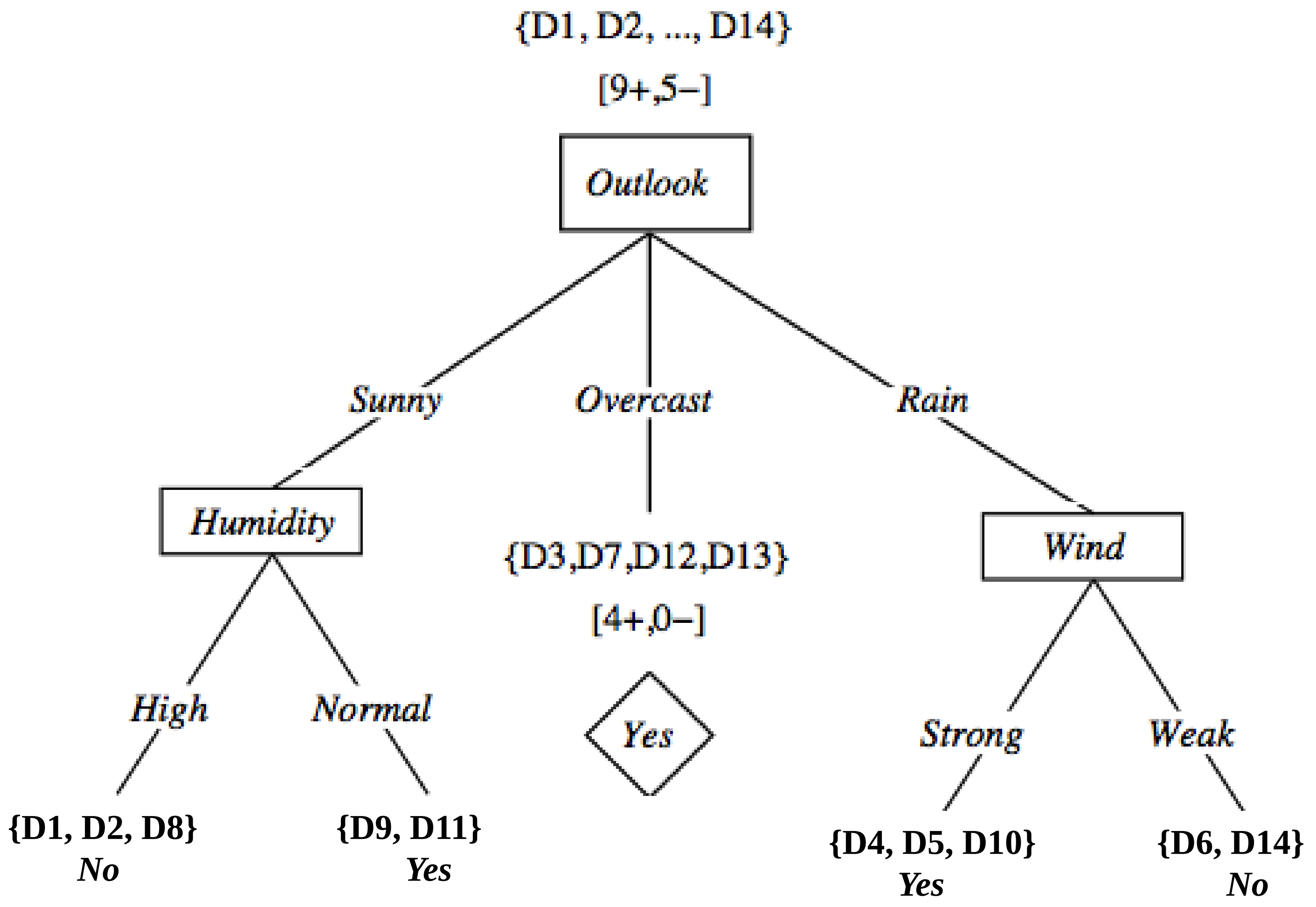
$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Second and third steps

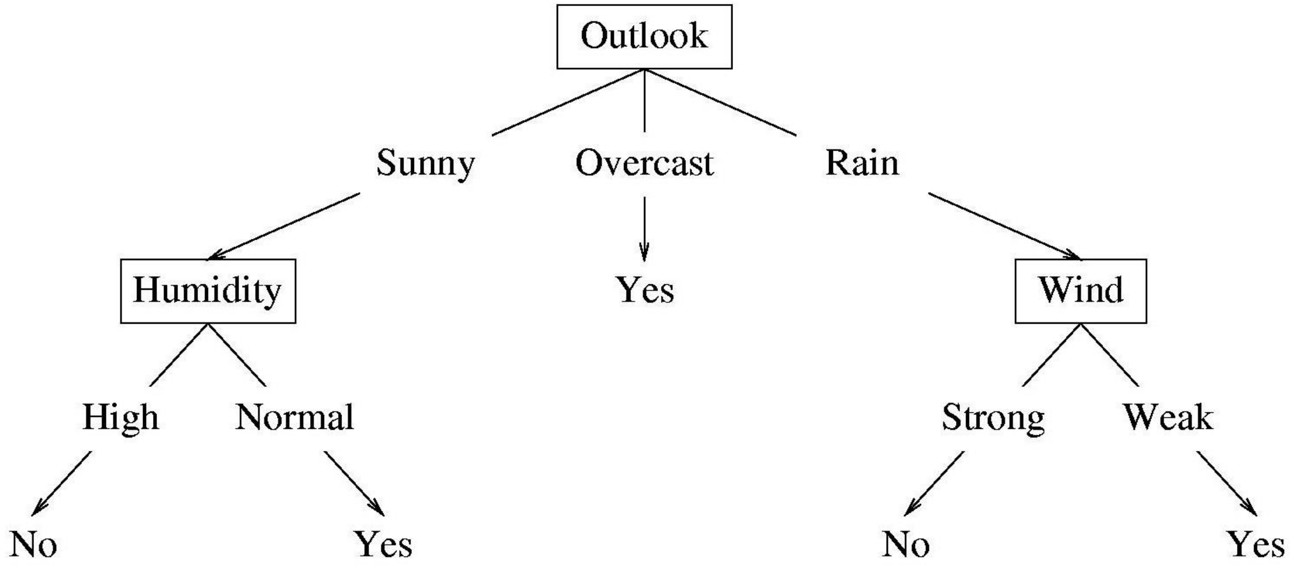


ID3: algorithm

```
ID3($X, T, Attrs$)    $X$: training examples:
                      $T$: target attribute (e.g. $PlayTennis$),
                      $Attrs$: other attributes, initially all attributes
 Create Root node
 If all X's are +, return Root with class +
 If all X's are –, return Root with class –
 If Attrs is empty return Root with class most common value of $T$ in $X$
 else
    $A \leftarrow$ best attribute; decision attribute for Root $\leftarrow A$
    For each possible value $v_i$ of $A$:
     - add a new branch below Root, for test $A = v_i$
     - $X_i \leftarrow$ subset of $X$ with $A = v_i$
     - If $X_i$ is empty then add a new leaf with class the most common value of $T$ in $X$
          else add the subtree generated by ID3($X_i$, $T$, $Attrs - \{A\}$)
 return Root
```

Refer to https://en.wikipedia.org/wiki/ID3_algorithm

Summary of Decision Trees (so far)



- Decision tree induction -> choose the best attribute
  - Choose split via information gain
  - Build tree greedily, recursing on children of split
  - Stop when we achieve homogeny
    - i.e., when all instance in a child have the same class

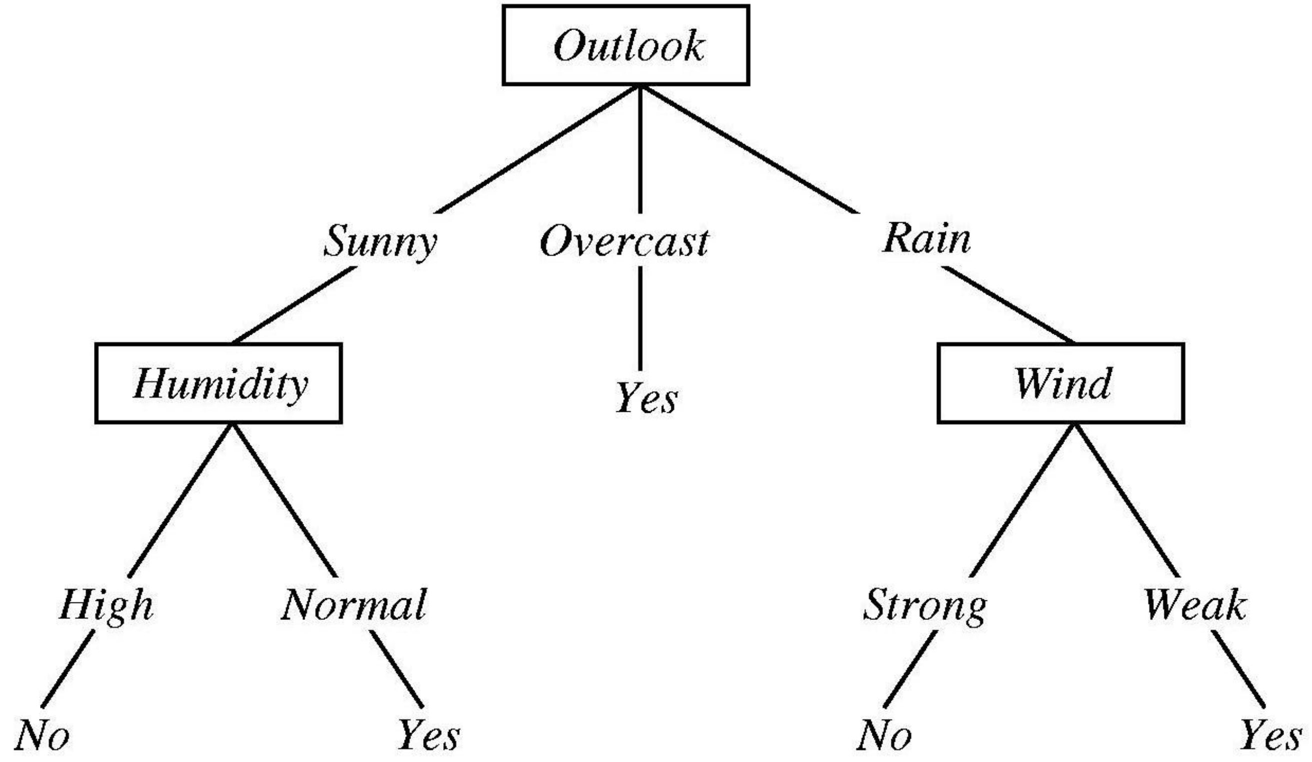
Overfitting

- Irrelevant attributes can result in *overfitting* the training example data
  - If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features

- If we have too little training data, even a reasonable hypothesis space will ‘overfit’

Overfitting in Decision Trees

Consider adding a noisy training example to the following tree:



What would be the effect of adding:

<outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No> ?

=> New noisy example causes splitting of second leaf node.

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

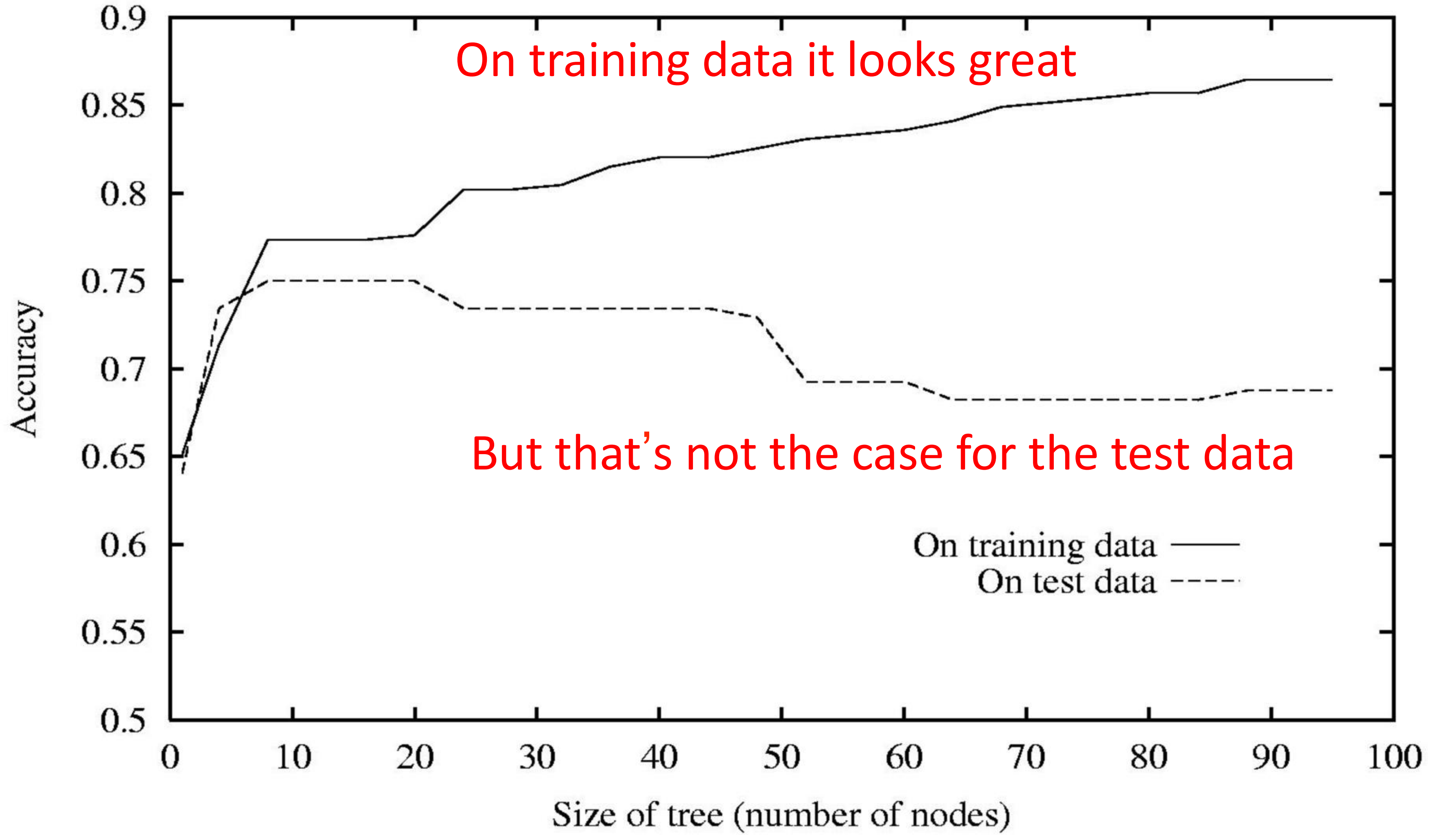
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

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

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

- How can we avoid overfitting?
  - Stop growing when data split is not statistically significant (e.g., chi-squared test)
  - Acquire more training data
  - Allow the tree to *overfit* the data (i.e., grow full tree), and then *post-prune* the tree

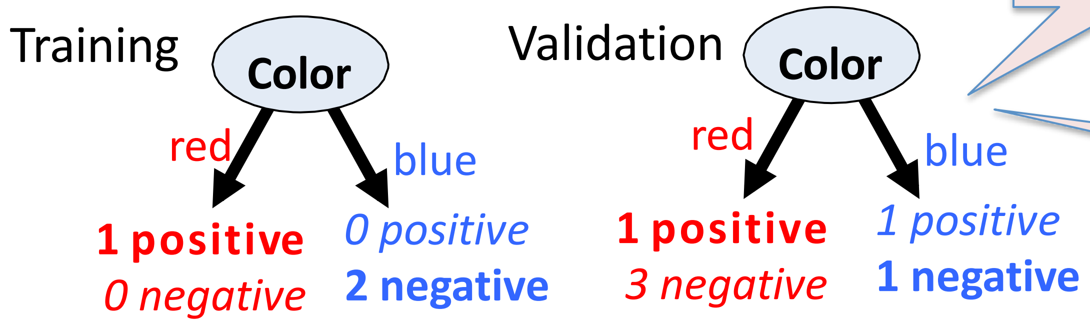
- Training and validation set
  - Split the training in two parts (training and validation) and use validation to assess the utility of post-pruning
    - Reduced error pruning

Reduced-Error Pruning

- Each node is a candidate for pruning
- *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
- Nodes are removed only if the resulting tree performs no worse on the validation set.
- Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.
- Pruning stops when no pruning increases accuracy

Pruning Decision Trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
- For example,

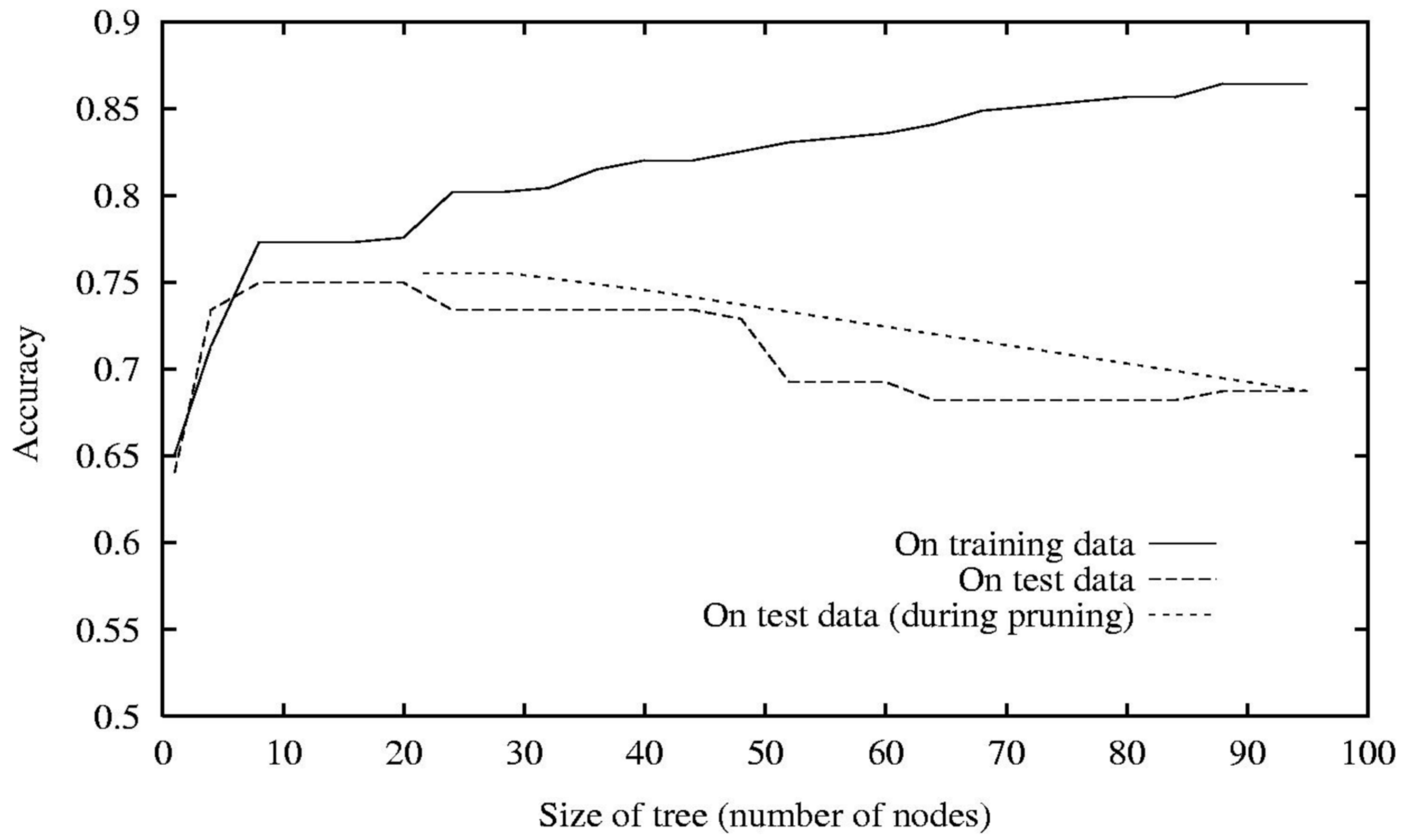


2 correct
4 incorrect

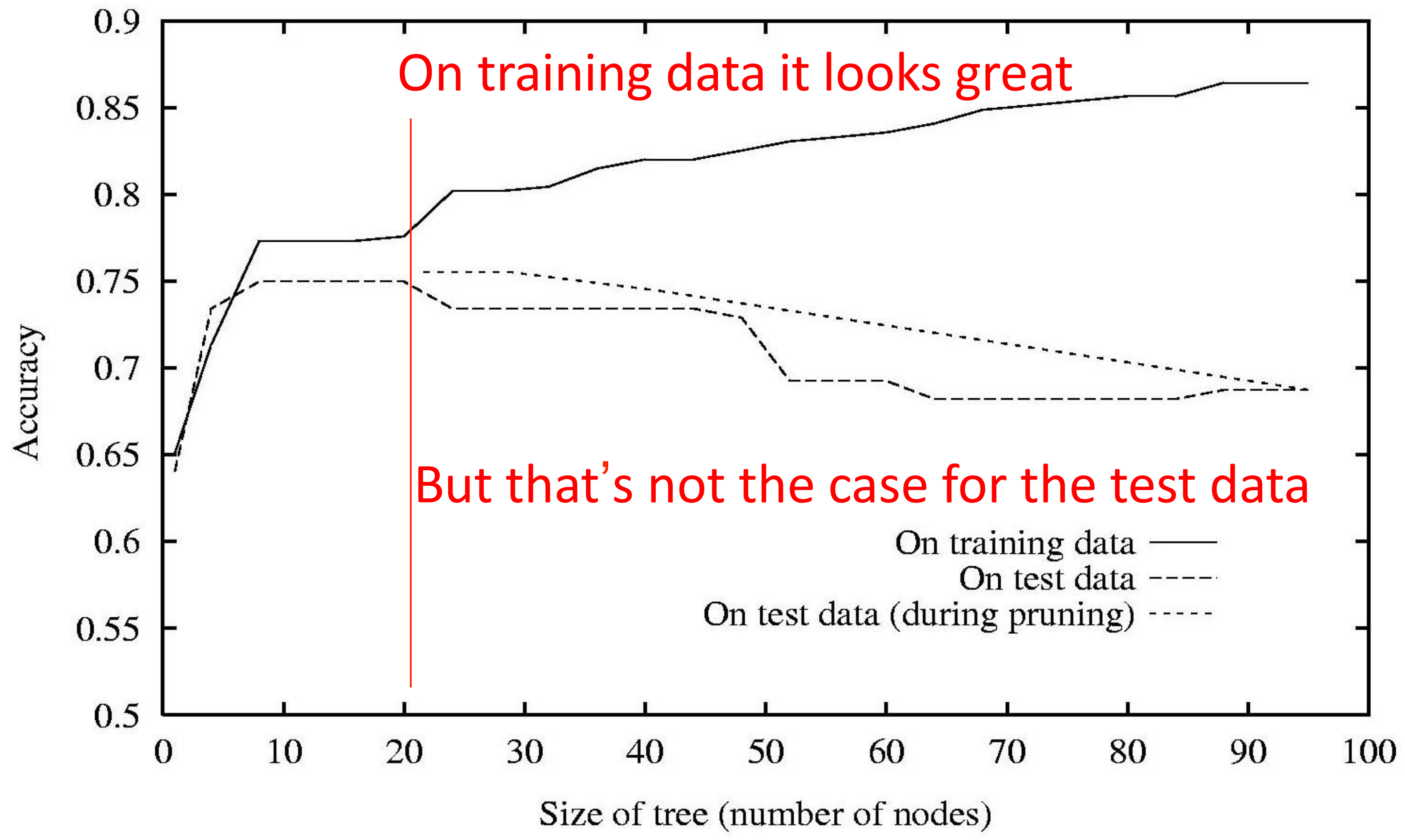
If we had simply predicted the majority class (negative), we make 2 errors instead of 4.

Pruned!

Effect of Reduced-Error Pruning



Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Summary: Decision Tree Learning

• Representation:	decision trees
• Bias:	prefer small decision trees
• Search algorithm:	greedy
• Heuristic function:	information gain
• Overfitting / pruning	

Evaluation

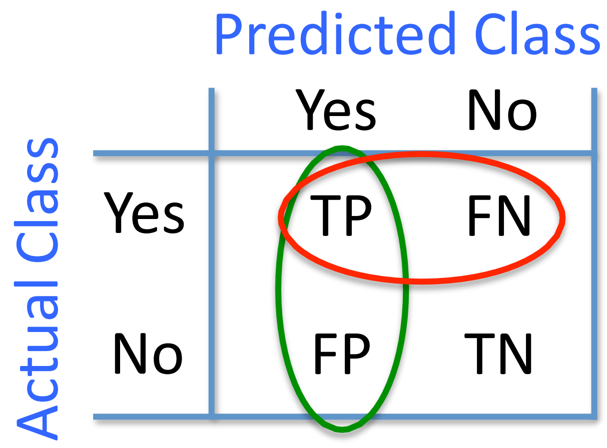
Classification Metrics

$$\text{accuracy} = \frac{\#\ \text{correct predictions}}{\#\ \text{test instances}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\#\ \text{incorrect predictions}}{\#\ \text{test instances}}$$

Confusion Matrix

- Given a dataset of *P* positive instances and *N* negative instances:



		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

$$\text{accuracy} = \frac{TP + TN}{P + N}$$

- Imagine using classifier to identify positive cases (i.e., for information retrieval)

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

Probability that a randomly selected result is relevant

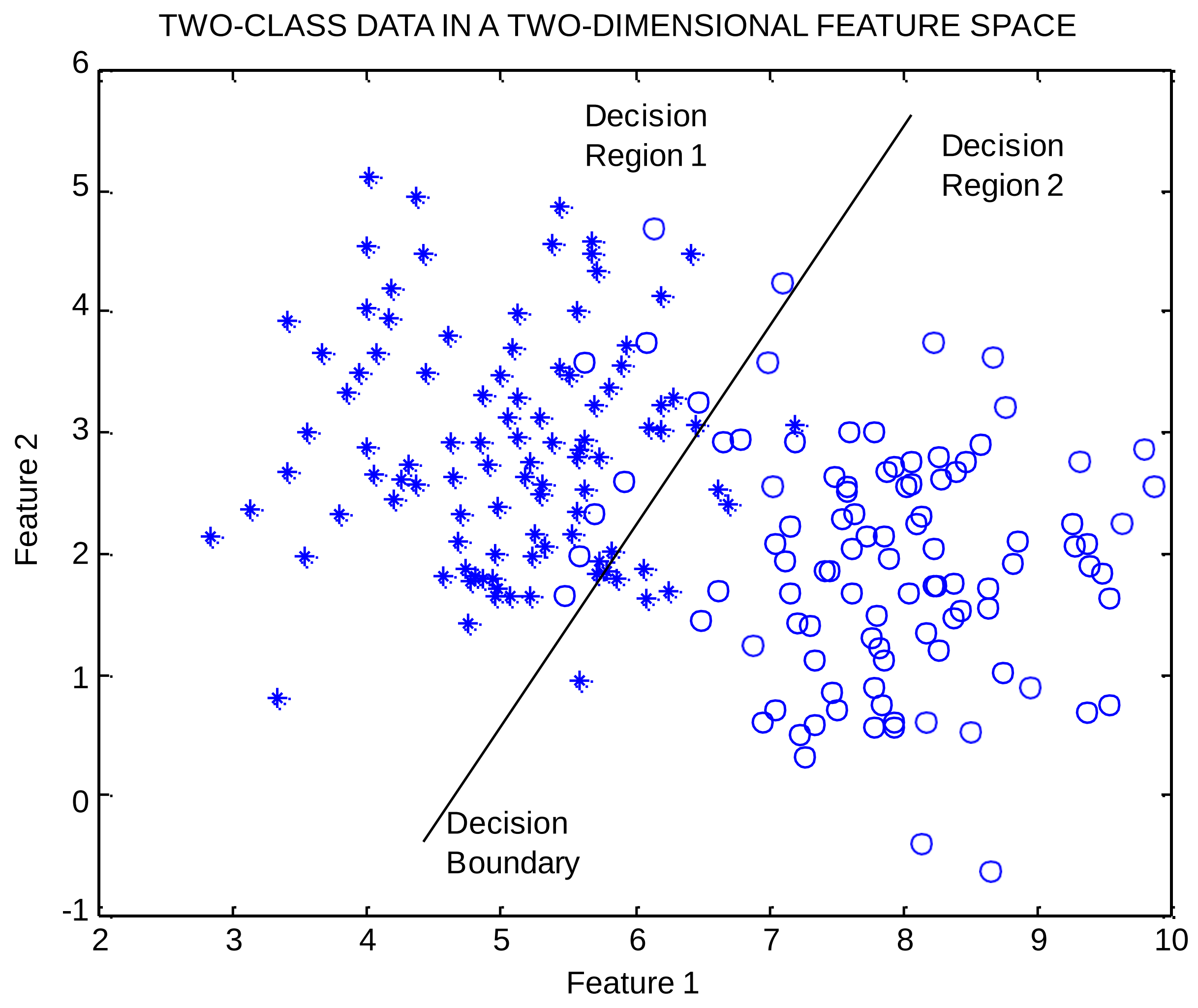
Probability that a randomly selected relevant document is retrieved

Training Data and Test Data

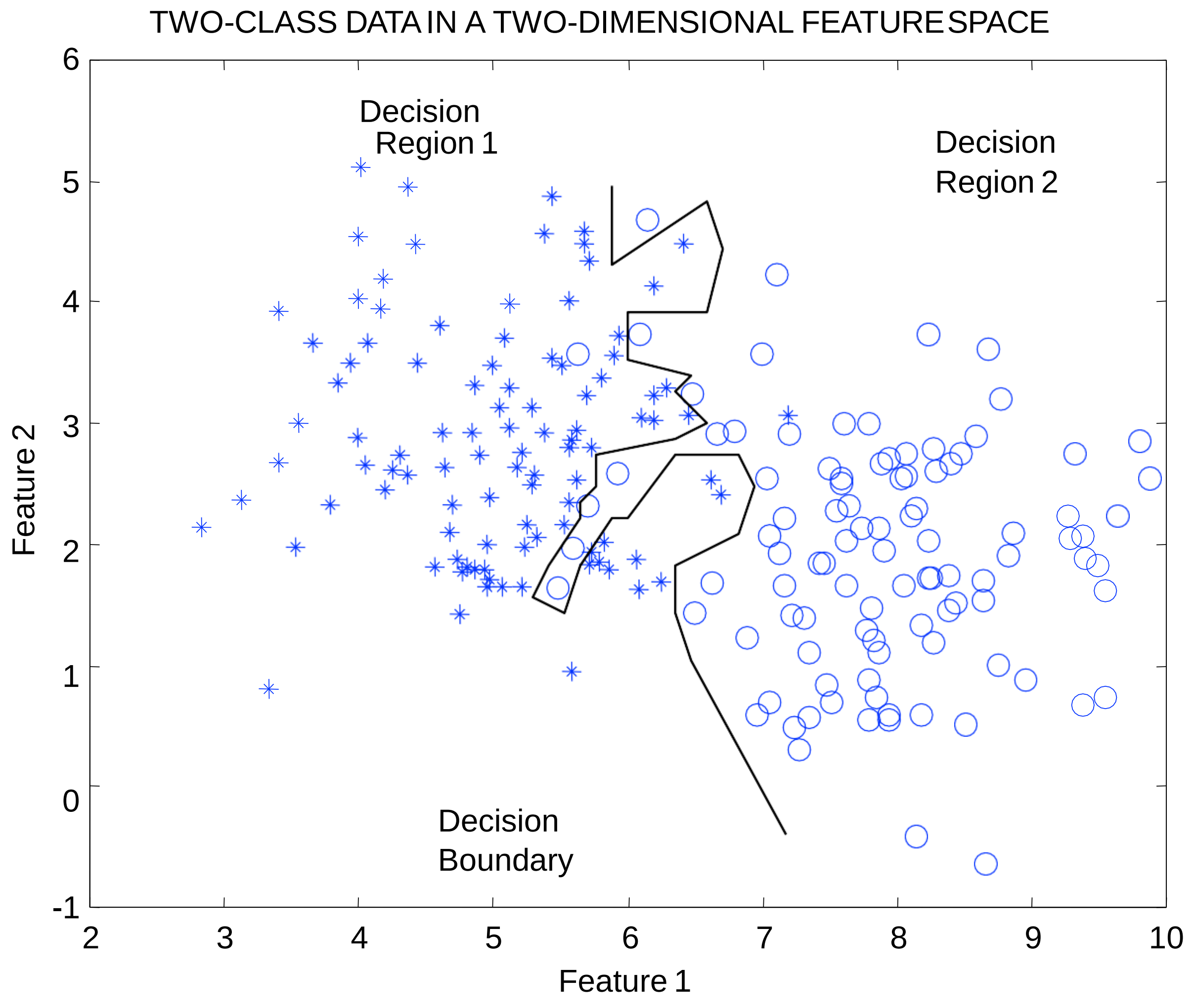
- Training data: data used to build the model
- Test data: new data, not used in the training process

- Training performance is often a poor indicator of generalization performance
  - Generalization is what we <u>really</u> care about in ML
  - Easy to overfit the training data
  - Performance on test data is a good indicator of generalization performance
  - i.e., test accuracy is more important than training accuracy

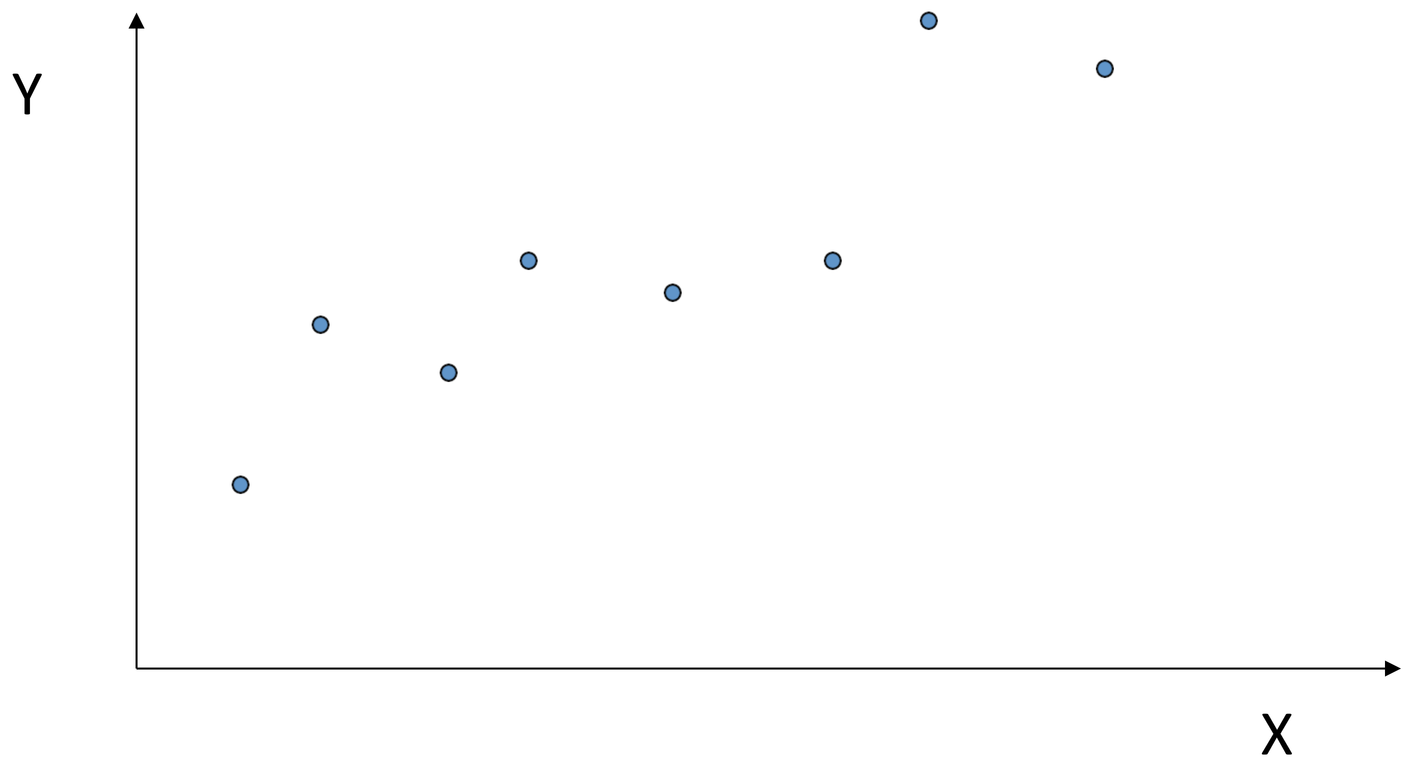
Simple Decision Boundary



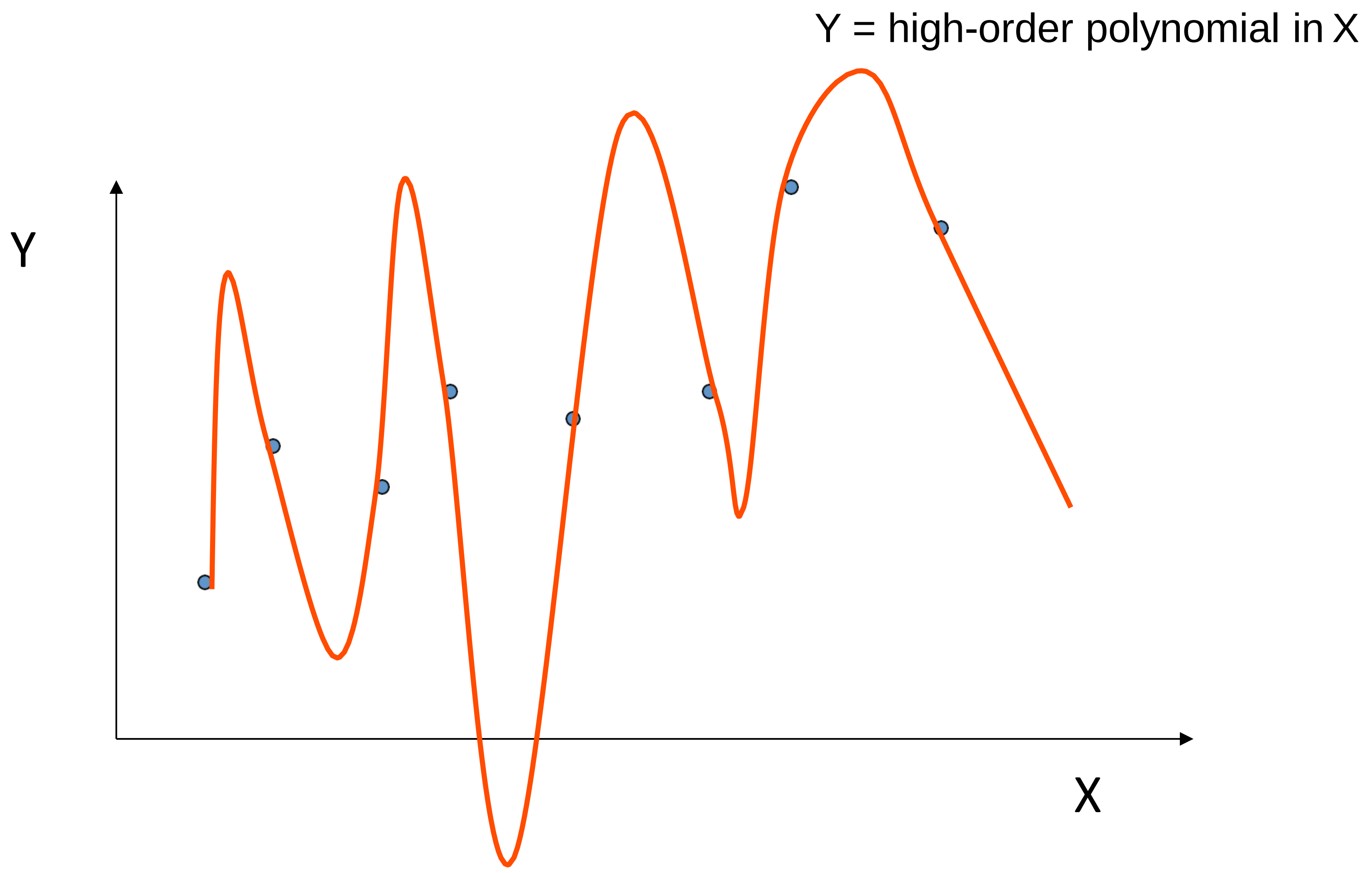
More Complex Decision Boundary



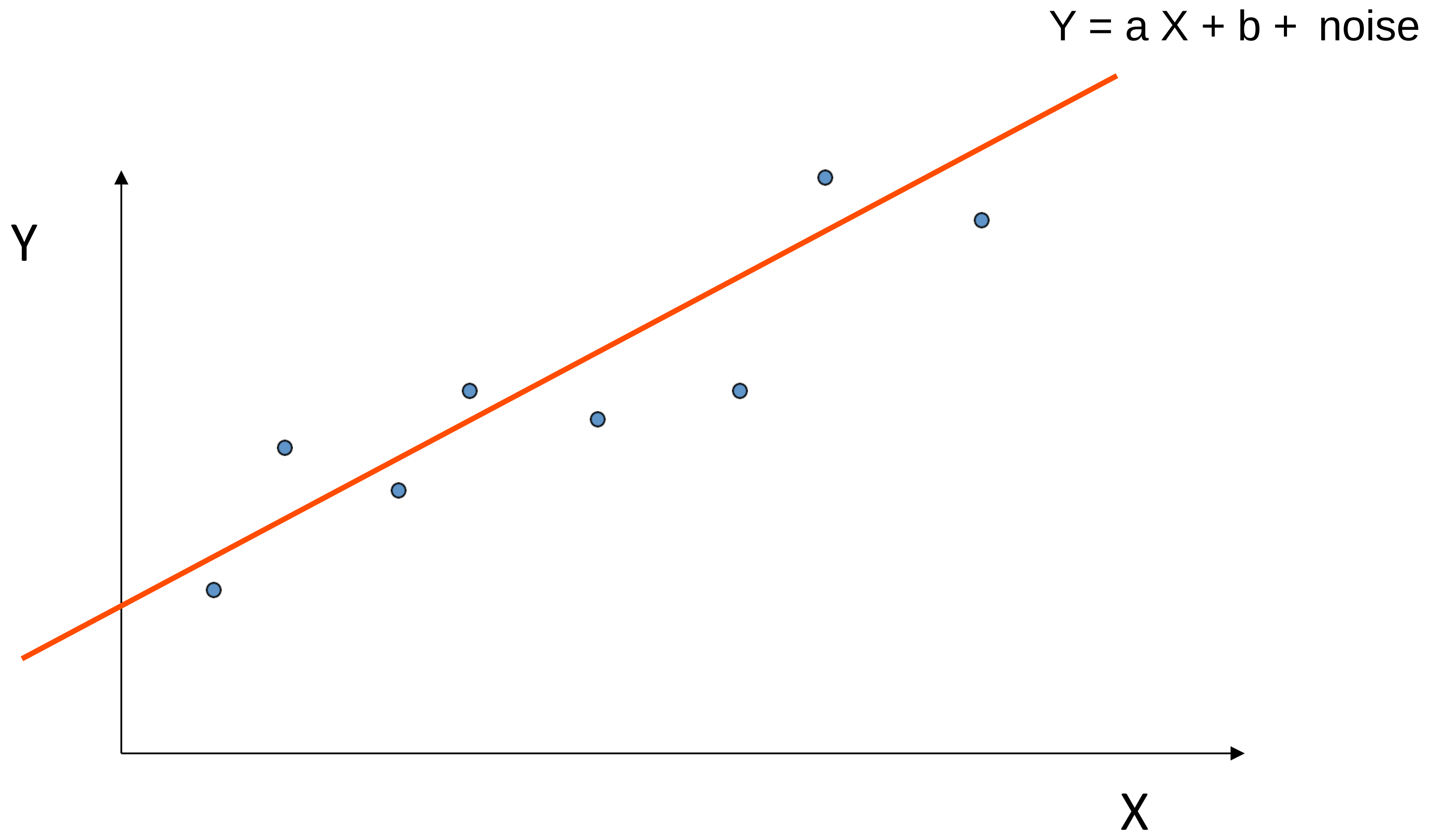
Example: The Overfitting Phenomenon



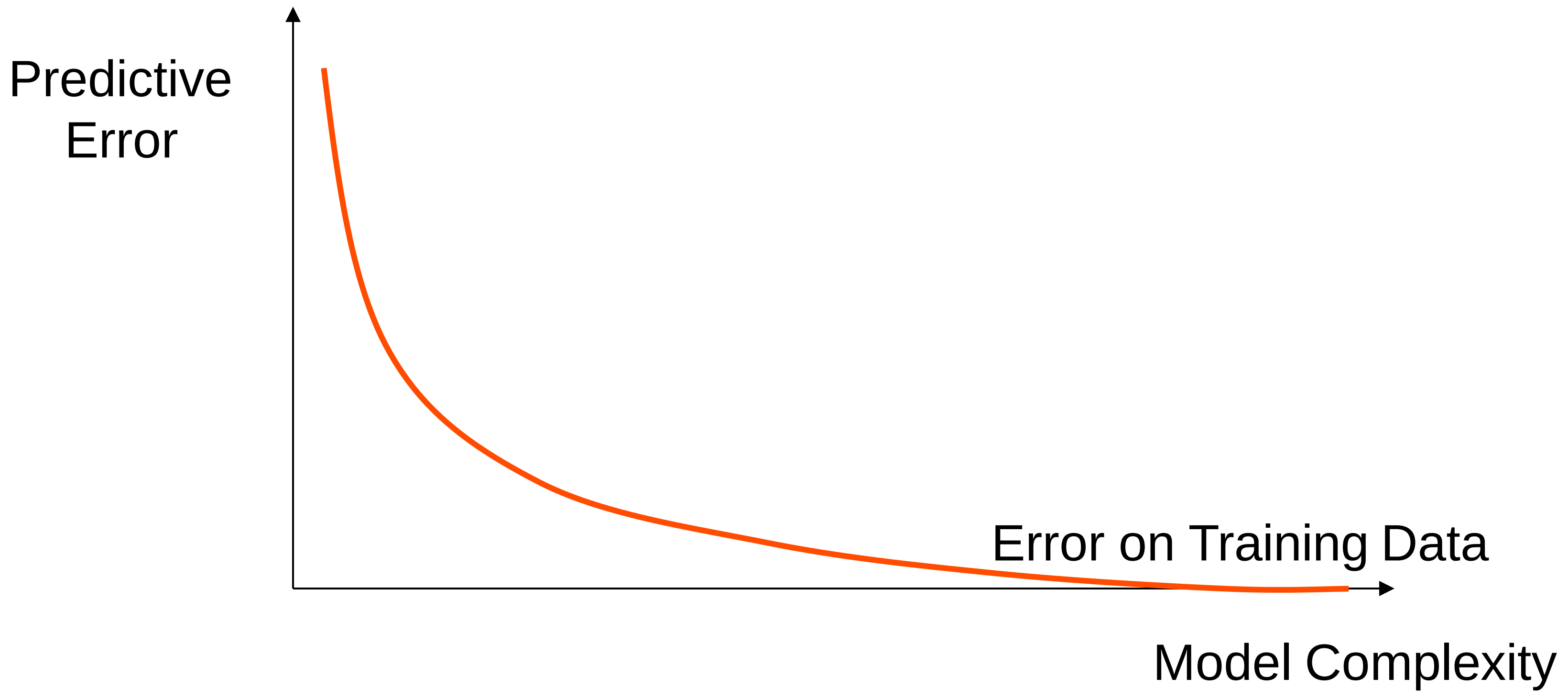
A Complex Model



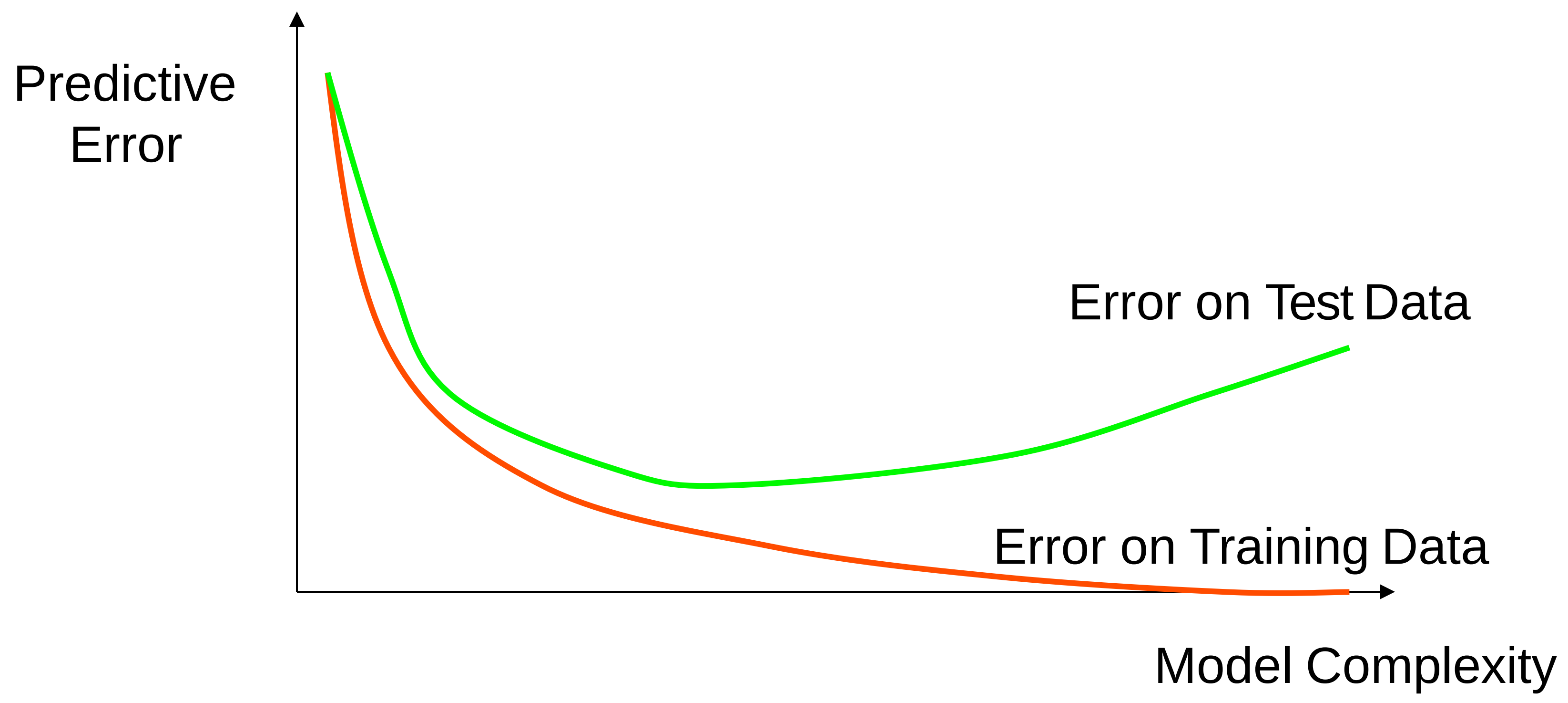
The True (simpler) Model



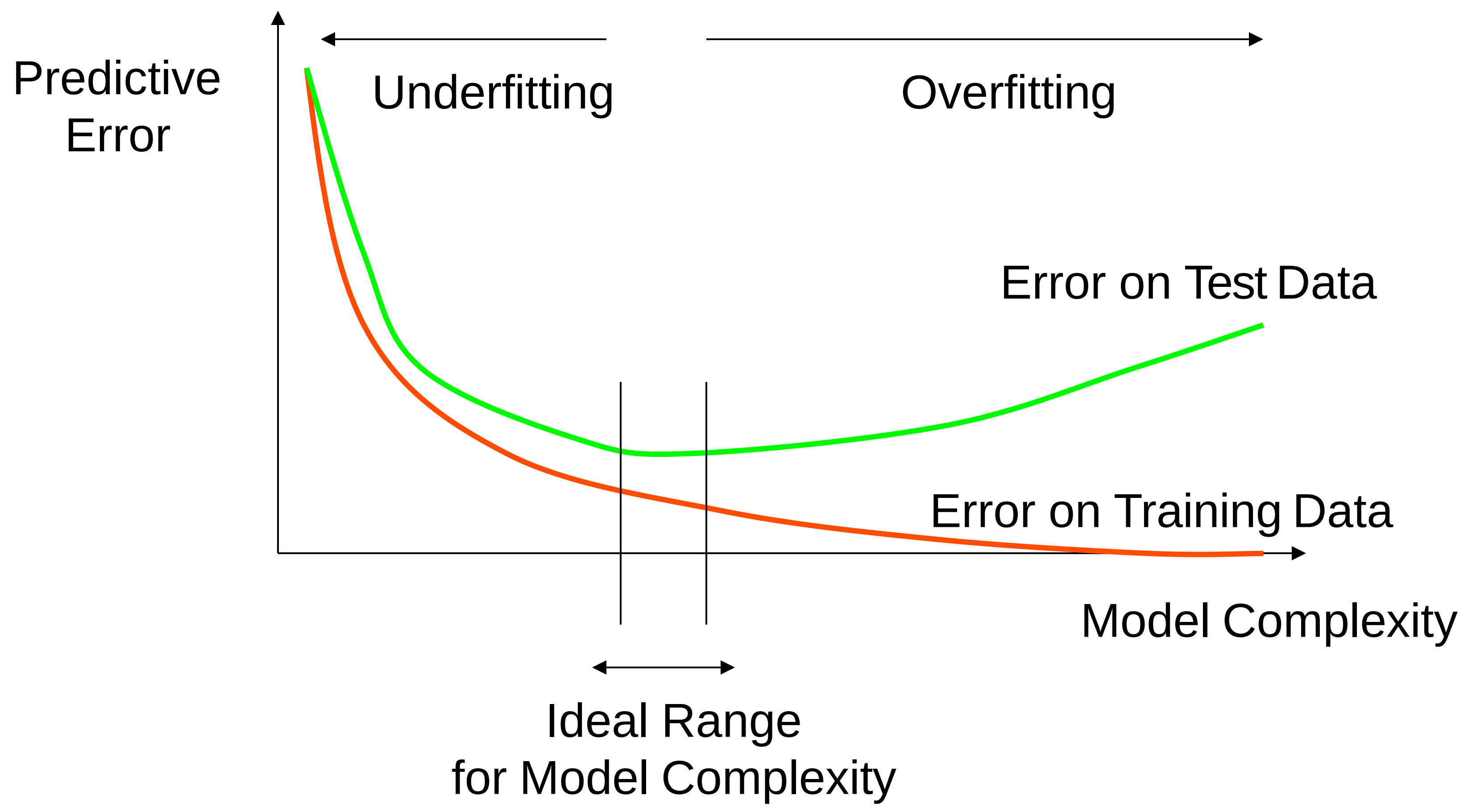
How Overfitting Affects Prediction



How Overfitting Affects Prediction



How Overfitting Affects Prediction



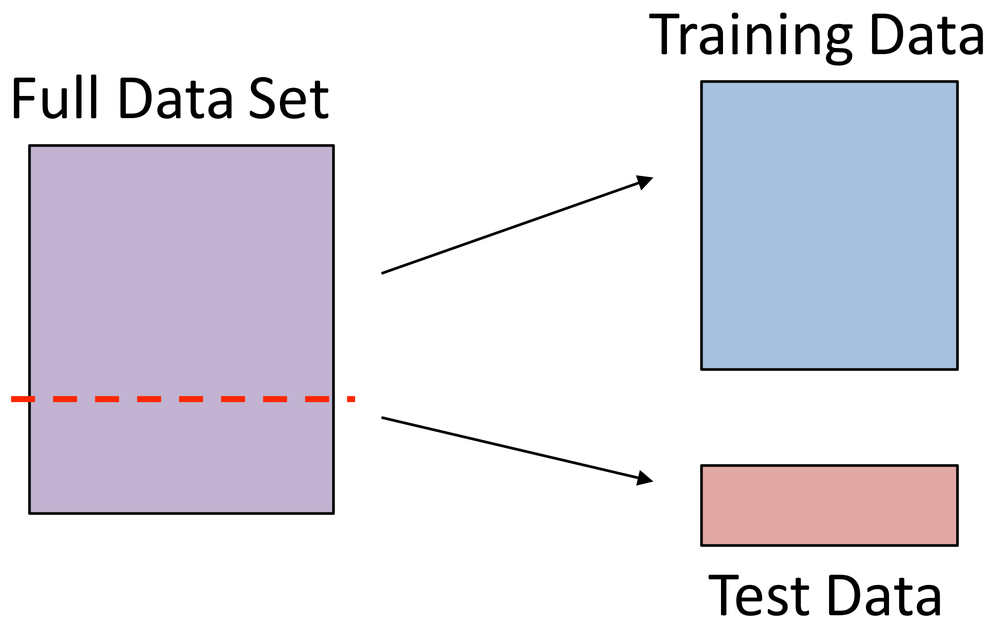
Comparing Classifiers

- Say we have two classifiers, *C1* and *C2*, and want to choose the best one to use for future predictions

- Can we use training accuracy to choose between them?
- No!

- Instead, choose based on test accuracy...

Training and Test Data



Idea:
Train each model on the "training data"...

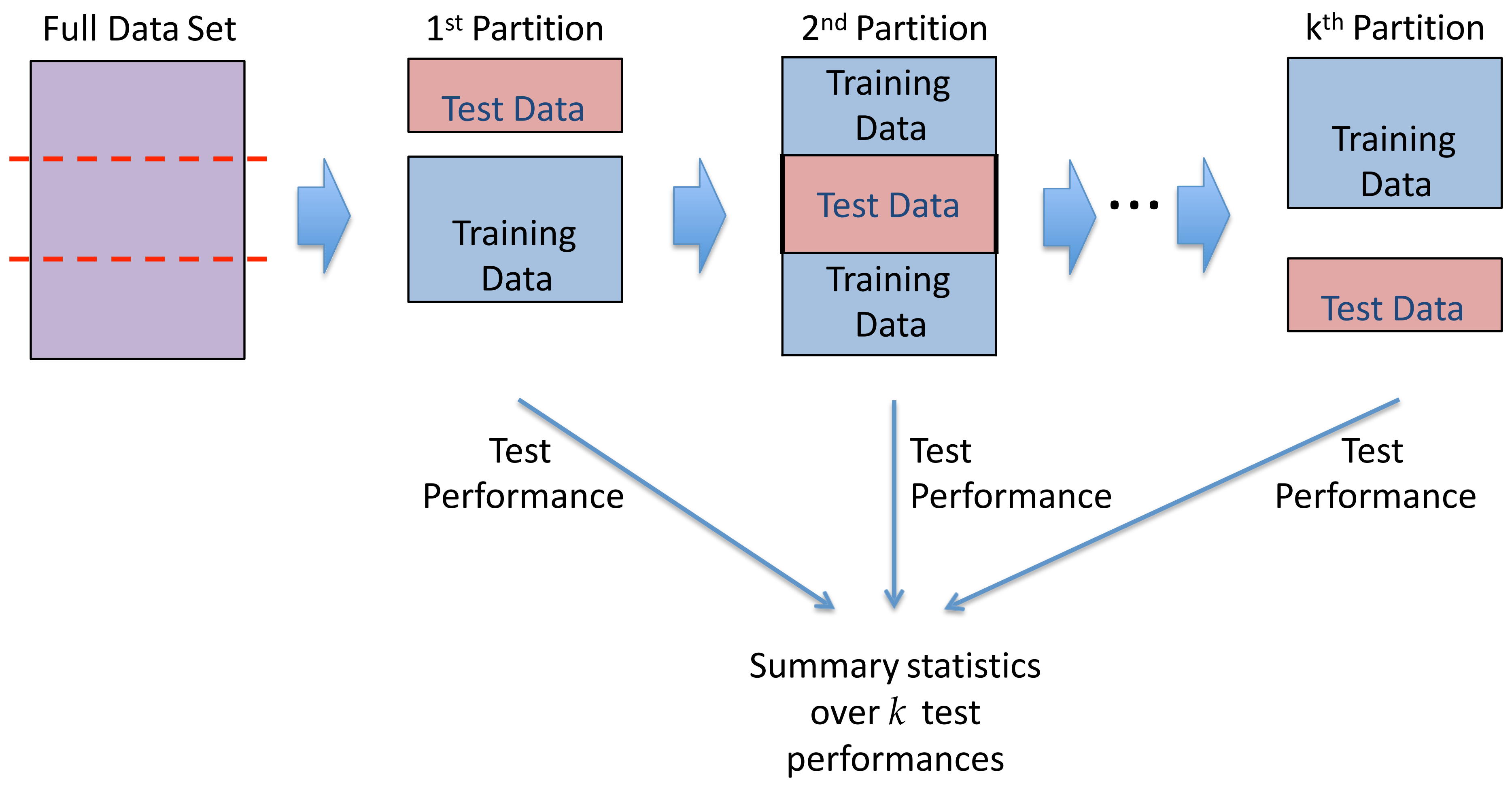
...and then test each model's accuracy on the test data

k-Fold Cross-Validation

- Why just choose one particular “split” of the data?
  - In principle, we should do this multiple times since performance may be different for each split

- *k*-Fold Cross-Validation (e.g., *k*=10)
  - randomly partition full data set of *n* instances into <u>*k* disjoint subsets</u> (each roughly of size *n*/*k*)
  - Choose each fold in turn as the test set; train model on the other folds and evaluate
  - Compute statistics over *k* test performances, or choose best of the *k* models

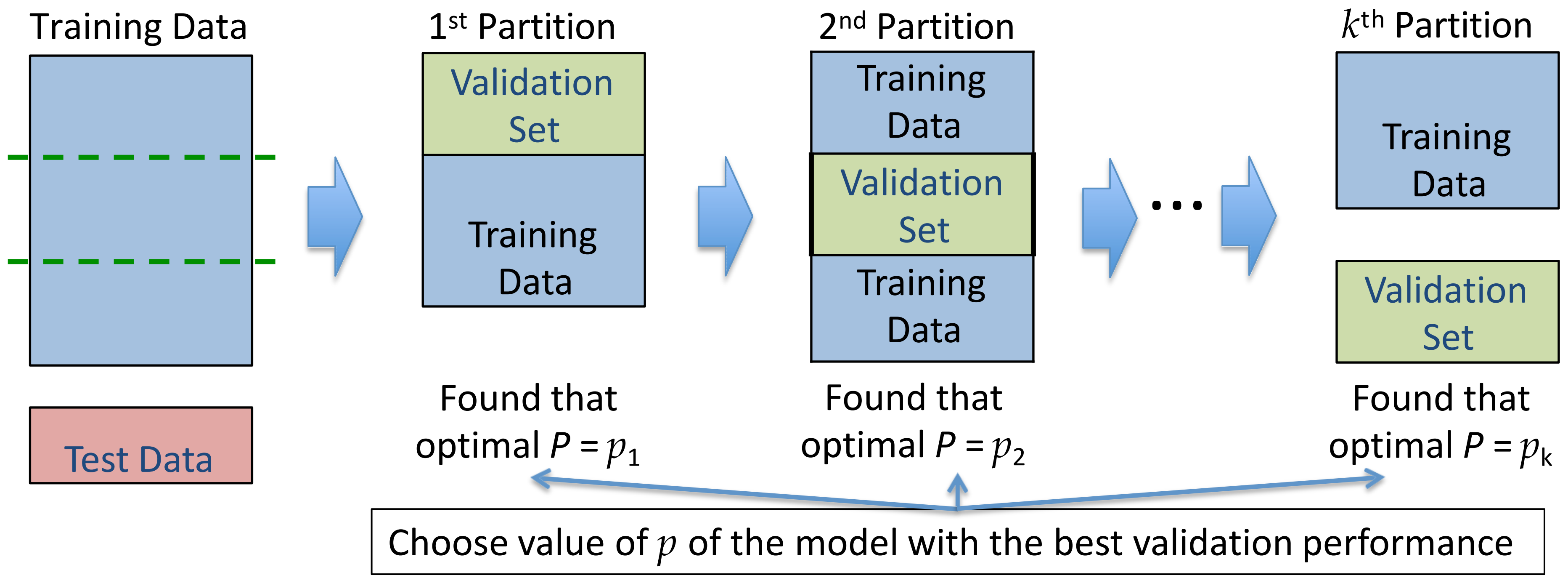
Example 3-Fold CV



Optimizing Model Parameters

Can also use CV to choose value of model parameter P

- Search over space of parameter values $p \in values(P)$
  - Evaluate model with $P = p$ on validation set
- Choose value p' with highest validation performance
- Learn model on full training set with $P = p'$



HW1

- https://canvas.wpi.edu/courses/57384/assignments/338207

- Due date is January 26th 11:59pm.