CS60050
Machine Learning

Decision Trees: Overfitting and Pruning

Somak Aditya
Assistant Professor
Department of CSE
August 14, 2024

Sudeshna Sarkar
Professor
CoE AI, Department of CSE

Are decision trees algorithms optimal?

- Well, what do we mean by optimal?
- Considering all possible decision trees (i.e., trees splitting on one feature per node),
- will the ID3 algorithm (each split maximizes mutual information; stopping when mutual information is zero)…
- produce the smallest decision tree that has lowest classification training error?
- No, they aren't optimal
- Decision trees are greedy algorithms, i.e., they make the best local decision without considering longer term possibilities.

▪ Better trees are possible, but it takes too long to search all combinations

Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features

Cons

- Greedy: each split only considers the immediate impact
  - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
- ***Liable to overfit!***

Slide credit: CMU MLD Henry Chai

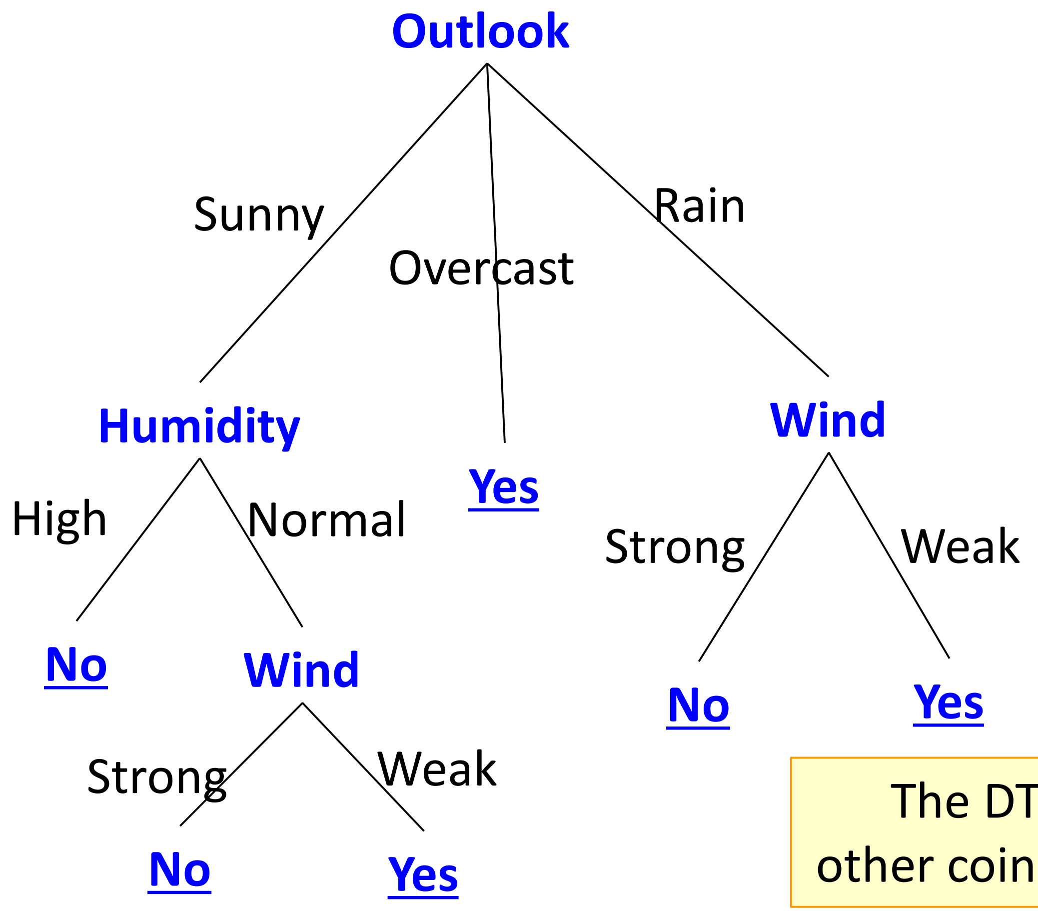
Overfitting in Decision Trees

- Many kinds of “noise” can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
  - The instance was labeled incorrectly (+ instead of -)

- Also, some attributes are irrelevant to the decision making process
  - e.g., color of a die is irrelevant to its outcome

Overfitting - Example

Consider adding a **noisy** training example to the following tree:

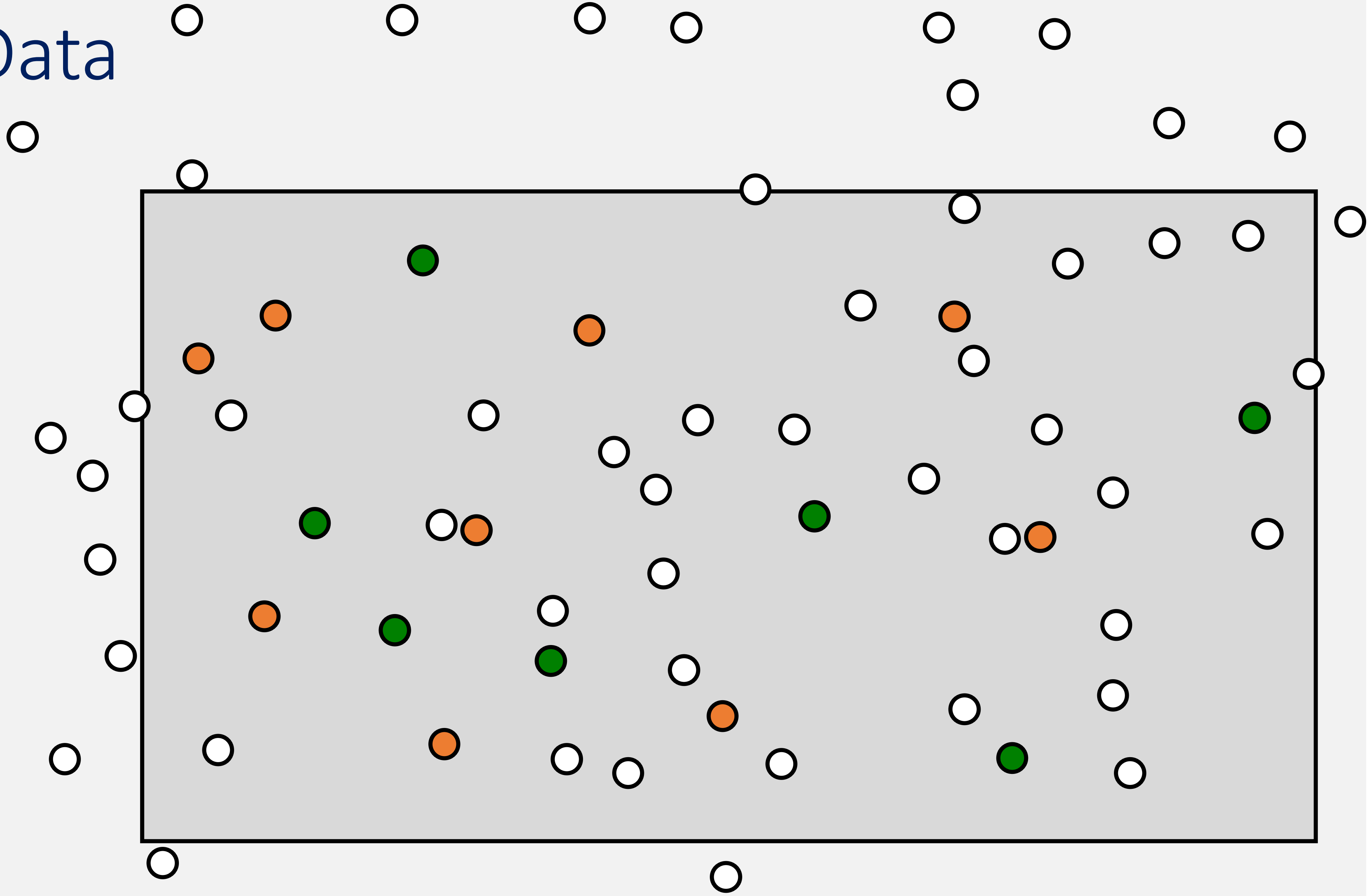


The DT may fit noise or other coincidental regularities

What would be the effect of adding:
<Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, playTennis=NO>

Training Data

Is (often) only a small set of the entire "instance space"



Error Rate

Consider a hypothesis h over

- error over all training data: $\text{error}(h, D_{train})$
- error rate over all test data: $\text{error}(h, D_{test})$
- true error over all data: $\text{error}_{\text{true}}(h, D)$

This is the quantity we care most about! But, in practice, $\text{error}_{\text{true}}(h, D)$ is **unknown.**

Learning a tree that classifies the training data perfectly may not lead to the tree with the *best generalization performance*.

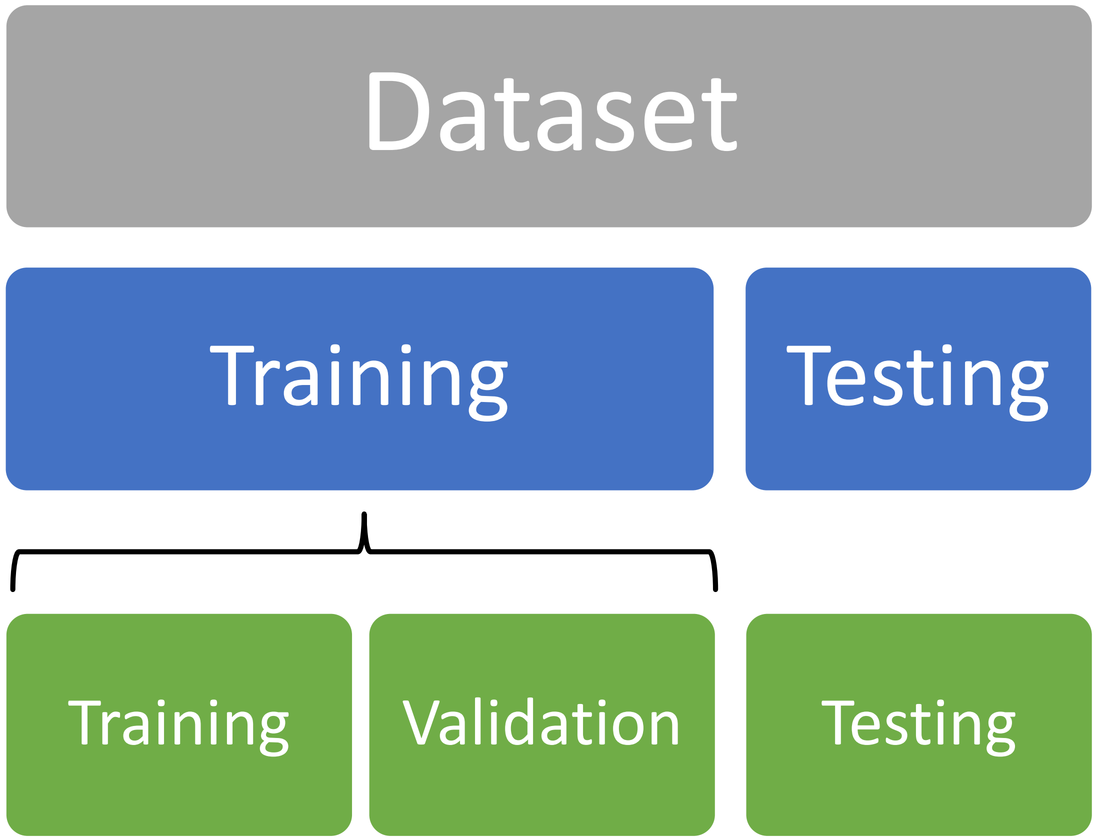
- Noise in the training data
- Very little data

Experimental Machine Learning

Split your data :

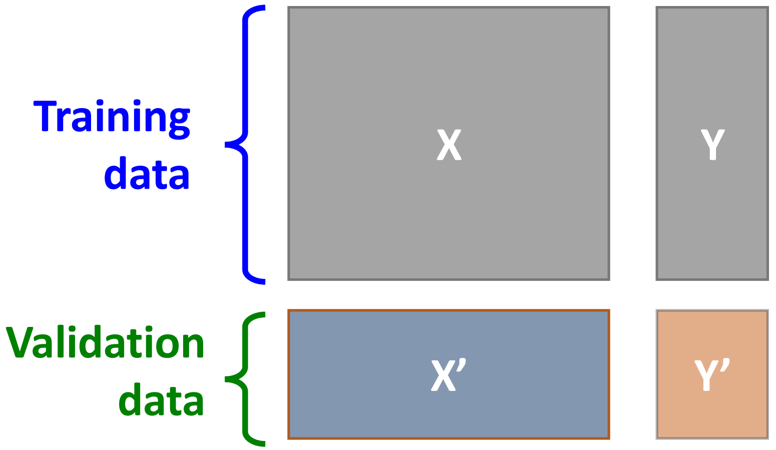
- Training data (e.g., 70-90%)
- Test data (e.g., 10-20%)
- Development data or Validation data (10-20%)

- You need to report performance on test data, but you are not allowed to look at it.
  - You are allowed to look at the development data (and use it to tune parameters)

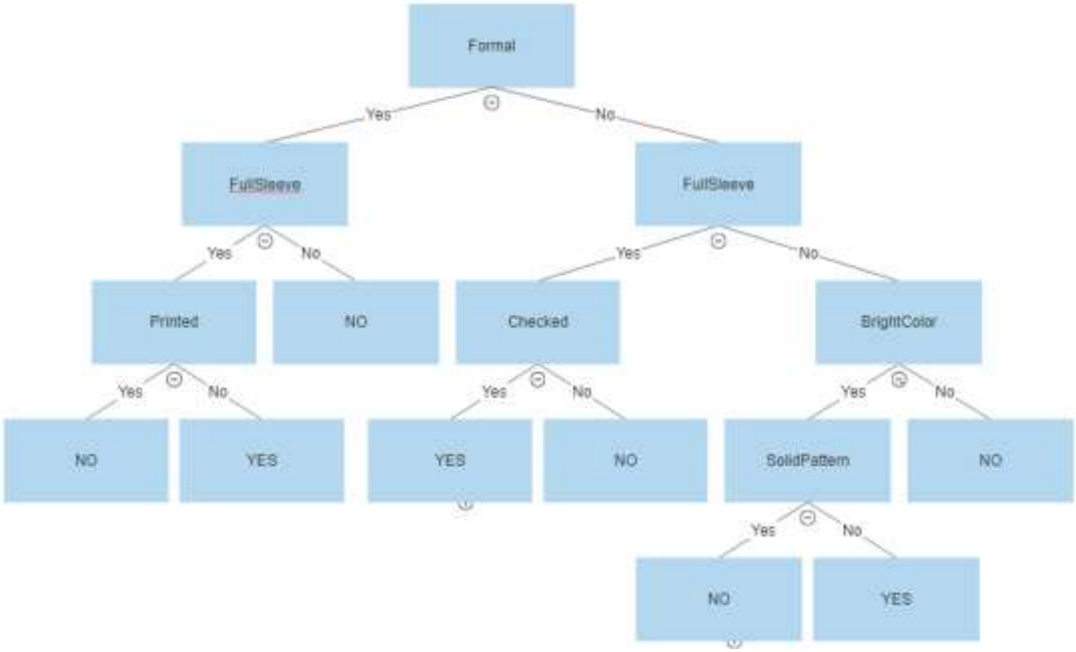


Validation

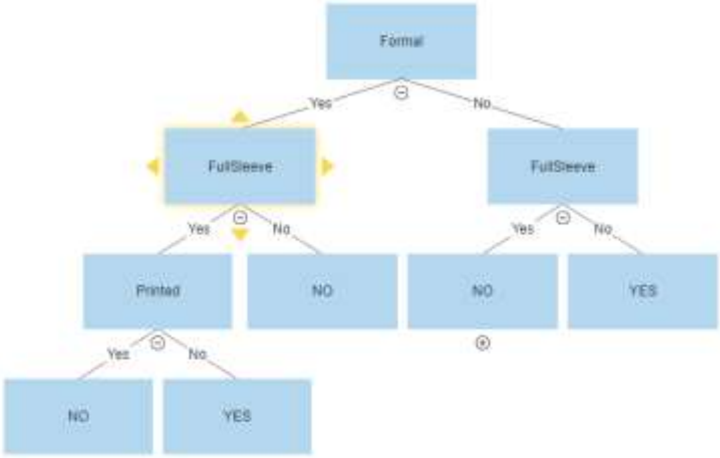
- Divide your data randomly into training and *Validation* data.
- Build your best model based on the training data only.
- Apply your model to the Validation data.
- Does your model predict y’ for the Validation data as well as it predicted y for the training data?



Which Decision Tree?



Training Error = 0.05
Test Error = 0.2

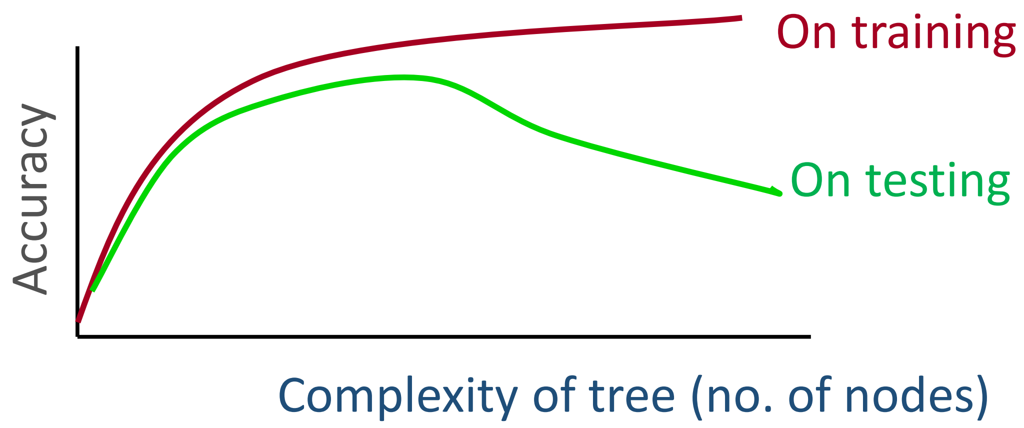


Training Error = 0.1
Test Error = 0.15

Overfitting

Overfitting :

- Fit the training data too well
- But fail to generalize to new examples



Why does Overfitting happen?

- Noise
- Irrelevant Features
- Insufficient Data
- Training data not representative

Overfitting results in decision trees that are more complex than necessary

Overfitting

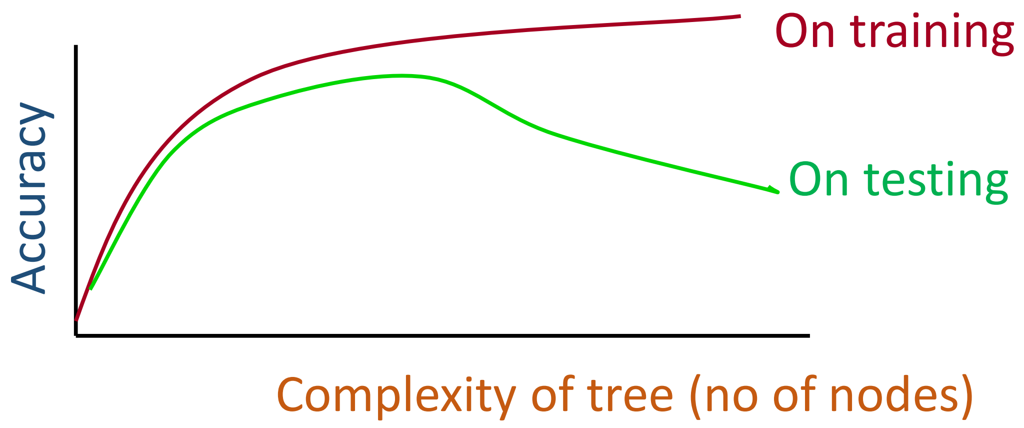
A hypothesis h is said to overfit the training data if there is another hypothesis h' such that h has smaller error than h' on the training data but h has larger error on the test data than h' .

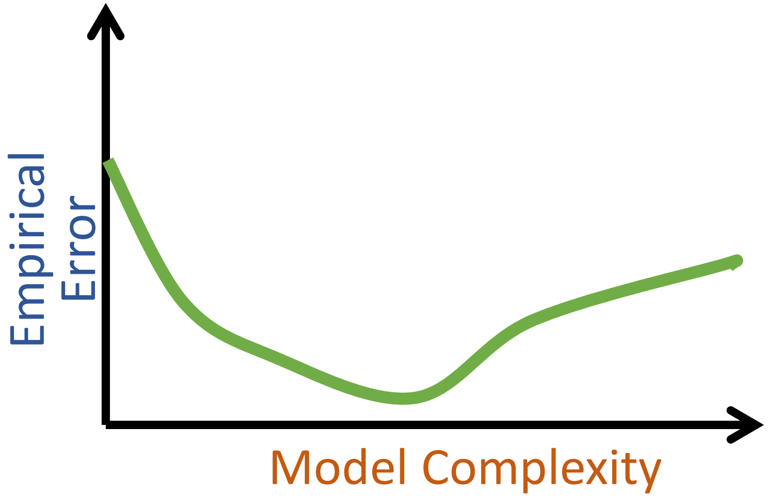
In other words, hypothesis h overfits if there is $h' \in \mathcal{H}$ such that

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

and

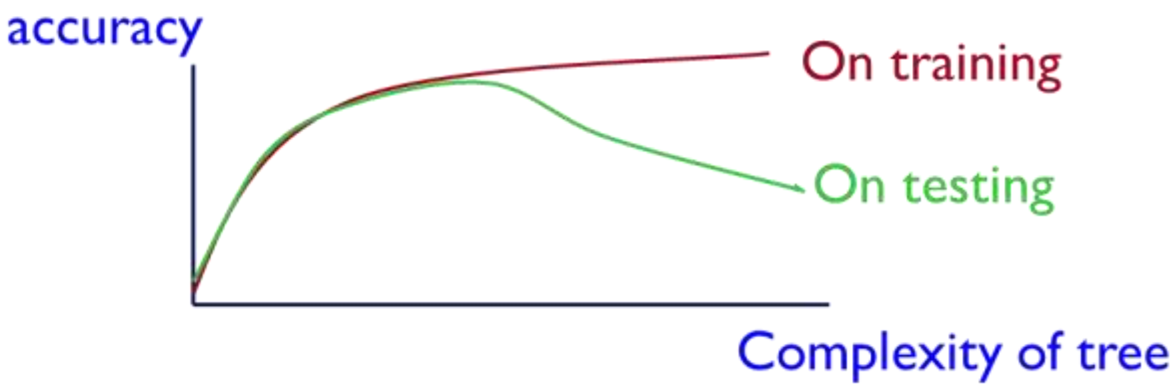
$$error_{true}(h) > error_{true}(h')$$



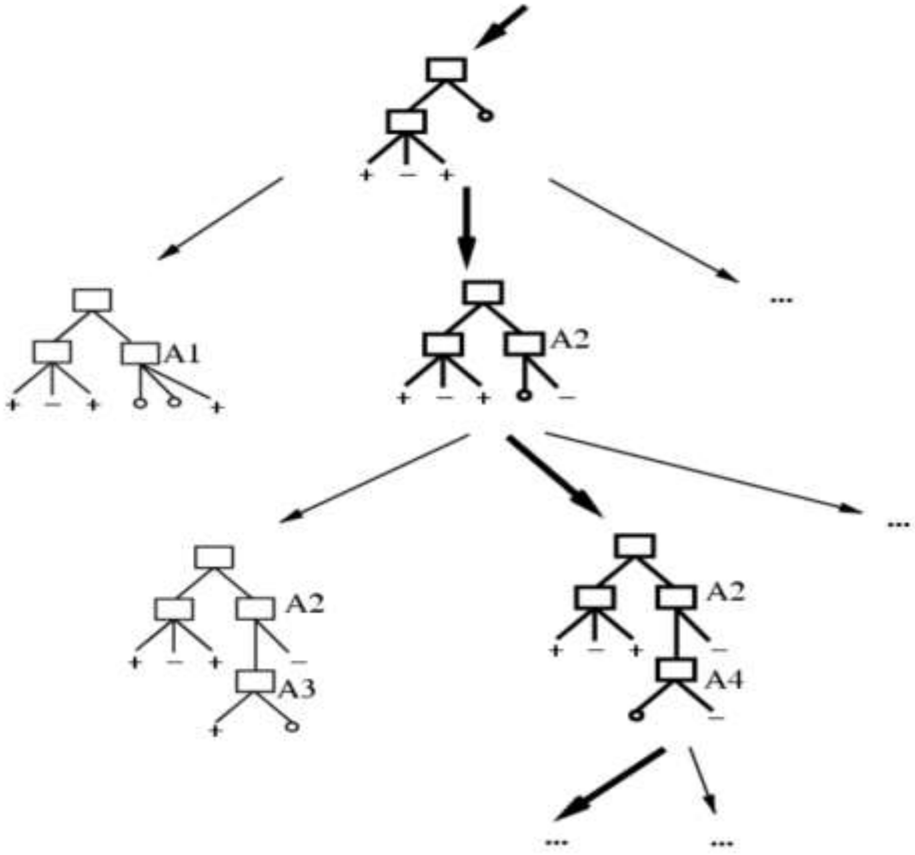


Overfitting in Decision Trees

- Your model shows much greater loss on the test data than on the training data.
- Example: a decision tree with so many levels that the typical leaf is reached by only one member of the training set.



Overfitting in Practice (ID3 – sklearn)



- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

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

Pruning a decision tree

- Pruning The Tree: remove unnecessary nodes to
  - make it more efficient and
  - solve overfitting problems.

1. Prepruning: Stop growing when data split not statistically significant
2. Postpruning: Grow full tree 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?

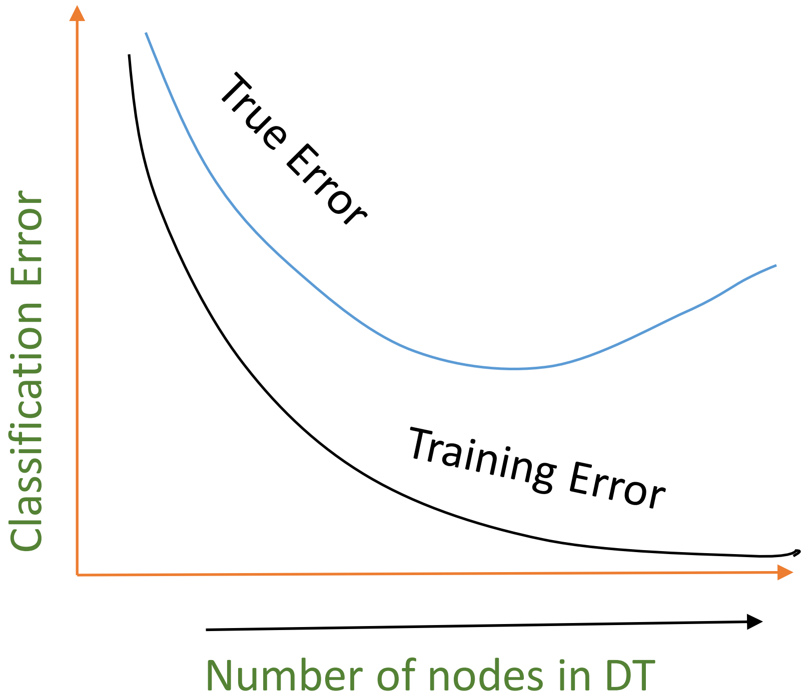
This is related to the notion of regularization – keep the hypothesis simple

Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes

Pre-Pruning (Early Stopping)

- Early Stopping: Stop the learning algorithm before tree becomes too complex



Stopping conditions:

- Do not split a node which contains too few instances
- Stop if expanding the current node does not improve impurity measures significantly (e.g., Gini or information gain)
- Limit tree depth

Reduced-error Pruning

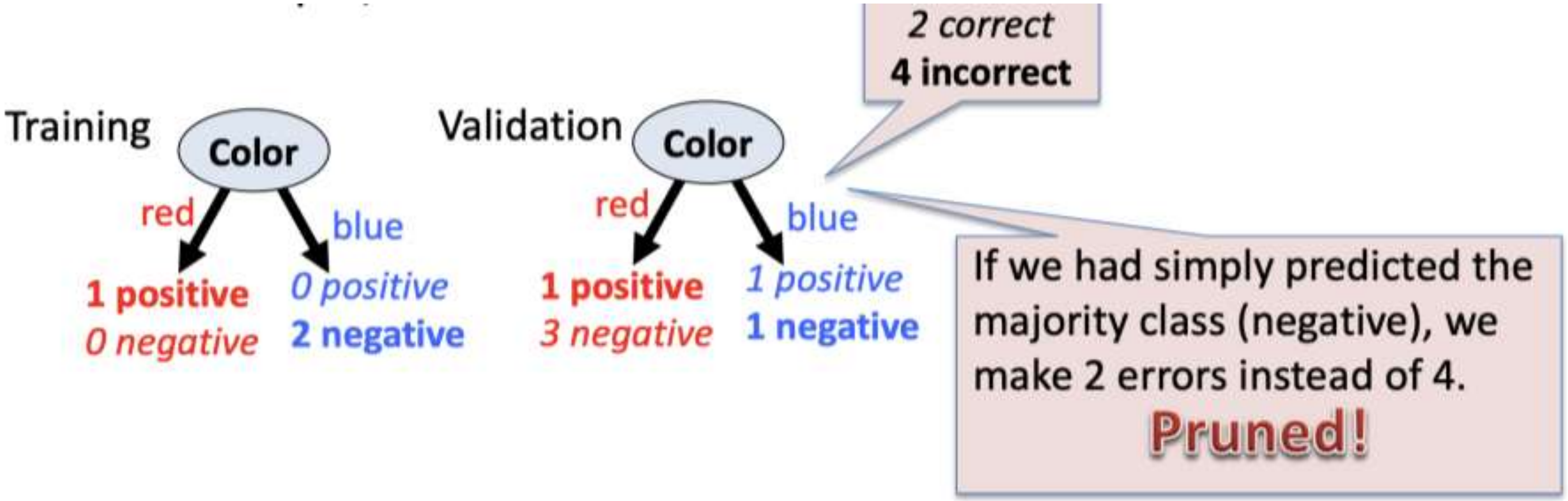
Partition data into train set and validation set

- Build a tree using the train set.
- Until accuracy on validation set decreases, do:
  - For each non-leaf node in the tree
    - ✓ Temporarily prune the tree below; replace it by majority vote
    - ✓ Test the accuracy of the hypothesis on the validation set
    - ✓ Permanently prune the node with the greatest increase in accuracy on the validation test.

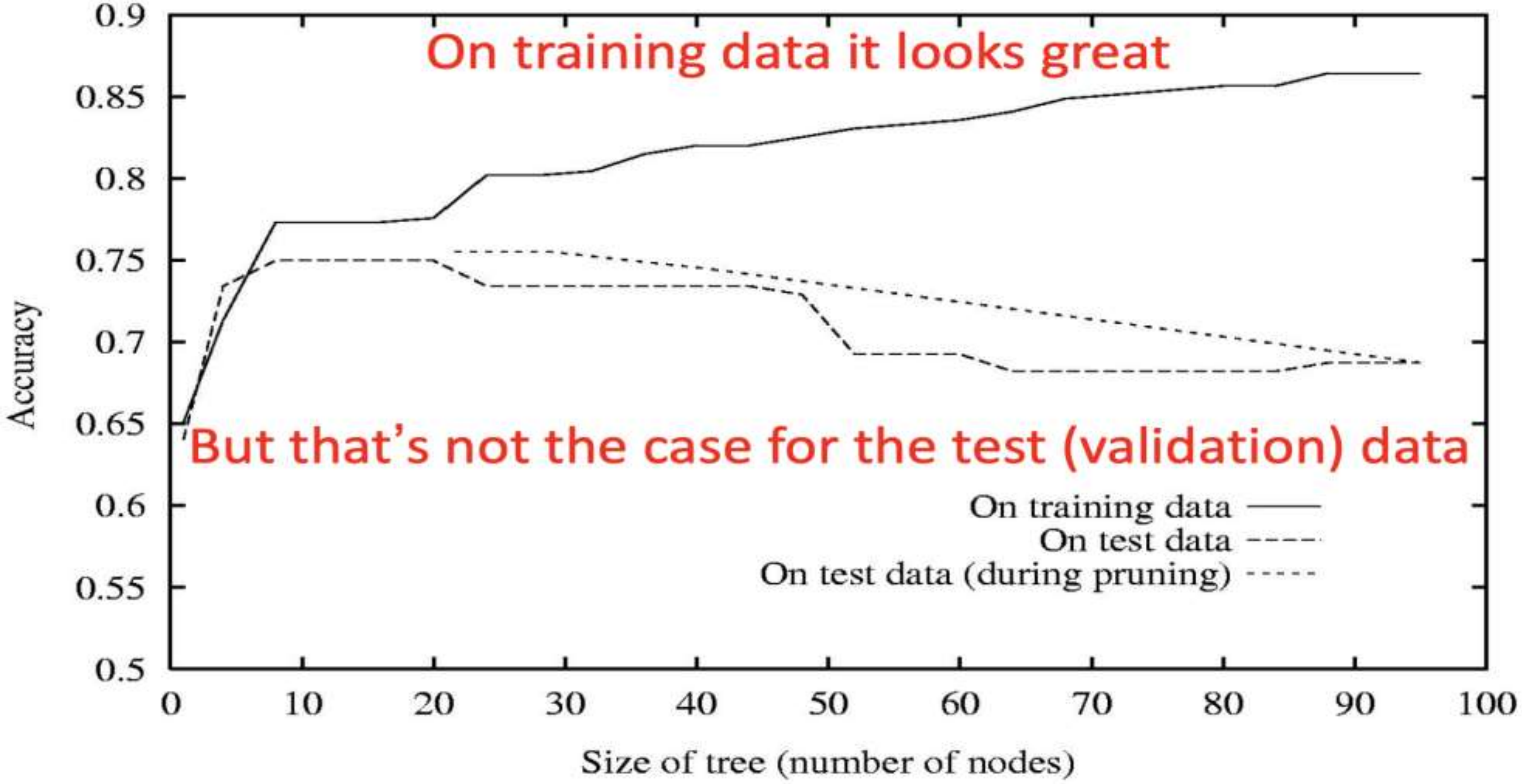
Pruning Decision Trees

Pruning the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that

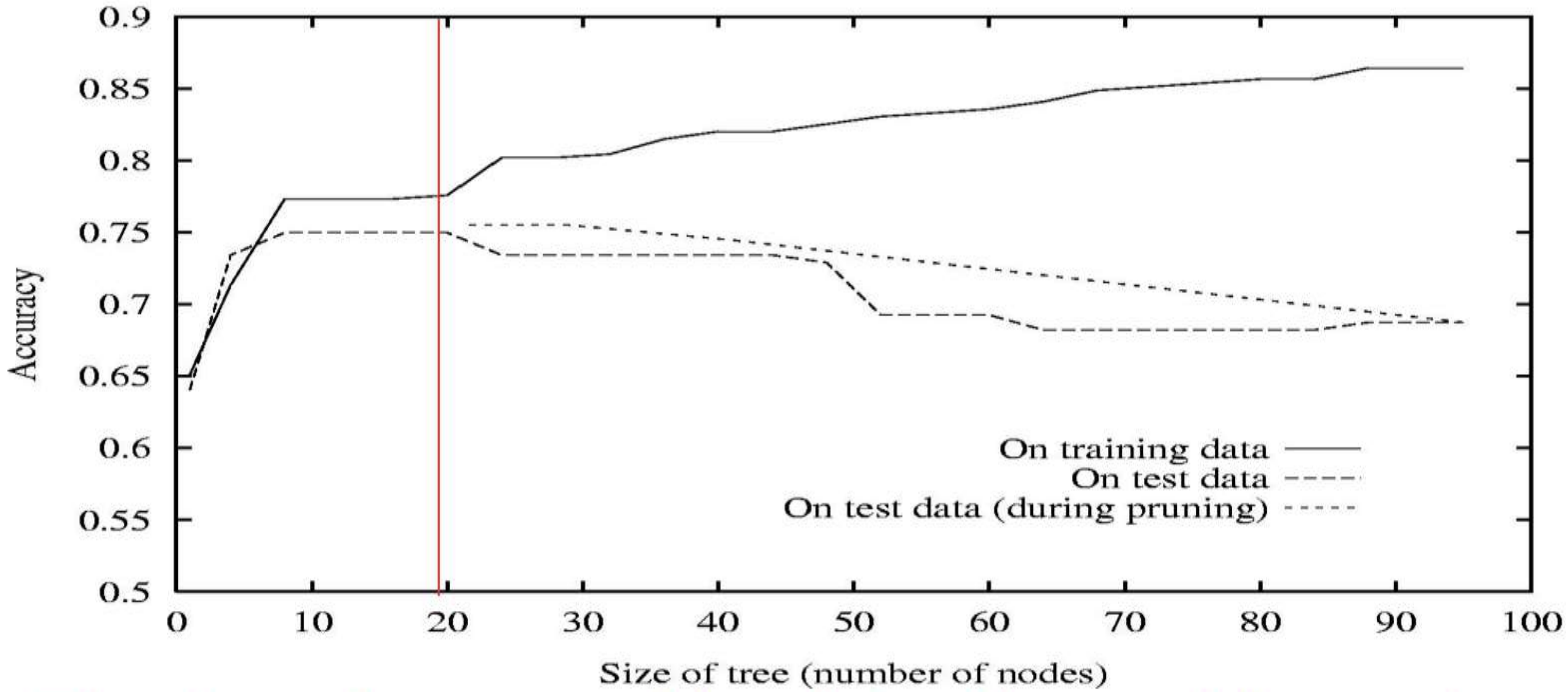
- the Expected Error Rate in the subtree > Expected error rate in the single leaf
- For example



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

Bias & Variance

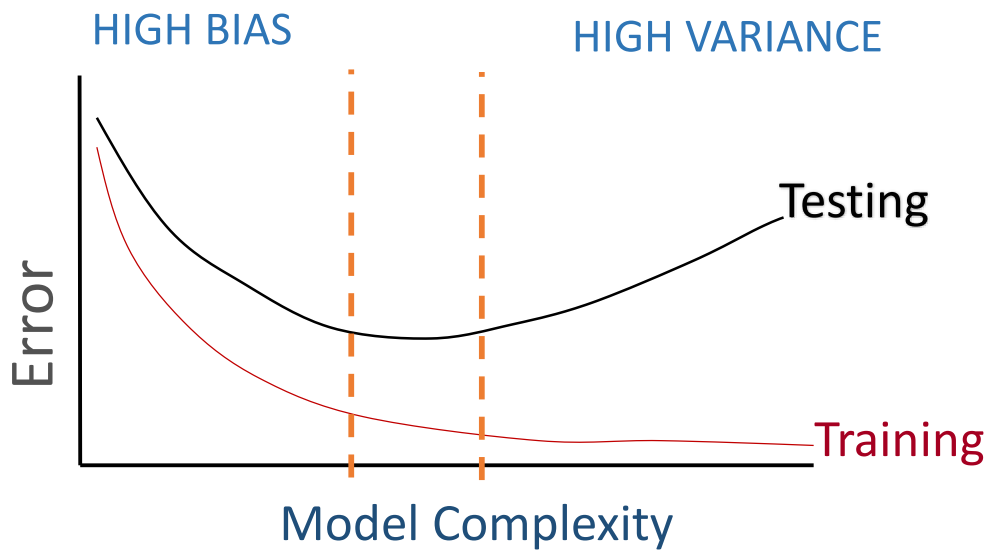
Overfitting vs Underfitting

Underfitting

- Not able to capture the concept
  - Features don't capture concept
  - Model is not powerful.

Overfitting

- Fitting the data too well



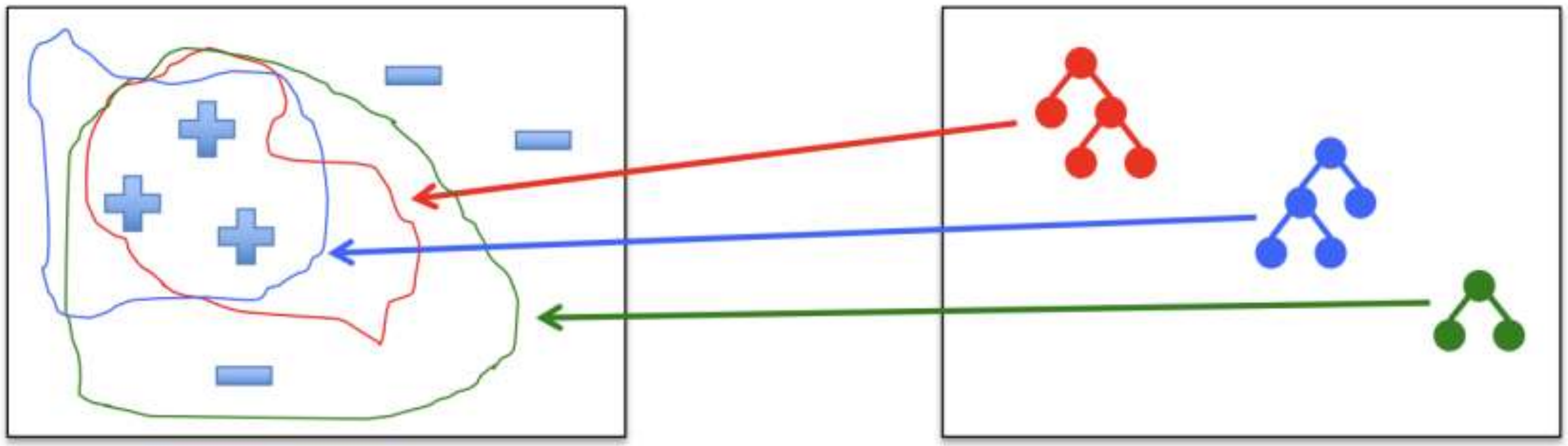
Function Approximation: The Big Picture

Instance Space $\mathcal{X} = \{0,1\}^d$

$\boldsymbol{x} = \langle x_1, x_2, \ldots, x_d \rangle \in \mathcal{X}$

Hypothesis Space

$H = \{h \mid h : \mathcal{X} \mapsto \{0,1\}\}$



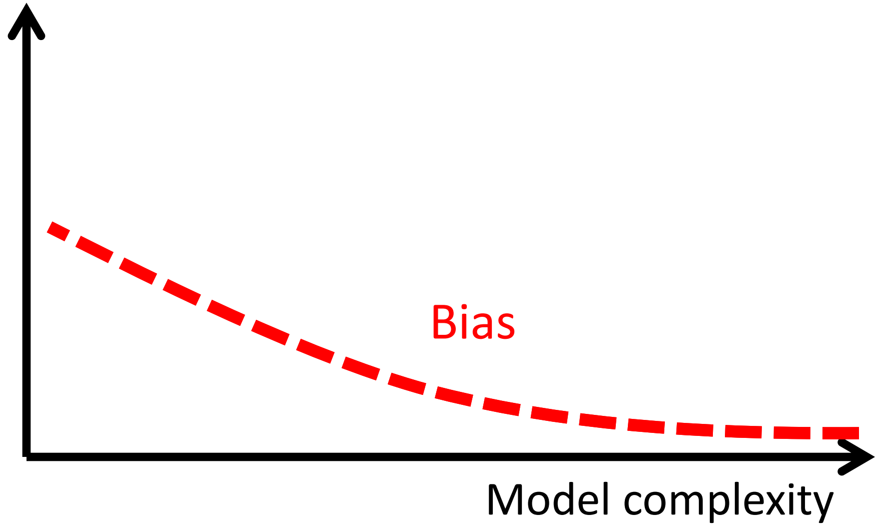
if $d = 20$, $|\mathcal{X}| = 2^{20}$

$|h| = 2^{|\mathcal{X}|} = 2^{2^{20}}$

- How many labeled instances are needed to determine which of the $2^{2^{20}}$ hypotheses are correct?
    - All 2^{20} instances in must be labeled!
- Generalizing beyond the training data (inductive inference) is impossible unless we add more assumptions (e.g., priors over H)

Bias of a Learner (~ mean error)

- How likely is the learner to identify the **target** hypothesis?
- Bias is low when the model is expressive (low empirical error)



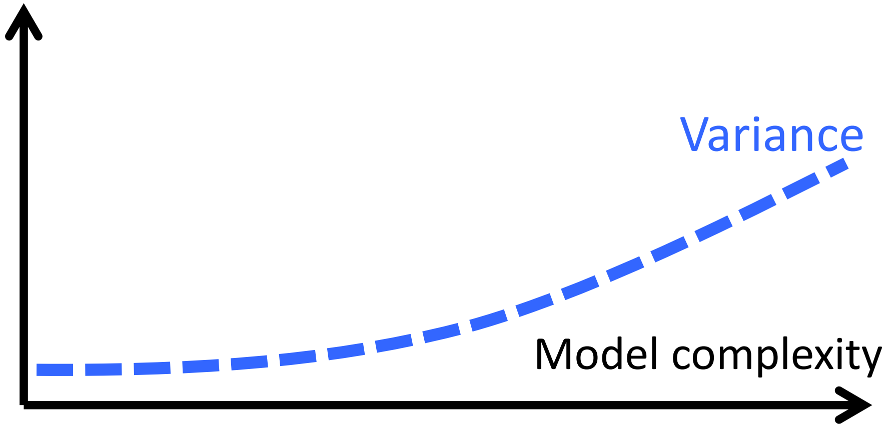
- Bias is high when the model is too simple
  - The larger the hypothesis space is, the easiest it is to be close to the true hypothesis.
  - For each data set D,
    - You learn a different hypothesis $h(D)$, that has a different true error $error_{true}(h)$;
    - difference of the mean of this random variable from the true error.

if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.

$$Bias = \mathrm{E}[f_D(X) - y]$$

Variance of a Learner

How susceptible is the learner to different subsets of the training data? (i.e. to different $D \sim P(\boldsymbol{X}, Y)$)



Variance increases with model complexity

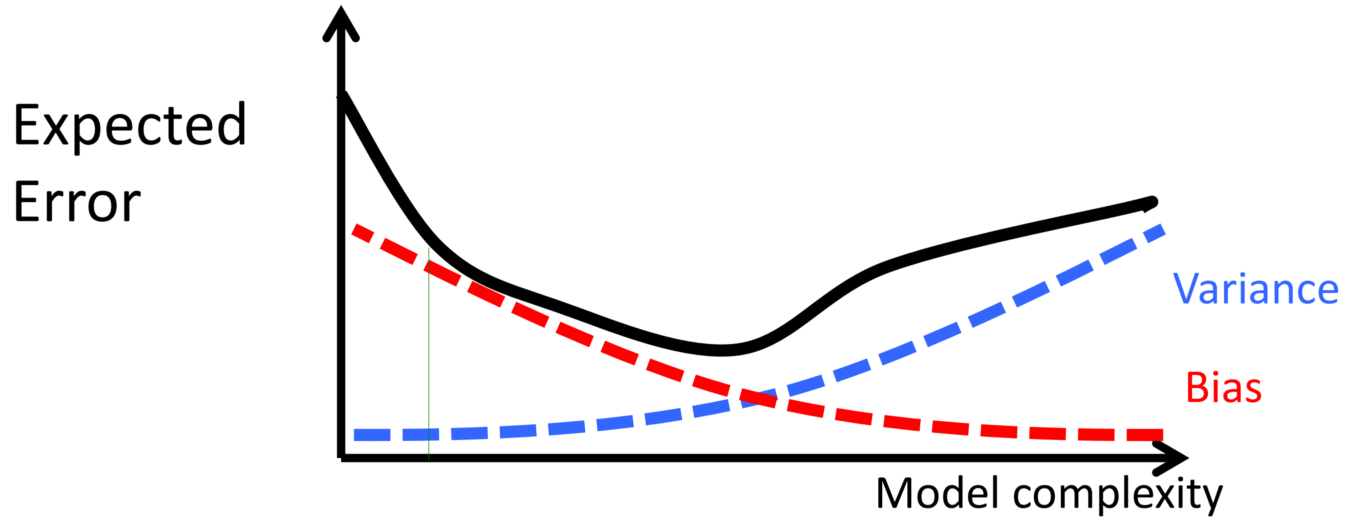
- The larger the hypothesis space, the more flexible the selection of the chosen hypothesis is as a function of the data.
- For each data set D,
  - you will learn a different hypothesis $h(D)$, that will have a different error $error_{true}(h)$;
  - Lets see the variance of this random variable.

if we train models $f_D(X)$ on many training sets D, the variance of the estimates:

$$Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$$

(~ std.dev among predictions)

Impact of bias and variance



Expected error ≈ bias + variance (why???)

Bias-Variance Decomposition of Squared Error

- Assume that $y = f(x) + \epsilon$
  - Noise ϵ is sampled from a normal distribution with 0 mean and variance $\sigma^2 : \epsilon \sim N(0, \sigma^2)$
  - Noise lower-bounds the performance (error) we can achieve.
- Recall the objective function:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - h_\theta\left(x^{(i)}\right) \right)^2$$

- We view this as an approximation of the expected value of the squared error: $E\left(y - h_\theta(x)\right)^2$

Bias-Variance Decomposition of Squared Error

$$E\big(y - h_\theta(x)\big)^2 = \mathrm{E}\Big[\big(\mathrm{y} - \mathrm{f(x)} + \mathrm{f(x)} - \mathrm{h}_\theta(\mathrm{x})\big)^2\Big]$$

$$= \mathrm{E}\Big[\big(\mathrm{y} - \mathrm{f(x)}\big)\hat{}2\,\Big] + \mathrm{E}\big[\mathrm{f(x)} - \mathrm{h}_\theta(\mathrm{x})\big)^2\Big] + 2E\big[\big(f(x) - h_\theta(x)\big)\big(y - f(x)\big)\big]$$

$$= \mathrm{E}\Big[\big(\mathrm{y} - \mathrm{f(x)}\big)\hat{}2\Big] + \mathrm{E}\big[\mathrm{f(x)} - \mathrm{h}_\theta(\mathrm{x})\big)^2\Big] + 2\big(\cancel{E[f(x)h_\theta(x)]} + \cancel{E[yf(x)]}$$

$$- \cancel{E[yh_\theta(x)]} - \cancel{E[f(x)^2]}\big)$$

Therefore

$$E\big(y - h_\theta(x)\big)^2 = \mathrm{E}\Big[\big(\mathrm{y} - \mathrm{f(x)}\big)^2\Big] + \mathrm{E}\Big[\big(\mathrm{f(x)} - \mathrm{h}_\theta(\mathrm{x})\big)^2\Big]$$

$$= \mathrm{E}[\epsilon^2] + \mathrm{E}\Big[\big(\mathrm{f(x)} - \mathrm{h}_\theta(\mathrm{x})\big)^2\Big]$$

Aside:
Definition of Variance
$\mathrm{var}(z) = \mathbf{E}[(z - \mathbf{E}[z])^2]$

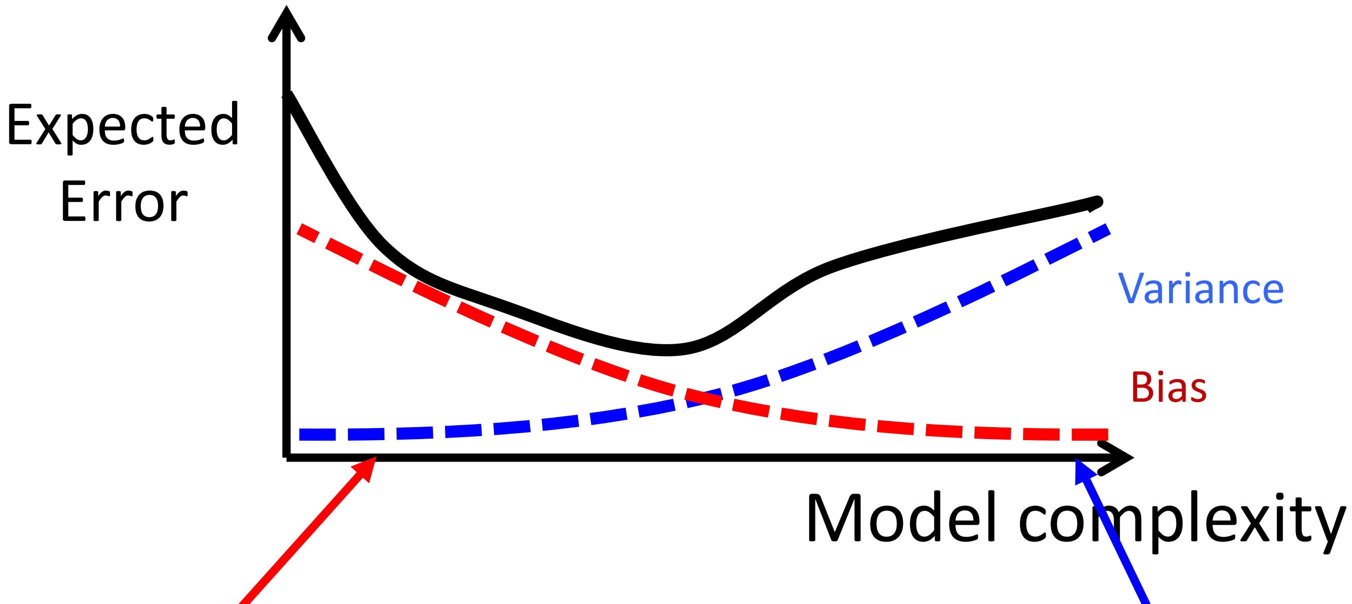
This is $var(\epsilon)$ since mean is 0.

Bias-Variance Decomposition of Squared Error

$$
\begin{aligned}
\mathrm{E}[(y - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^2] &= \mathrm{var}(\epsilon) + \mathrm{E}[(f(\boldsymbol{x}) - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^2] \\
&= \mathrm{var}(\epsilon) + \mathrm{E}[(f(\boldsymbol{x}) - \mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})] + \mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})] - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^2] \\
&= \mathrm{var}(\epsilon) + \mathrm{E}[(f(\boldsymbol{x}) - \mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})])^2] + \mathrm{E}[(\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})] - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^2] \\
&\qquad + 2\mathrm{E}[(\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})] - h_{\boldsymbol{\theta}}(\boldsymbol{x}))(f(\boldsymbol{x}) - \mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})])] \\
&= \mathrm{var}(\epsilon) + \mathrm{E}[(f(\boldsymbol{x}) - \mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})])^2] + \mathrm{E}[(\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})] - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^2] \\
&\qquad + 2\left(\cancel{\mathrm{E}[f(\boldsymbol{x})\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})]]} - \cancel{\mathrm{E}[\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})]^2]} - \cancel{\mathrm{E}[f(\boldsymbol{x})h_{\boldsymbol{\theta}}(\boldsymbol{x})]} + \cancel{\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})\mathrm{E}[h_{\boldsymbol{\theta}}(\boldsymbol{x})]]}\right)
\end{aligned}
$$

cancels cancels

Model complexity



Simple models:
High bias and low variance

Complex models:
High variance and low bias

BIAS

- Error caused because the model can not represent the concept
- Bias is the expected difference between the model prediction and the true y's.
- Higher Bias:
  - Decision tree of lower depth
  - Linear functions
  - Important features missing

if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.

$$Bias = \mathrm{E}[f_D(X) - y]$$

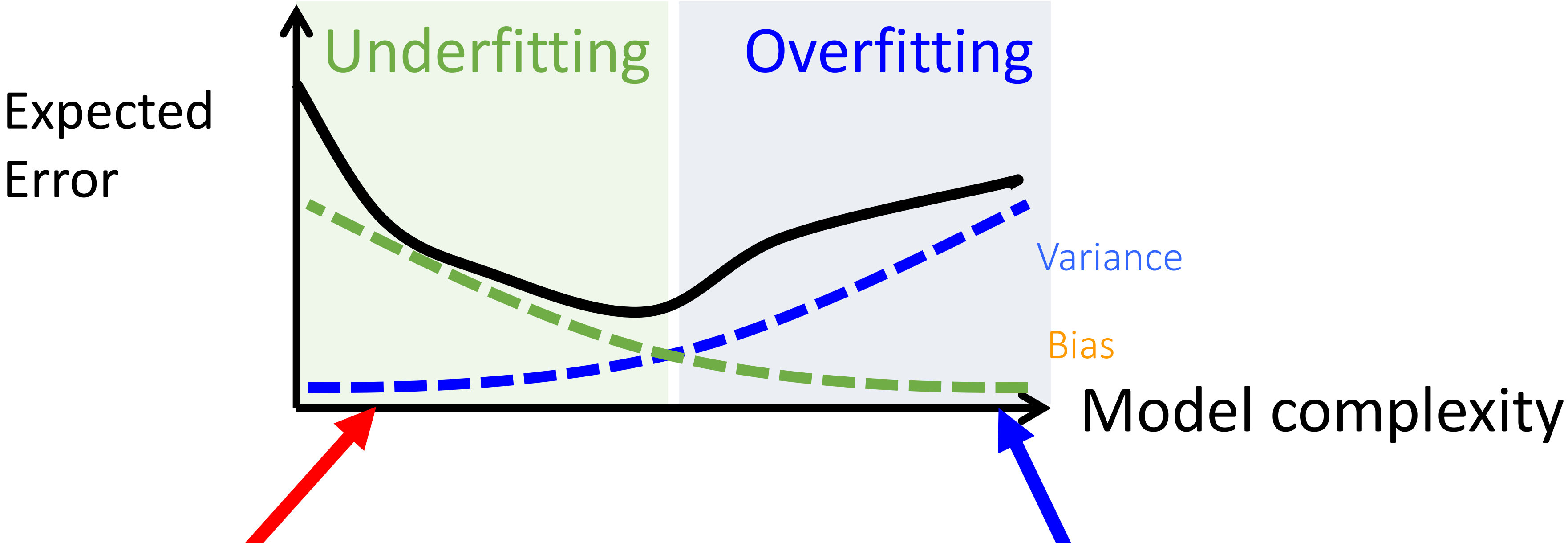
VARIANCE

- Error caused because the learned model reacts to small changes (noise) in the training data
- High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs
- Higher Variance
  - Decision tree with large no of nodes
  - High degree polynomials
  - Many features

if we train models $f_D(X)$ on many training sets D, variance is the variance of the estimates:

$$Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$$

Underfitting and Overfitting



Simple models:
High bias and low variance

Complex models:
High variance and low bias

This can be made more accurate for some loss functions.

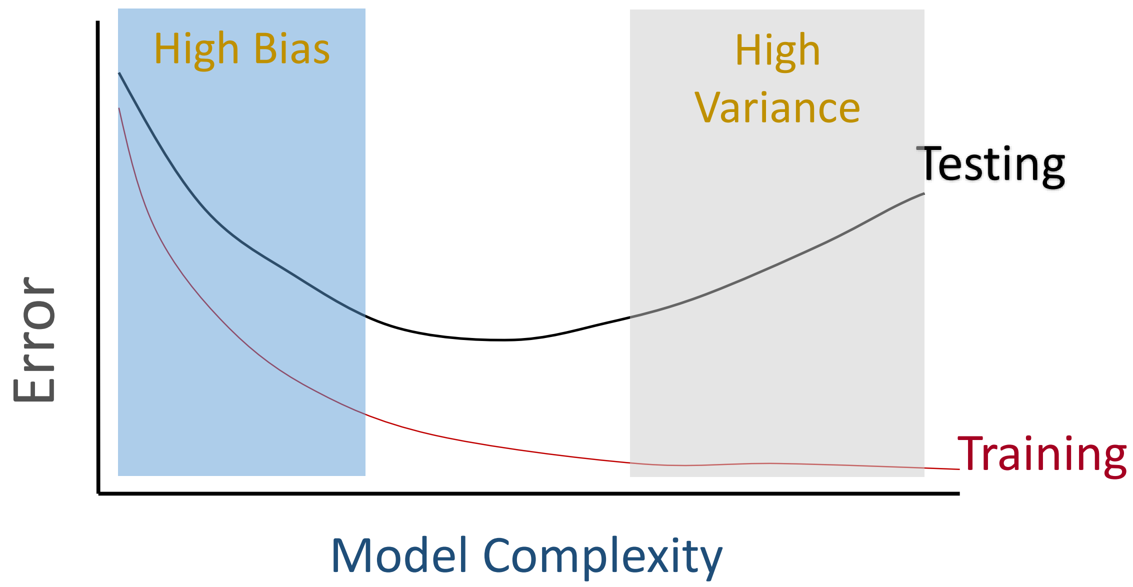
We will discuss a more precise and general theory that trades expressivity of models with empirical error

Bias and Variance Tradeoff

There is usually a bias-variance tradeoff caused by model complexity.

Complex models often have lower bias, but higher variance.

Simple models often have higher bias, but lower variance.



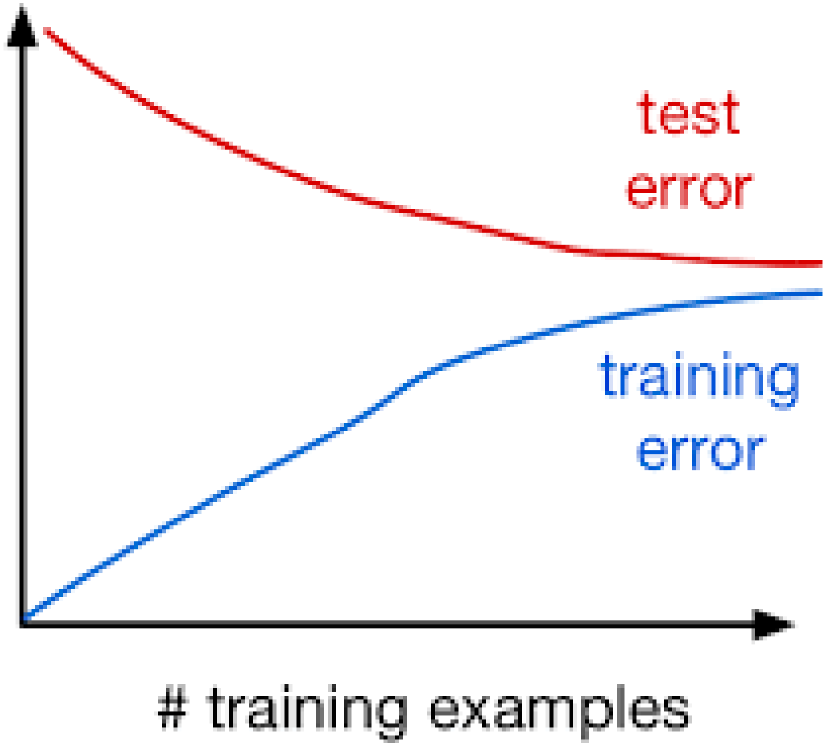
Trade-Offs

$$Error \approx Function(Complexity, TrainingDataSize)$$

- There is a trade-off between these factors:
  - Complexity of Model $c(H)$
  - Training set size, m,
  - Generalization error, E on new data

1. As m *increases*, E decreases
2. As $c(H)$ *increases*,
   1. first E *decreases* and then E *increases*
   2. the training error *decreases* for some time and then stays constant (frequently at 0)

As m increases, E decreases



Model complexity

2. As c (H) increases, first E decreases and then E increases
3. As *c* (H) *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)