Supervised learning (II)

MSBD 5001 Fall 2018,Lecture 3

Kai Chen, CSE, HKUST

Recall from Last Lecture

- Supervised learning methods
  - Linear methods, kNN, decision tree
- We assumed training & testing data are i.i.d. from same distribution
  - Often hard to guarantee
- Where we stopped: What if training & testing data are not i.i.d.?

Not i.i.d.?

- I.i.d. assumption: The training data has the same characteristics of testing data
  - Or training data is “almost the same” as testing data
- We train the model to fit the training data.
- Learn to much on training data, can’t generalize well on testing data.
- We call this overfitting: a modeling error that a function is too closely fit to a limited set of data points.

Why is Overfitting a Problem?

- 3 steps:
  - 1. Collect data, extract feature
  - 2. Select model
  - 3. Train the model
  - We want to use the model for later analysis, e.g., predict some outcome
- Mission no. 1 is to make testing error small, NOT training error!

Outline

- Generalization of models: How is my model performing on unseen data?
  - Overfitting vs Underfitting
  - Why do overfitting & underfitting happen?
- Selecting good models
  - Cross validation
- Improving the models
  - Regularization
  - Ensemble learning

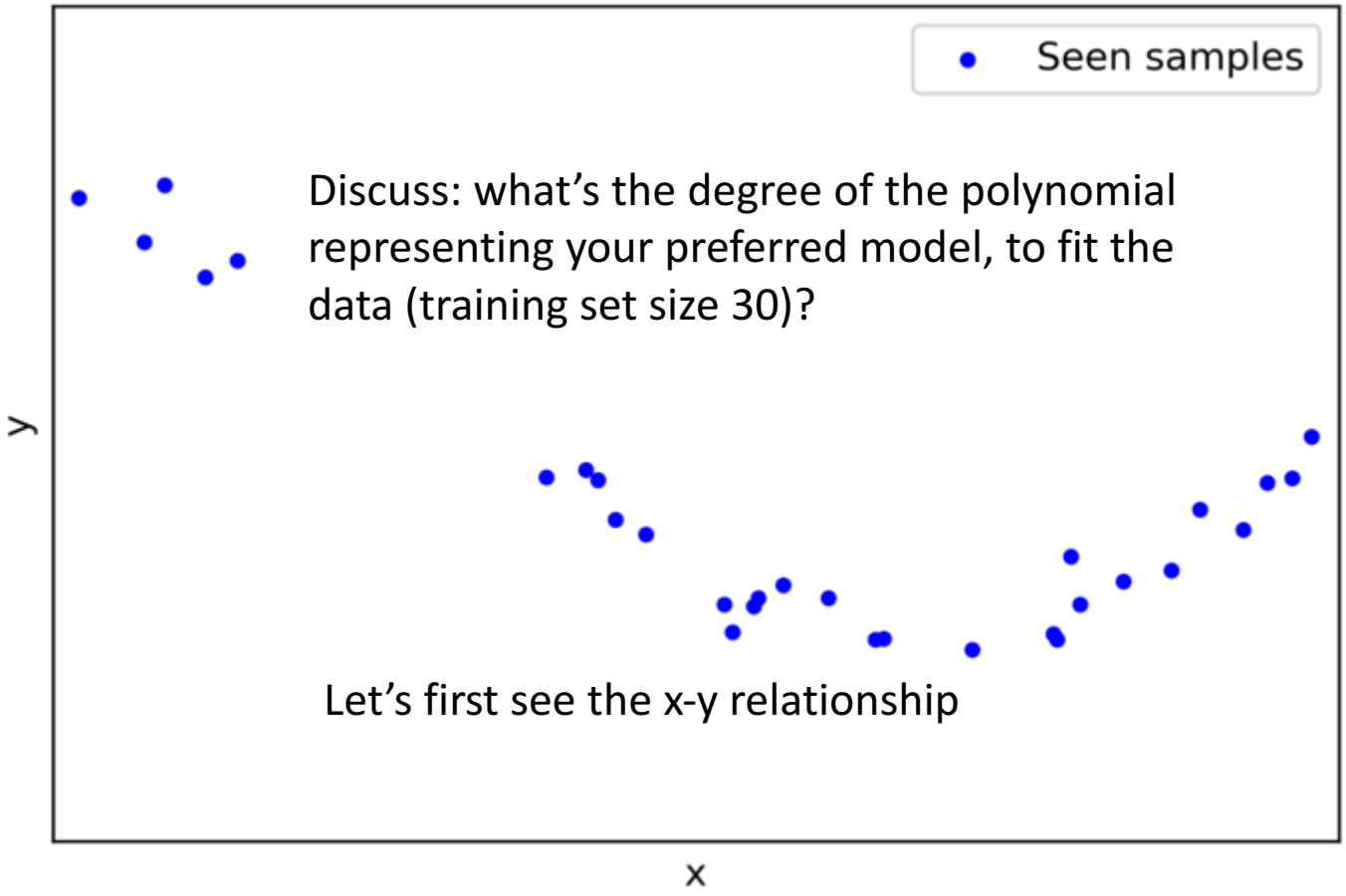
Bad Generalization with Overfitting

- Goal is to have models generalize well: perform well on unseen data
- Simulation: Suppose we use a polynomial function to fit an unknown dataset.
  - How should we choose the degree of polynomial?
- We plot the simulation process to show our setup
  - Code available on Canvas

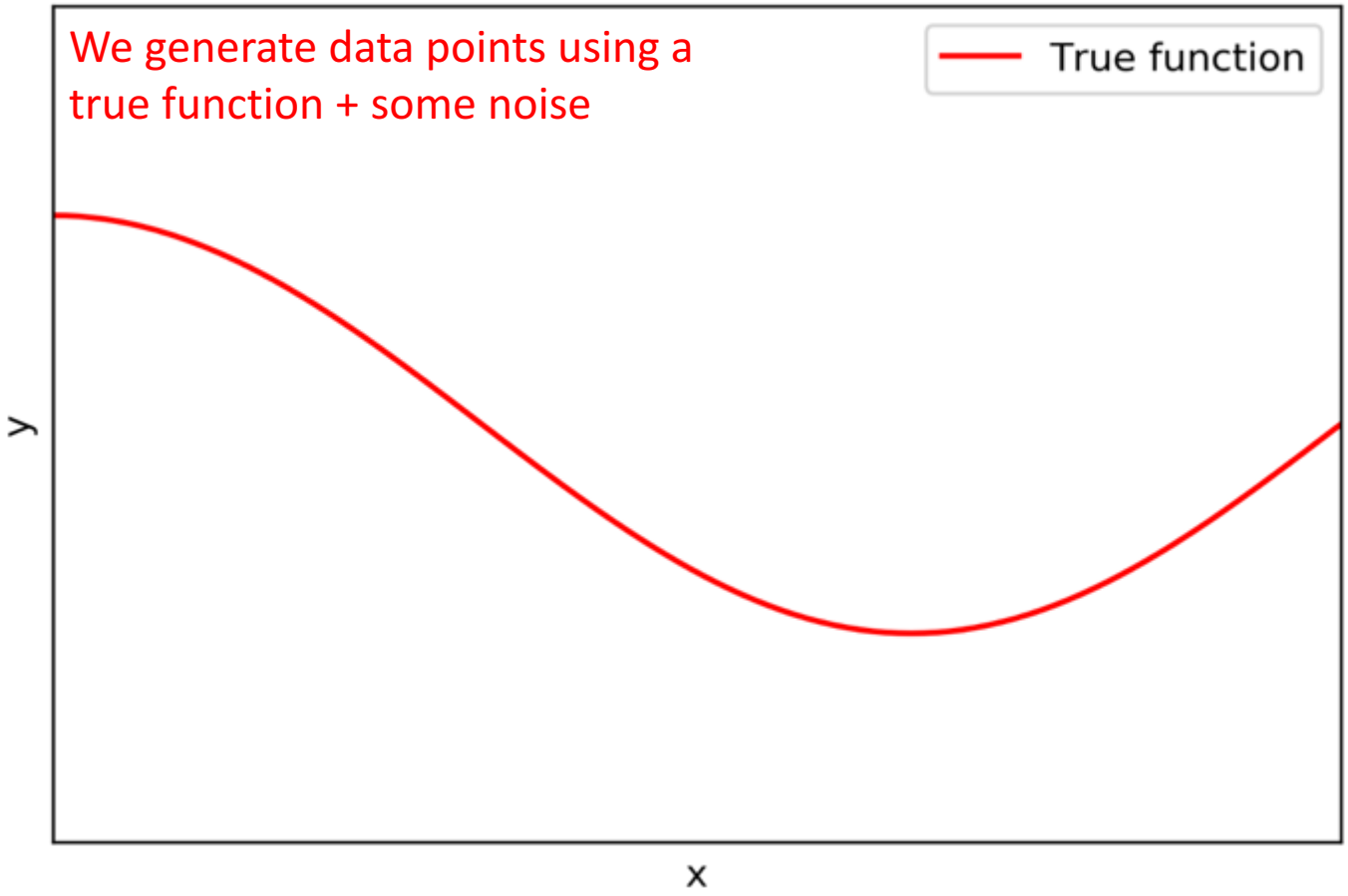
Plotting a Polynomial

- Using a polynomial of degree N to fit:
  - $y = \sum_{i=1}^{N} w_i x^i$
- Higher degree has more complex curves:
  - There exists a k-degree polynomial that pass through k+1 data points (x,y).
  - Is this what we want?
  - Let's see an example

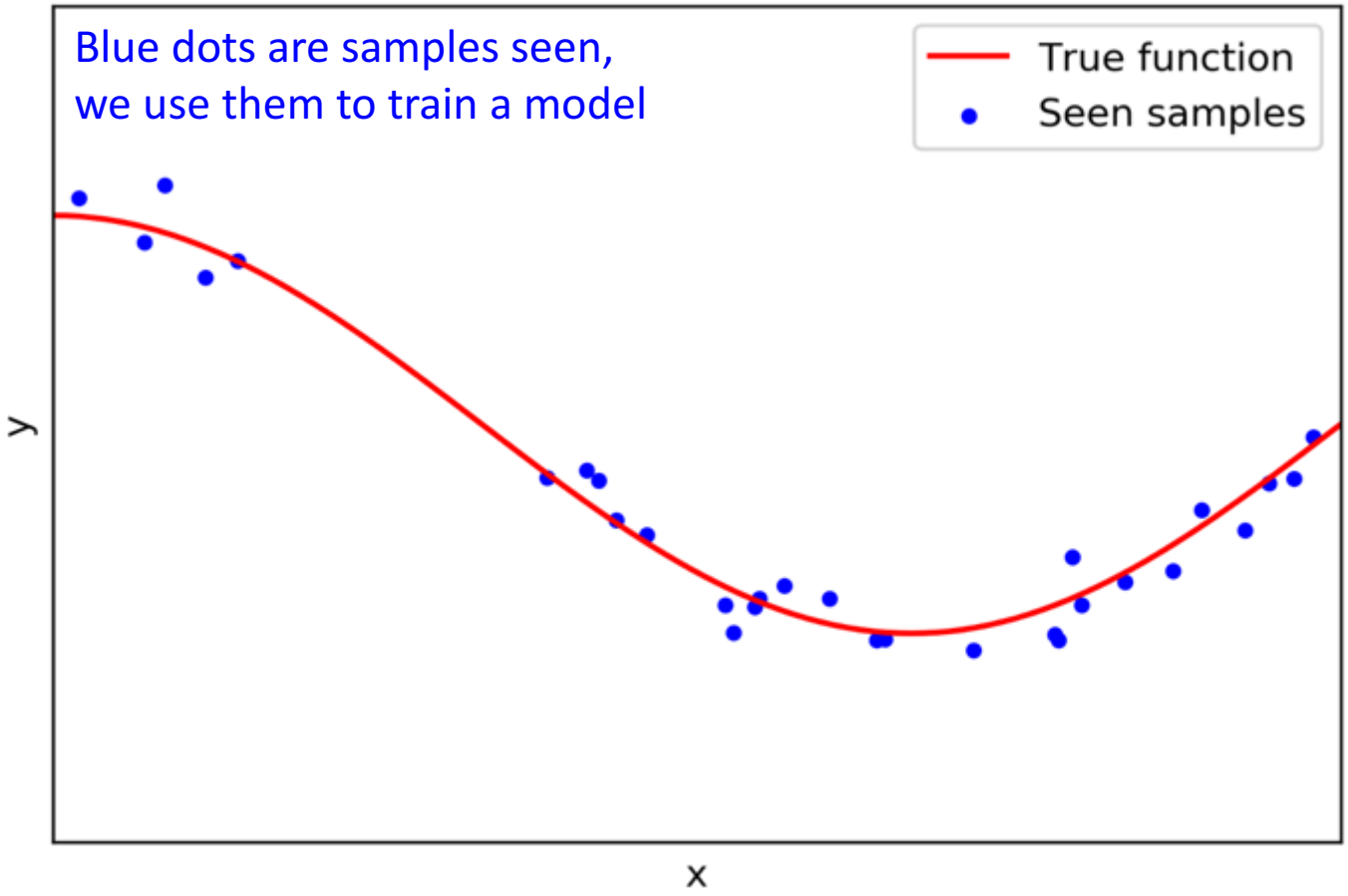
Polynomial Degree?



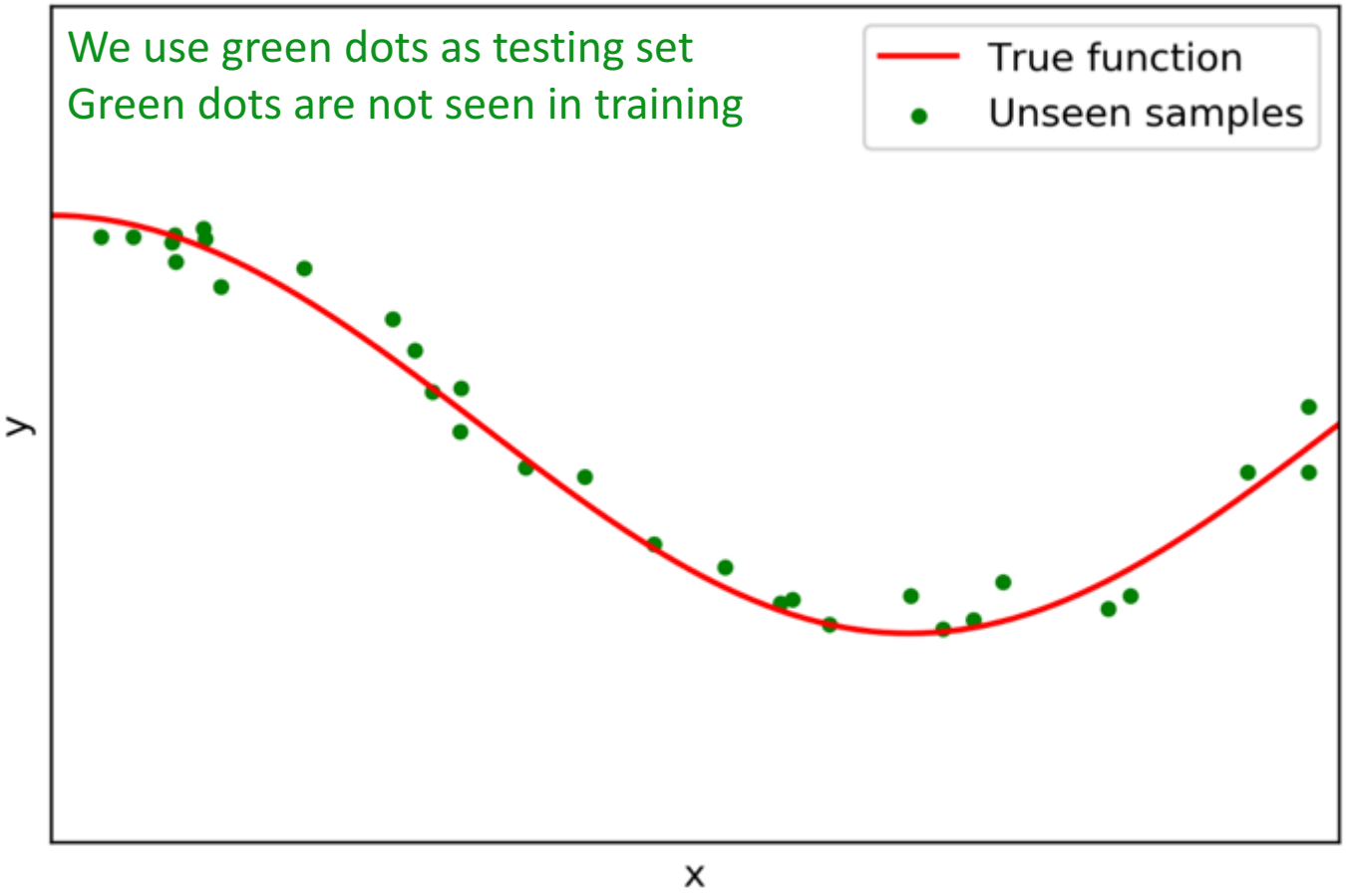
Simulation Setup



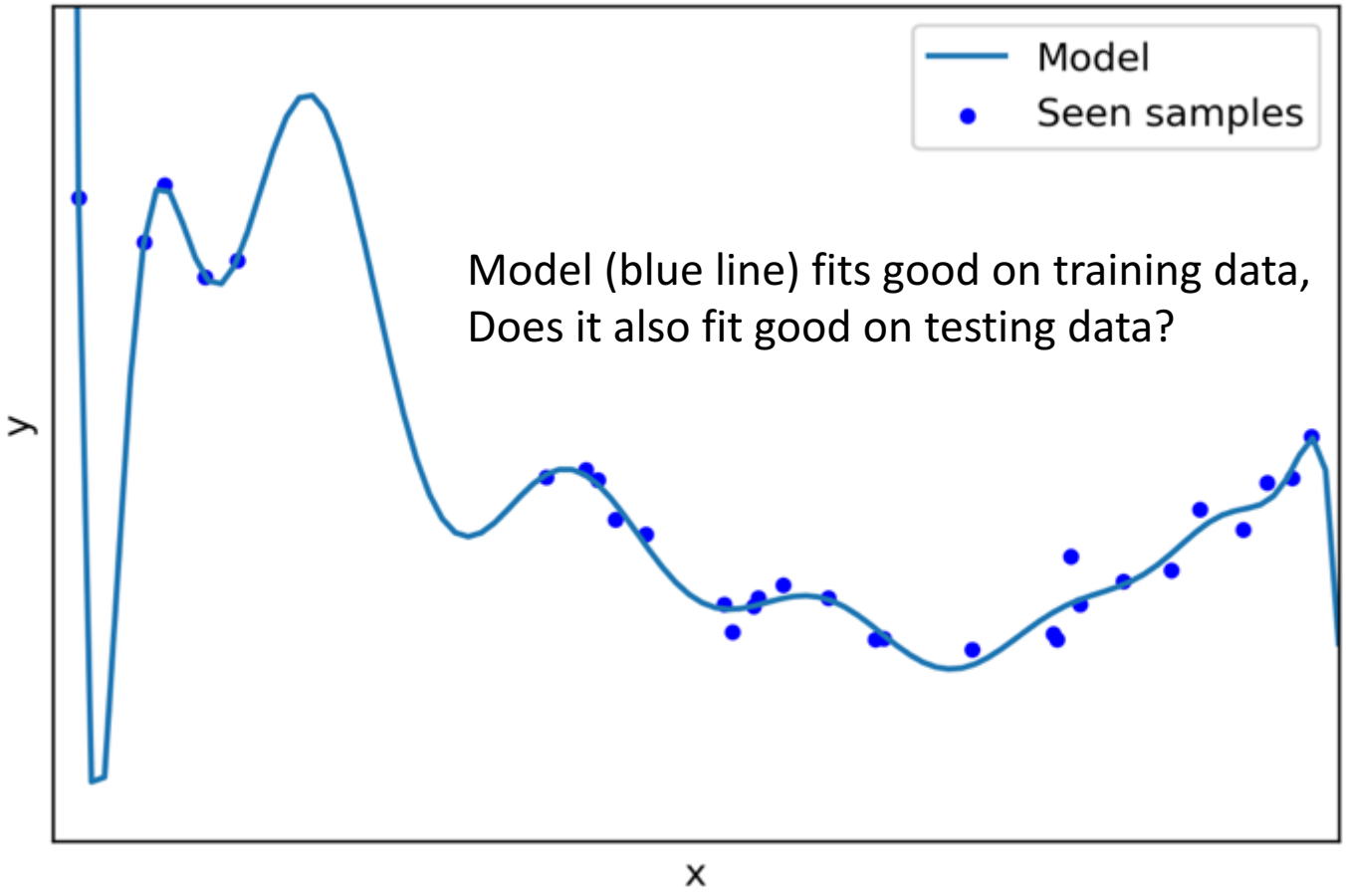
Simulation Setup



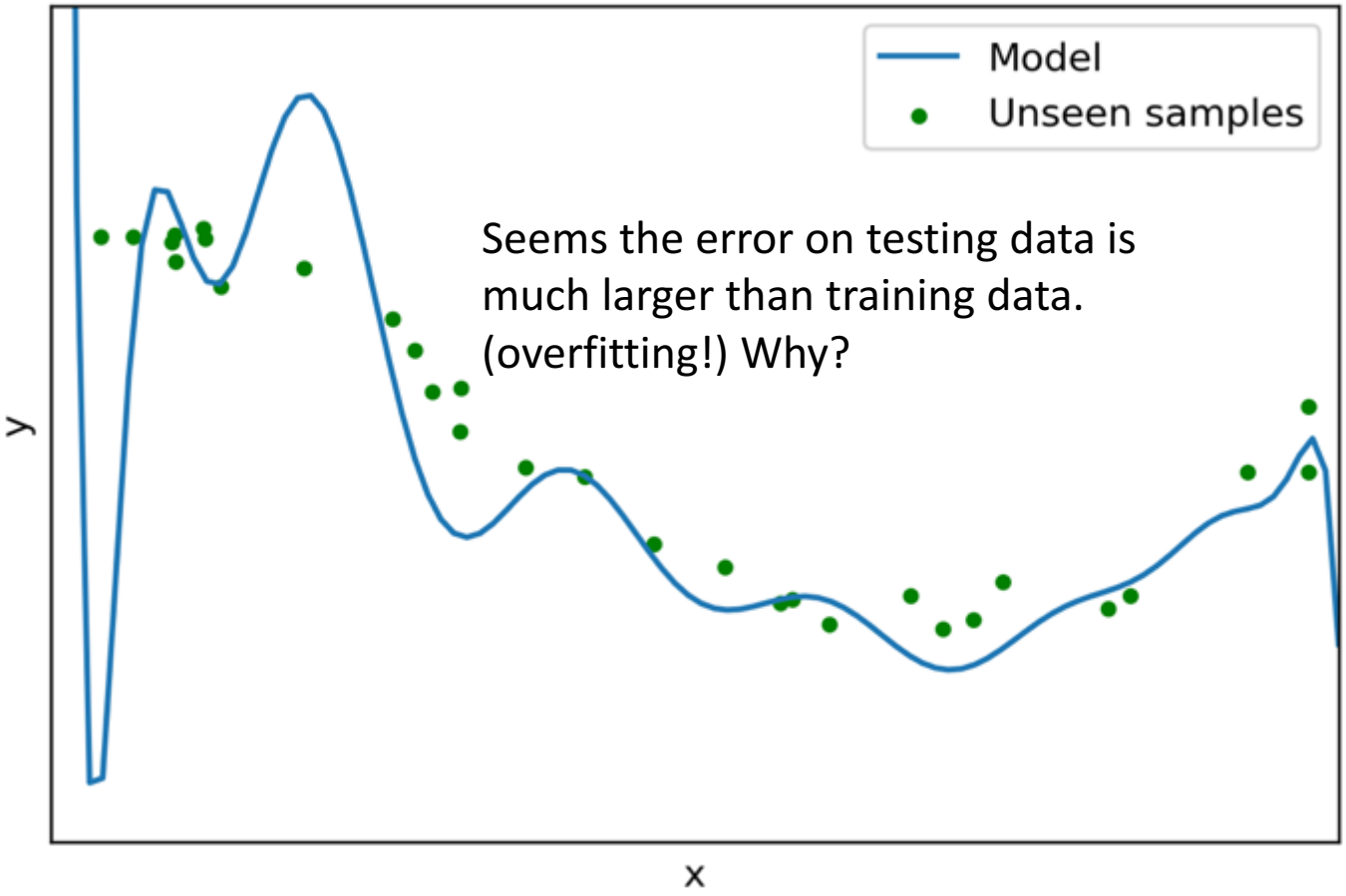
Simulation Setup



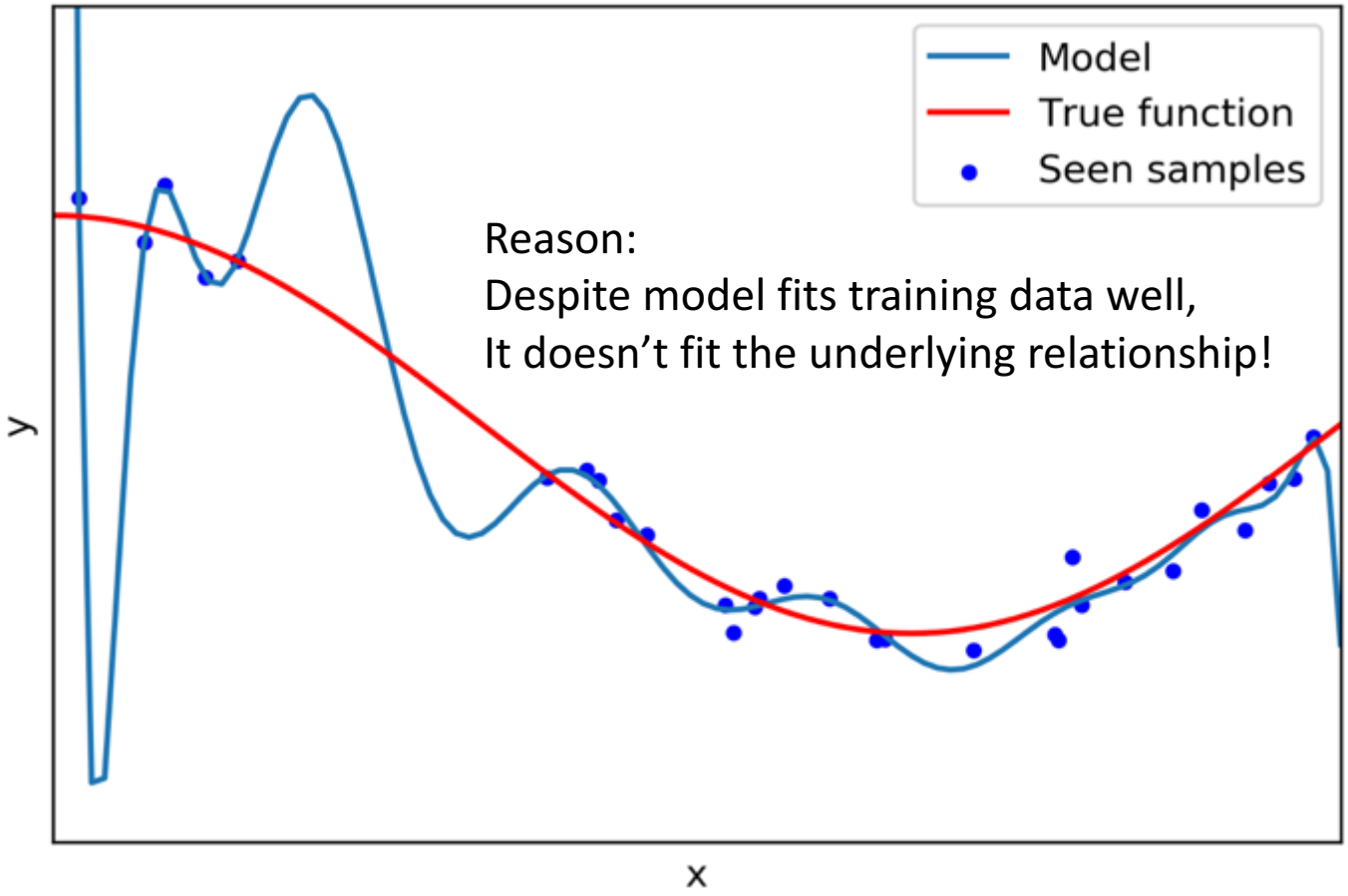
Overfitting with degree=15



Overfitting with degree=15



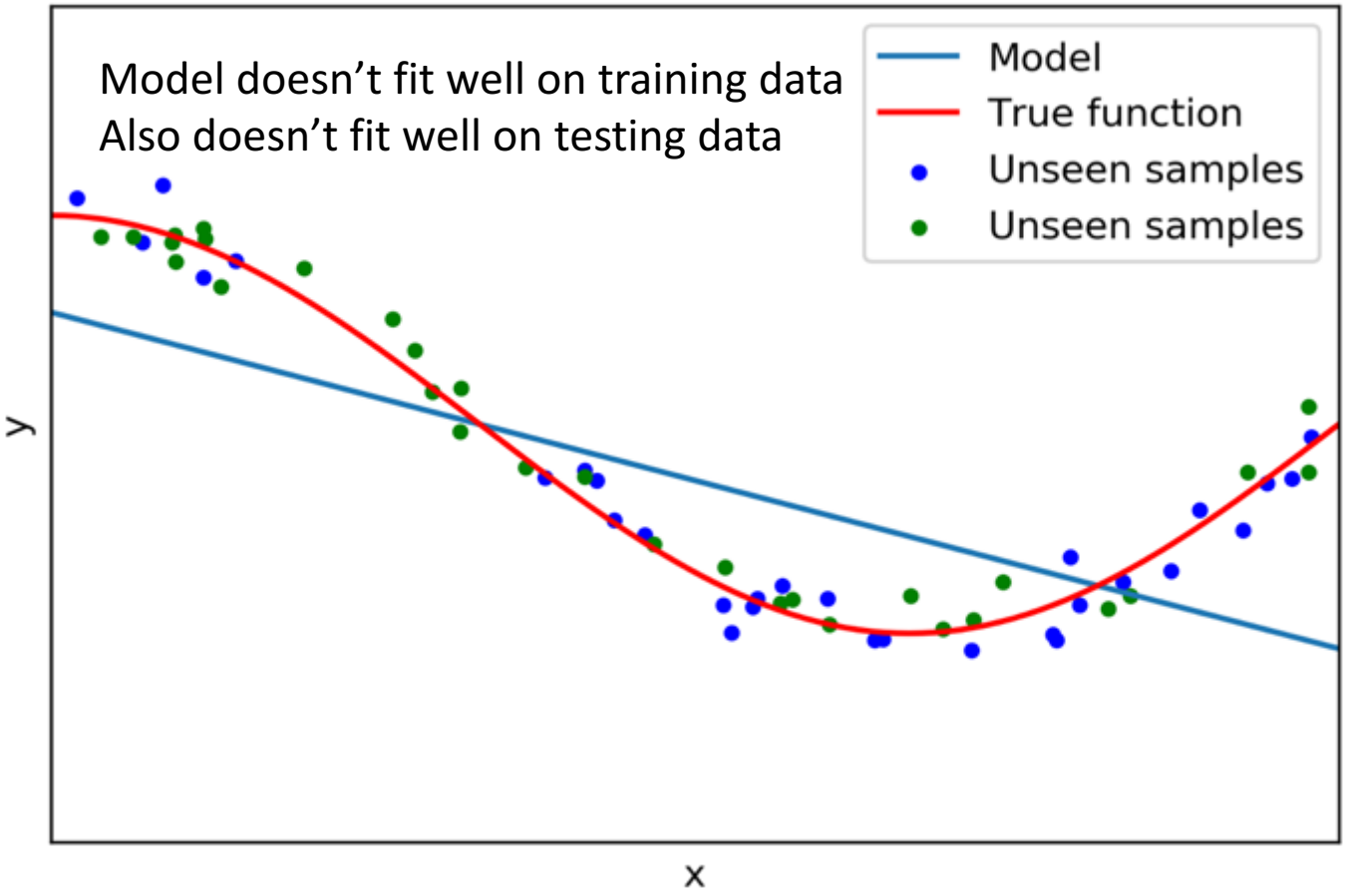
Overfitting with degree=15



Overfitting vs Underfitting

- **Overfitting** is a modeling error which occurs when a function is too closely fit to a limited set of data points
- As the name suggests, underfitting is the “contrast” case of overfitting
- **Underfitting** occurs when the model or the algorithm does not fit the data well enough
  - Not often seen as overfitting, not of “severe” impact.
  - Not discussed much today

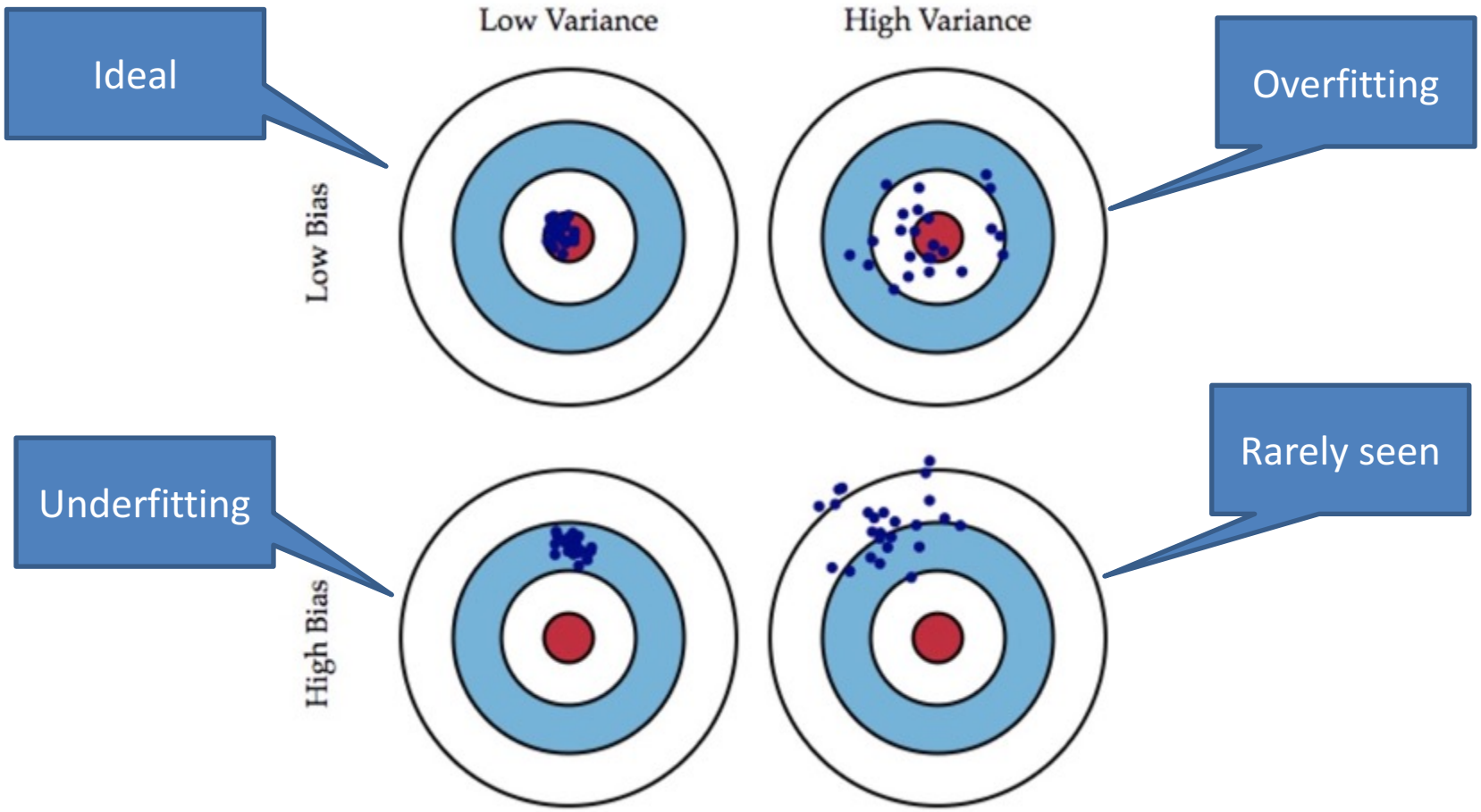
Underfitting Example



Errors: Bias and Variance

- Bias: difference between the expected or average prediction of our model and the correct value which we are trying to predict.
  - High bias: High error on training and testing set
- Variance error from sensitivity to small fluctuations in the training set.
  - How much will model move around its mean
  - High variance: model fit to random noise of data, rather than valid information (overfitting)

Visualized Relationship



Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Tradeoff

- Consider regression problem, with error measured as squared error:

$$L_f(x, y) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Bias-variance tradeoff tells us that, the expected error on an unseen dataset can be expressed as relationship between bias, variance, and noise:
- Expect error = Bias^2 + Variance + Noise
- Discuss in tutorial

Proof

- Assume $y = f(X) + \sigma$
  - σ is of mean 0, variance non-zero (represents noise) and f is deterministic. Use $\hat{f}$ to represent the selected model to approximate f
  - Can you prove the following?

$$\mathrm{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \left(\mathrm{Bias}\left[\hat{f}(x)\right]\right)^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$$

where

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right]$$

and

$$\mathrm{Var}\left[\hat{f}(x)\right] = \mathrm{E}[\hat{f}(x)^2] - \left(\mathrm{E}[\hat{f}(x)]\right)^2$$

Hints for the Proof

- Prove:

$$\mathrm{E}\left[\left(y-\hat{f}\left(x\right)\right)^2\right]=\left(\mathrm{Bias}\left[\hat{f}\left(x\right)\right]\right)^2+\mathrm{Var}\left[\hat{f}\left(x\right)\right]+\sigma^2$$

- Linearity of expectation:

$$\mathbb{E}[A+B]=\mathbb{E}[A]+\mathbb{E}[B]$$

- Variance:

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

- f is deterministic, so

$$\mathbb{E}(y)=\mathbb{E}(f)=f$$

Proof

$$
\begin{aligned}
\mathrm{E}\left[(y-\hat{f})^2\right] &= \mathrm{E}[y^2+\hat{f}^2-2y\hat{f}] \\
&= \mathrm{E}[y^2]+\mathrm{E}[\hat{f}^2]-\mathrm{E}[2y\hat{f}] \\
&= \mathrm{Var}[y]+\mathrm{E}[y]^2+\mathrm{Var}[\hat{f}]+\left(\mathrm{E}[\hat{f}]\right)^2-2f\,\mathrm{E}[\hat{f}] \\
&= \mathrm{Var}[y]+\mathrm{Var}[\hat{f}]+\left(f^2-2f\,\mathrm{E}[\hat{f}]+(\mathrm{E}[\hat{f}])^2\right) \\
&= \mathrm{Var}[y]+\mathrm{Var}[\hat{f}]+(f-\mathrm{E}[\hat{f}])^2 \\
&= \sigma^2+\mathrm{Var}[\hat{f}]+\mathrm{Bias}[\hat{f}]^2
\end{aligned}
$$

Bias-Variance Tradeoff

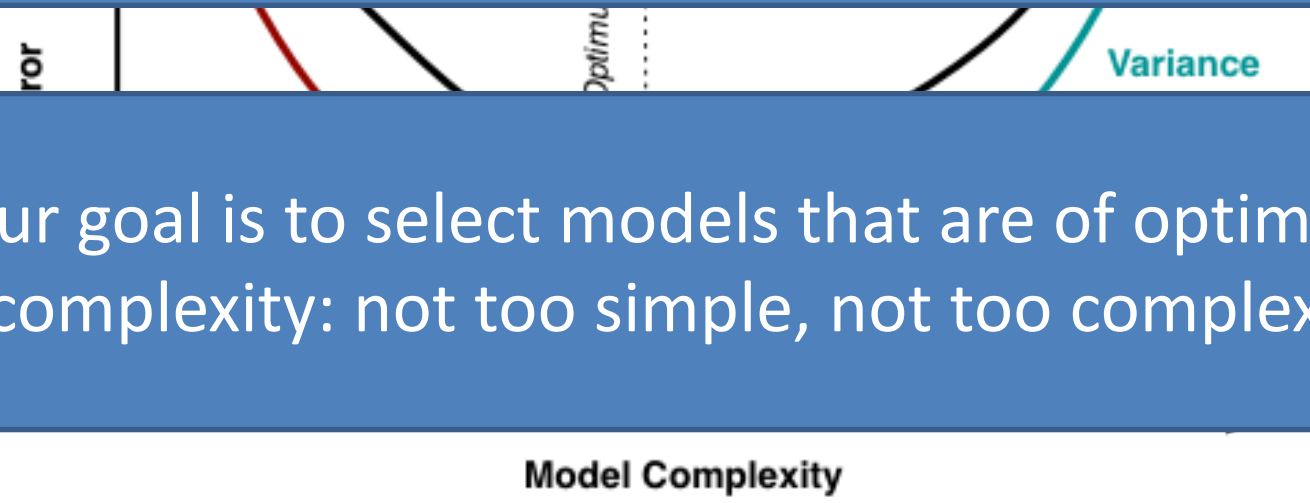
- Expect error = Bias^2 + Variance + Noise; where:
  - Bias: difference between the expected or average prediction of our model and the correct value which we are trying to predict.
  - Variance: variability of model prediction for a given data point or a value which tells us spread of our data.
  - Noise is irreducible error caused by data.
- We first discuss bias and variance

Bias-Variance-Model Complexity

- Complicated models have low bias and high variance:
  - Low bias: complicated models capture a lot of feature in the training data, perform well on training
  - High variance: Testing set may not have the same feature
  - Overfitting!
- Simple models have low variance and high bias:
  - Just the other way round for complicated models
  - Underfitting!

Bias-Variance-Model Complexity

Core of bias-variance tradeoff: You cannot minimize bias and variance simultaneously



Our goal is to select models that are of optimal complexity: not too simple, not too complex

What is “complex” model

- Complex models are those ”flexible” models that can fit the data as well as possible.
- E.g.
  - Polynomial models with higher degrees. They can fit data that lower degree model cannot.
  - Models with larger parameters. They can fit data that model with small parameters cannot do.

Just Right: Regression with degree=4

Notes

- If bias is high, we can use more complex model
  - But this increases variance
- Rest of this lecture: How to reduce variance AND keeping bias at a low value?
- Larger training dataset reduces variance
  - Training and testing data can be more “similar”
- Noise is irreducible on same data
  - But ”cleaner” dataset reduces noise.
  - That’s why we need cleansing

Outline

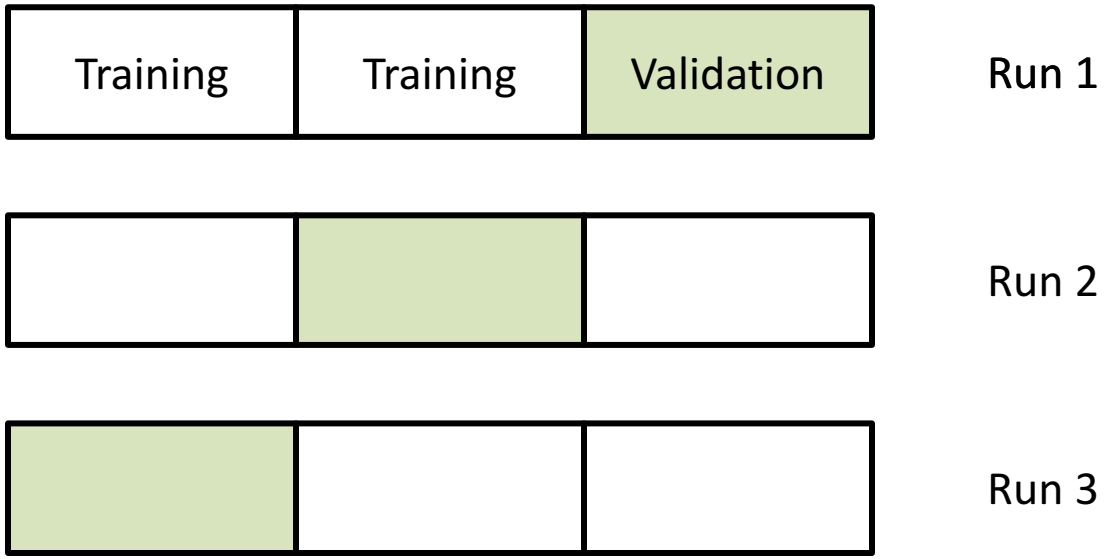
- Generalization of models: How is my model performing on unseen data?
  - Overfitting vs Underfitting
  - Why do overfitting & underfitting happen?
- Selecting good models
  - Cross validation
- Improving the models
  - Regularization
  - Ensemble learning

Intuitive Solution: Validation

- Split training data into training and validation data, use validation data as testing data.
  - If model generalize well on validation data, then should also generalize well on testing data.
  - Disadvantage: wasting part of original training data – smaller dataset increases the chance of overfitting!

Cross Validation

- Partition training data into several groups.
- Repeat: 1 group as validation set, train new model
- Performance metric: error on validation data.



K-fold Cross Validation

- K-fold cross validation:
  - Equally split data in k folds.
  - Each time use 1 fold as validation data, rest as training data, until every fold has been used as validation set once.
  - Previous page: 3-fold cross validation
  - Typically select K=5 or 10, but no formal rule

More on Cross Validation

- K-fold can be used for large dataset.
- Leave-one-out can be used when dataset is small.
  - Use only 1 sample for validation, rest for training.
- Select models with cross-validation
  - Use cross validation to evaluate performance of different models
  - Select the best model among candidates

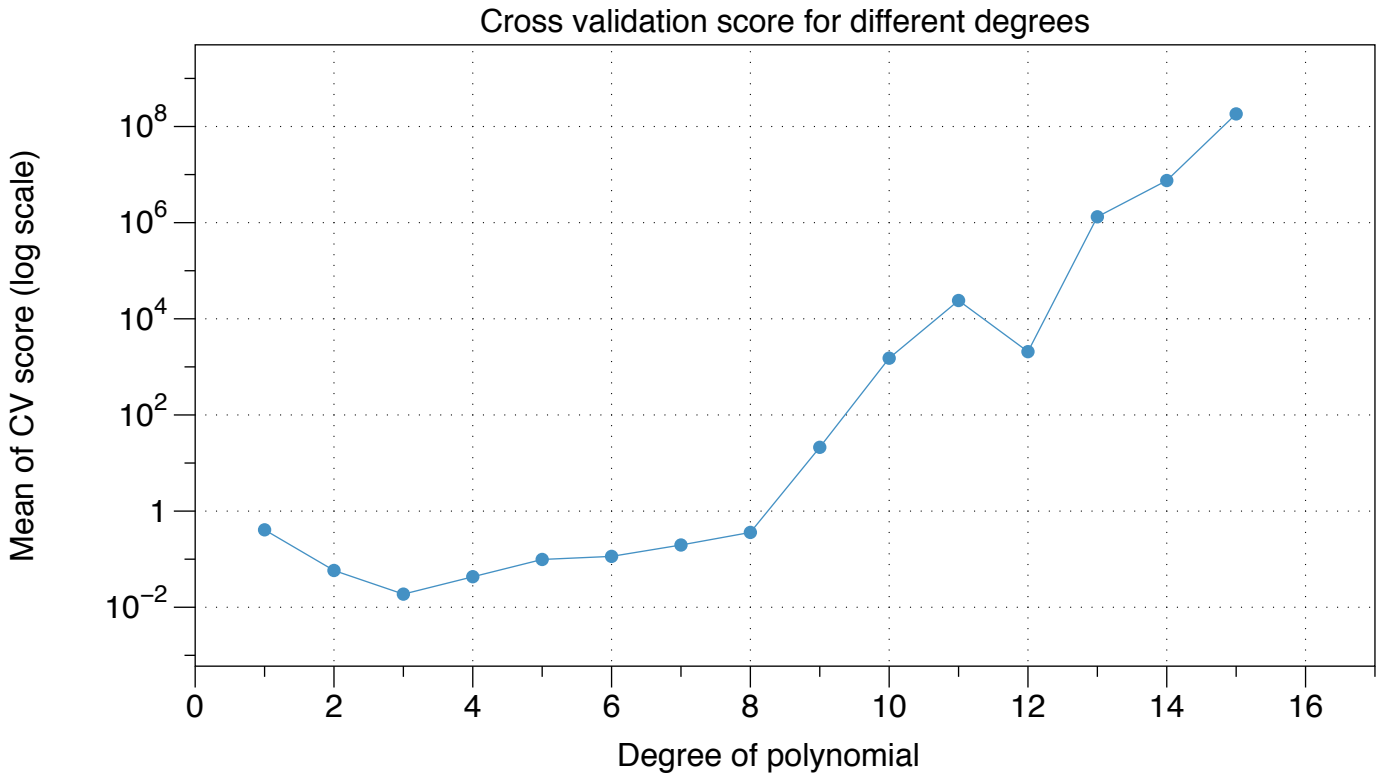
In-Class Programming Exercise

- Can you download the simulation data from canvas, and run cross validation on our simulation data?
- Use 10-fold cross validation. Get score for degree 1, 4, 15. (change “degree” parameter would do)
- Use cross_val_score at the suggested block

Degree	Score (the small the better)
1	0.4077
4	0.0432
15	182815433

This score is the inverse value of output

10-fold CV on Simulation Data



From cross validation, degree 3 gives best model

Outline

- Generalization of models: How is my model performing on unseen data?
  - Overfitting vs Underfitting
  - Why do overfitting & underfitting happen?
- Selecting good models
  - Cross validation selects model. How can we improve model?
- Improving the models
  - Regularization: penalize complex models
  - Ensemble learning: use multiple models

Regularization

- Regularization: Prevent overfitting by reducing "flexibility" of the model.
- As discussed previously, complex models may overfit.
  - Add regularization terms to reduce model complexity.

$$\min_f \sum_{i=1}^{n} l(f(x_i), y_i) + \lambda R(f)$$

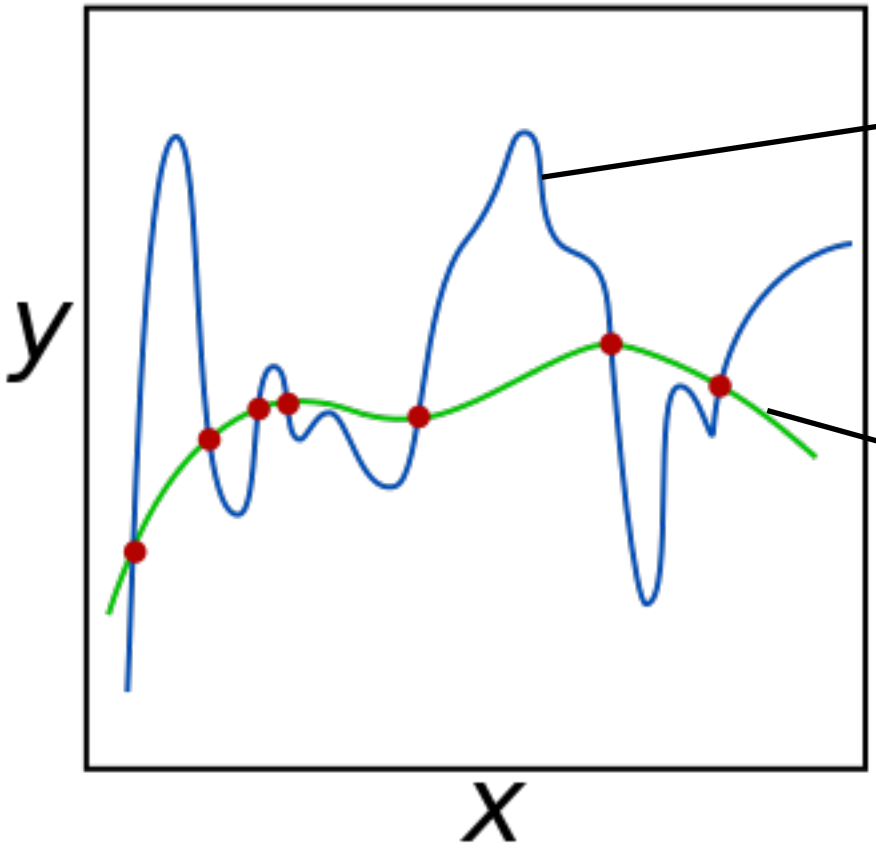
Loss

Regularization term: impose a penalty on complexity of f
e.g. sum-of-squares of all parameters

Example: L2 Regularization

- $\min_{f,w} \sum_{i=1}^{n} l(f(x_i), y_i) + \lambda \sum_{j=1}^{m} (w_j)^2$, where (w) are parameters of function f.
- Prevent parameter having too large absolute values: restrict flexibility of model → reduce variance → prevent overfitting.

Visualized Regularization



Blue model too complex, can be overfitting

Use regularization to make model simpler, as green model

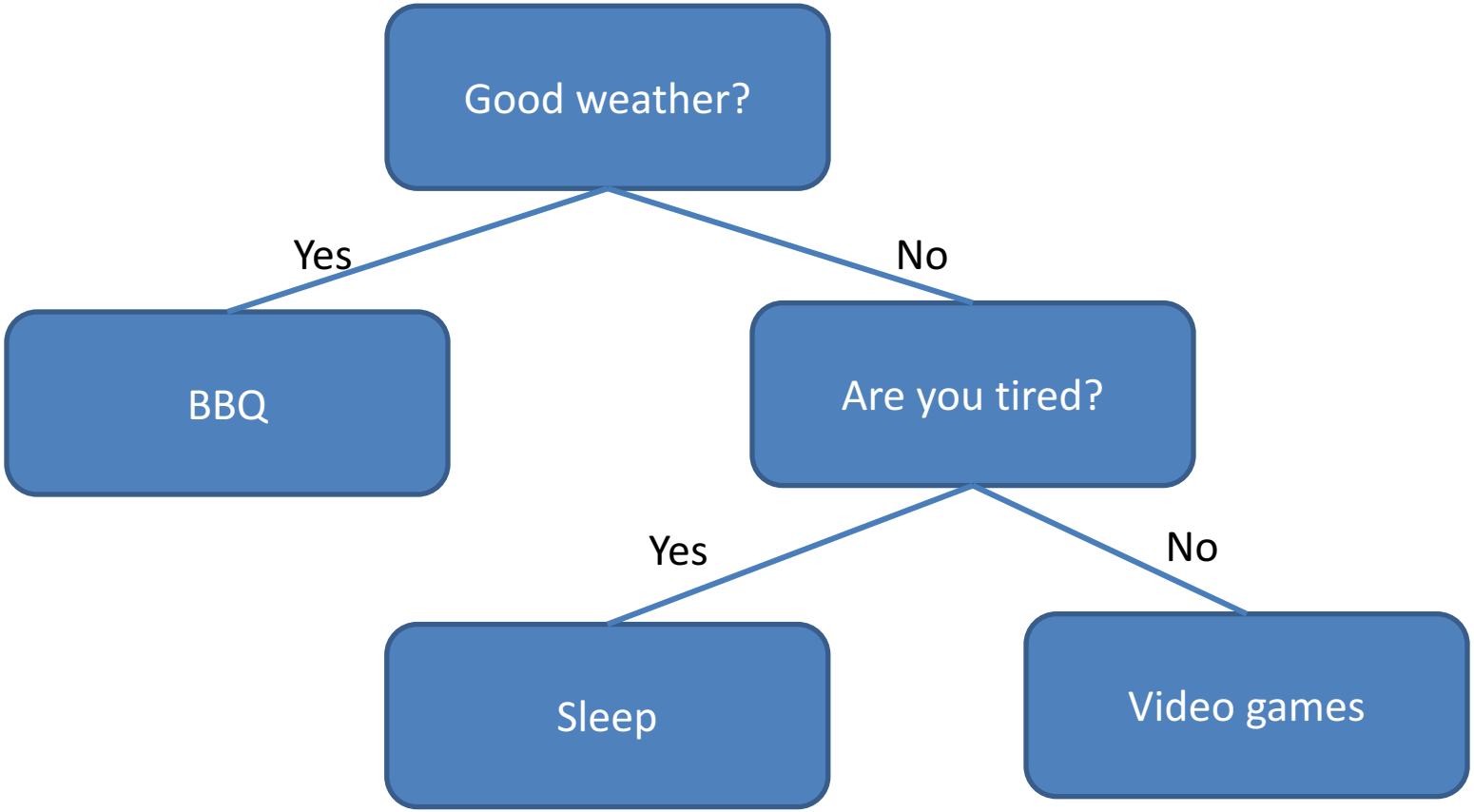
Outline

- Generalization of models: How is my model performing on unseen data?
  - Overfitting vs Underfitting
  - Why do overfitting & underfitting happen?
- Selecting good models
  - Cross validation
- Improving the models
  - Regularization: penalize complex models
  - Ensemble learning: use multiple models

Review: Decision Trees

- Represents a data on a tree such that:
  - Each internal node: test one attribute Xi
  - Each branch from a node: selects one value for Xi
    - If the value is continuous, select value against a threshold
    - E.g., Over 90% score: A range; Below 90%: B or below
  - Each leaf node: predict Y

Example


Good weather?
Yes
No
BBQ
Are you tired?
Yes
No
Sleep
Video games

Problems of Decision Trees

- Standard decision trees can achieve low bias
  - Training set error can be zero! You can always train to the last branch
  - Large variance: overfitting
- Early stopping with fixed nodes or fixed depth:
  - May incur high bias
- We use decision trees as example for the rest of this lecture.

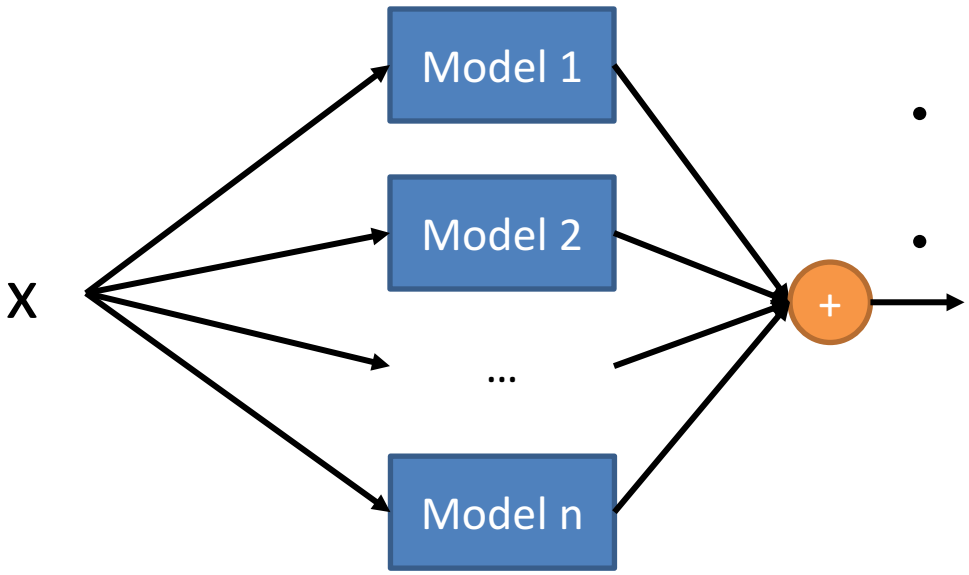
Ensemble Learning: Intuitive Explain

- Mix different models: Use multiple models to complement each other.
- “wisdom of the masses exceeds that of any individual” or “三个臭皮匠赛过诸葛亮” in Chinese

Ensemble learning

- We will discuss following ensemble learning methods:
  - Averaging (voting)
  - Bagging
  - Random forest
  - Boosting

Simple Averaging



- For regression: Simply average the results predicted by different trees , can take weighted average
- For classification: Or select most predicted value
- Also called “voting”

Models are trained with the same datasets, this gives many similar models.
How can we train “different” models?

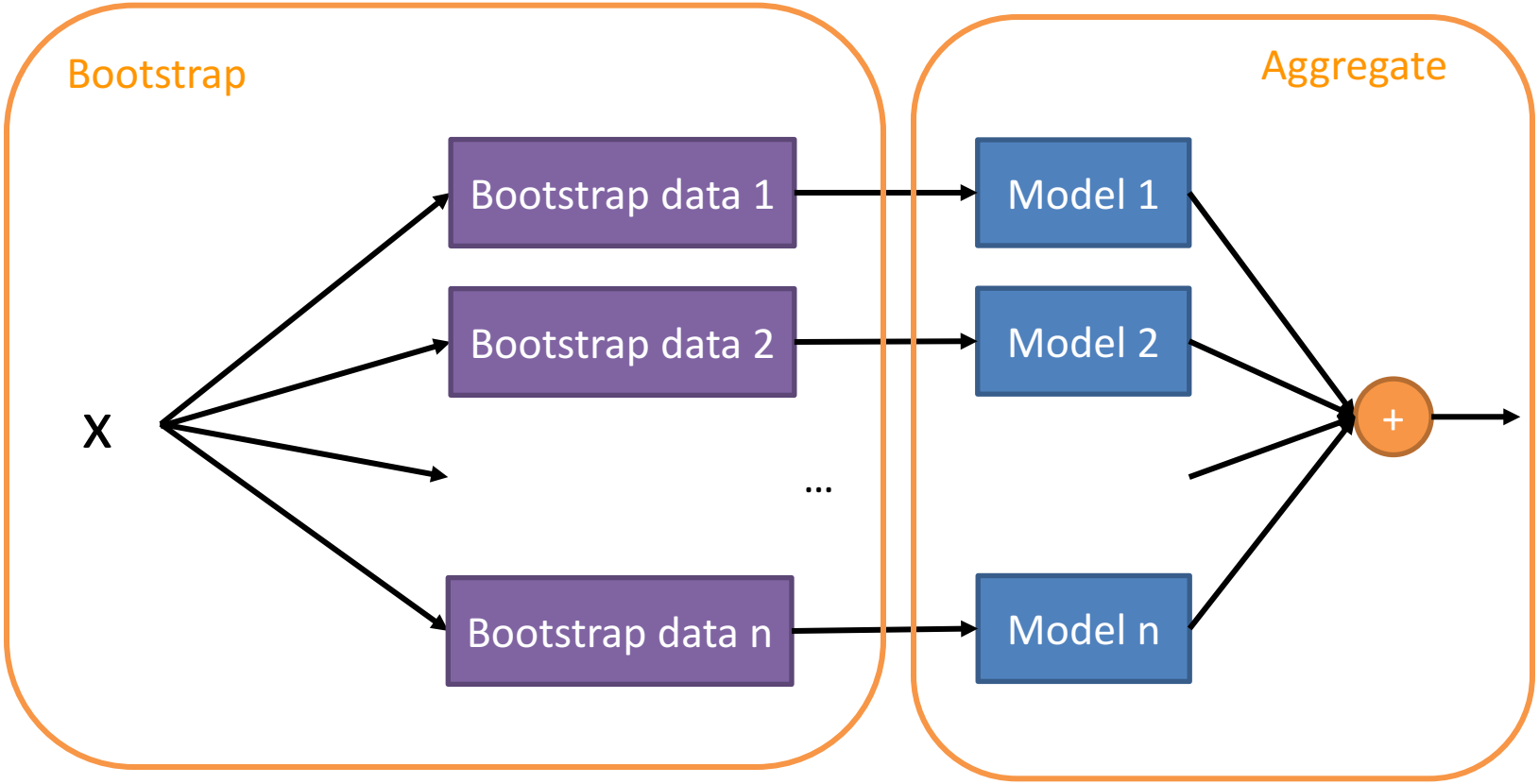
Bagging

?

Bagging

- Bagging = bootstrap aggregating.
- Bootstrap:
  - Draw B bootstrap samples, each with size N, with replacement
  - Train B classifiers, each with a bootstrap sample.
- Aggregating:
  - Voting over classifiers

Bagging



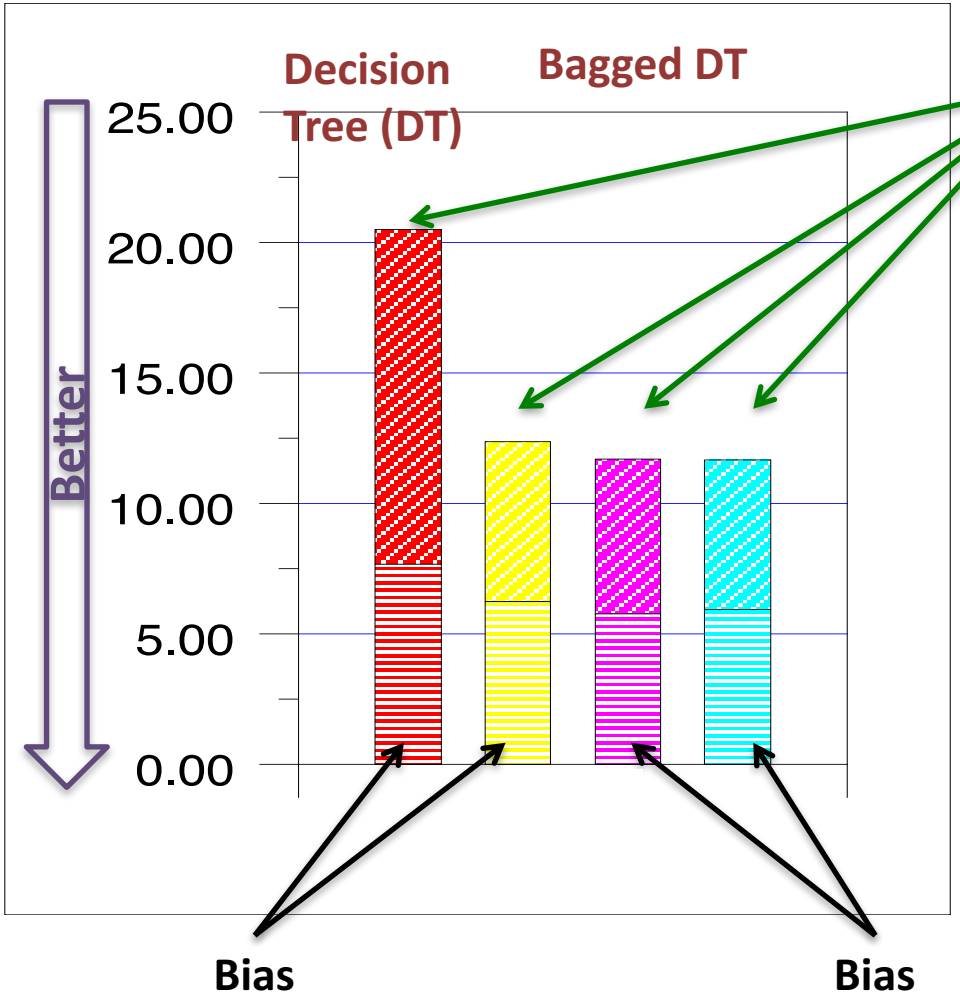
Bagging = Generating different models with different, bootstrapped data + voting

= bootstrap + aggregating

Why Bagging Helps

- Bagging gets similar bias: Data are from re-sampling
- Bagging dramatically reduces the variance of unstable models (e.g., trees)
  - Averaging / voting reduces model variance.
  - Therefore avoid overfitting
  - More difficult to interpret the model

Bagging Reduces Variance



Bagging keeps similar bias, but reduces variance a lot

"An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants"
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

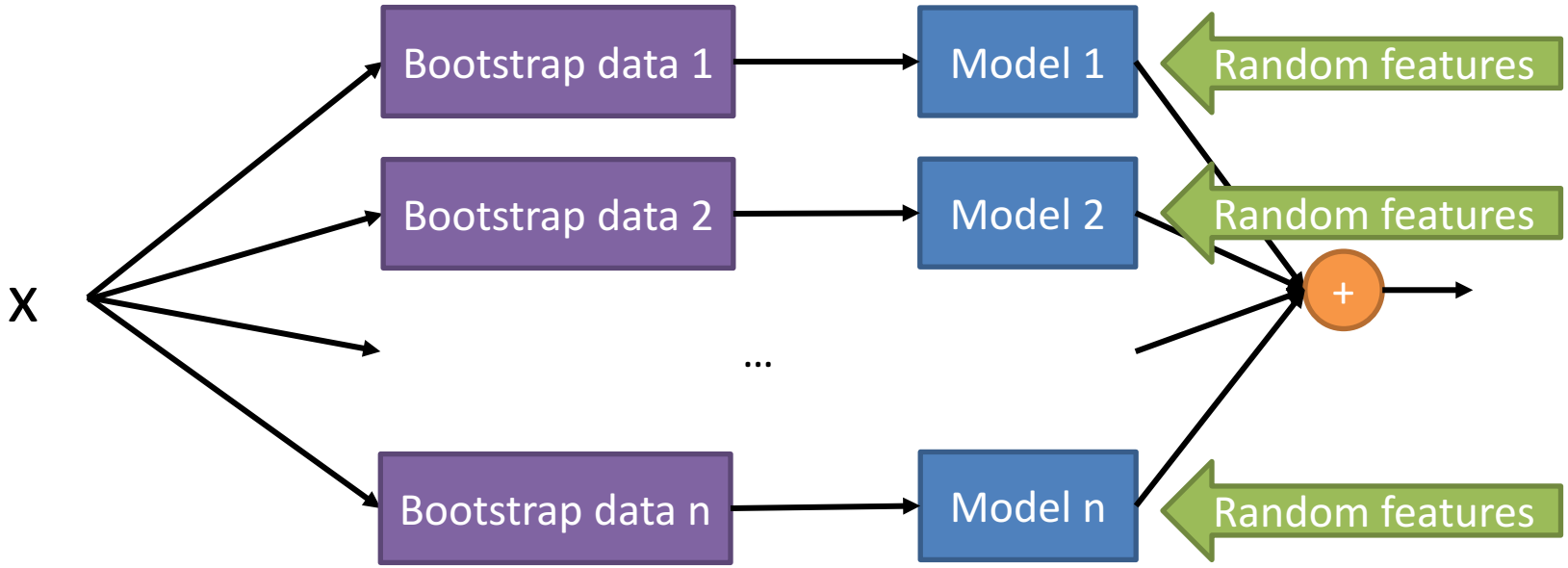
Random Forest

- Refinement of bagged trees.
- Problem: we want the trees to be independent, don't want them to be very similar
  - But bootstrapping data doesn't help that much: still drawn from same dataset.
  - At each tree split: A random sample of m *features* are drawn. Only these m features are considered for splitting
  - Typically m is sqrt(p) or p/3 where p is total number of features.

Random Forest

- Similar prediction method as bagging: Voting or averaging among different models.
- Random forest tries to improve bagging by de-correlating trees.

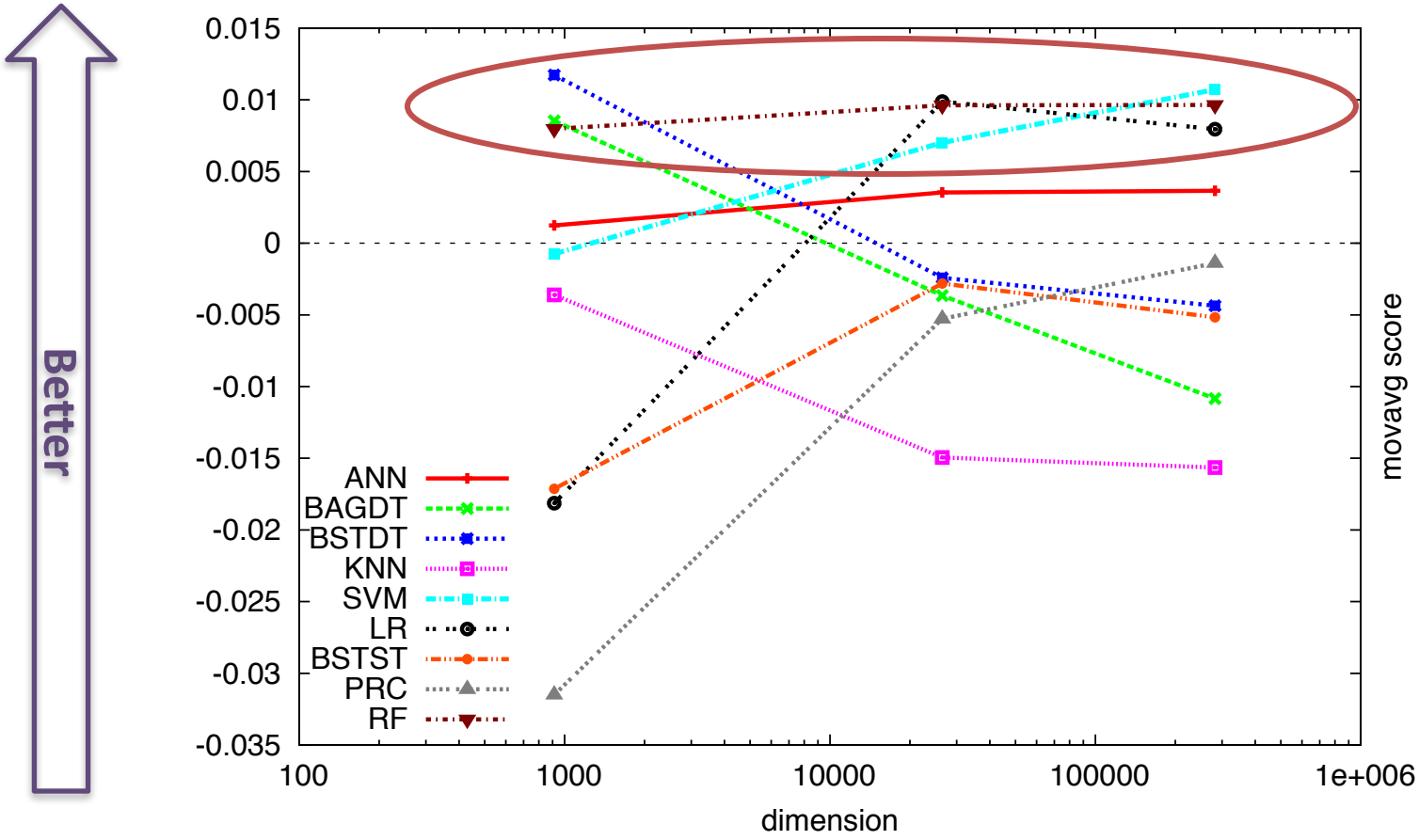
Difference with Bagging



Random forest = Bagging + random subset feature selection at every split

Comparison of Different Models

Random Forest (RF) performs well in every case



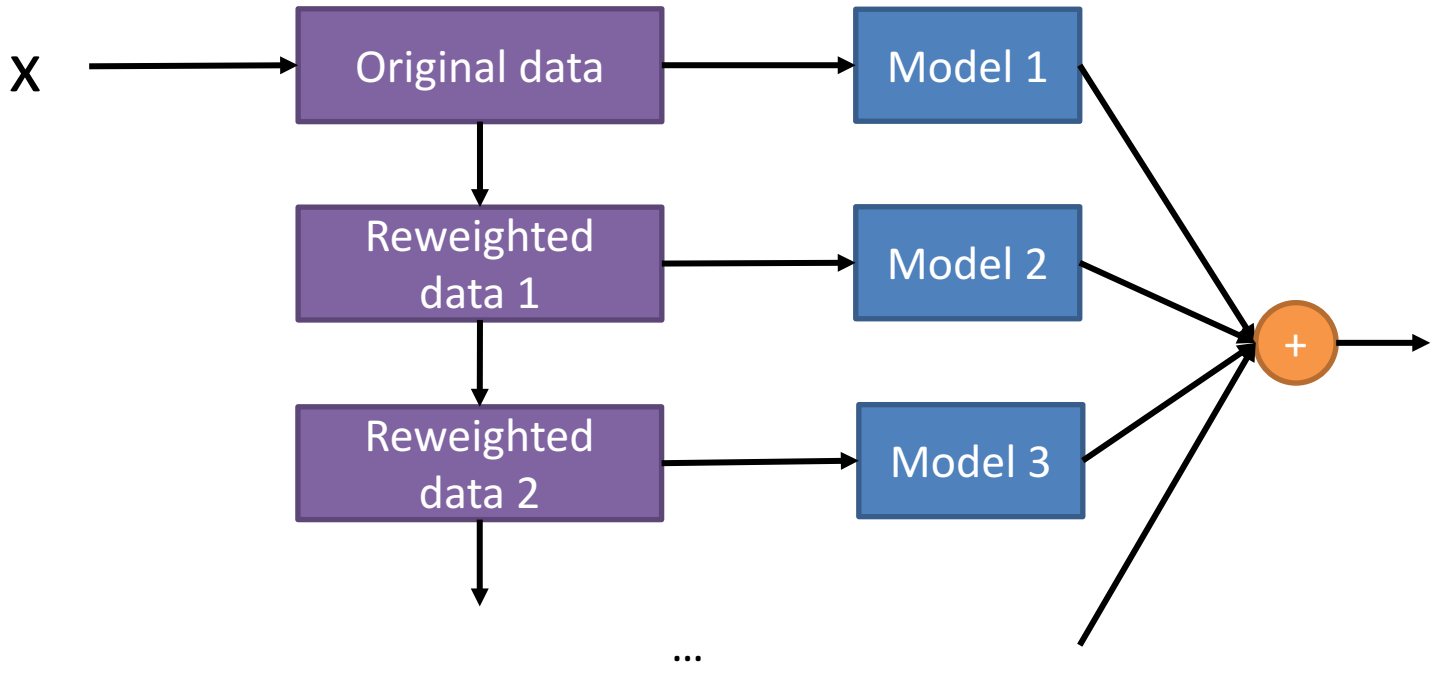
Average performance over many datasets Random Forests perform the best

"An Empirical Evaluation of Supervised Learning in High Dimensions"
Caruana, Karampatziakis & Yessenalina, ICML 2008

Boosting

- Random forest & bagging: trees can be trained in parallel.
- Boosting: trees should be trained sequentially.
  - Start with original training sample
  - In each iteration:
    - Train a classifier, update its weight
    - Update the weight of samples in training data.
  - Repeat the iteration with weight-updated data.
- Final classifier: weighted average of models.

Boosting

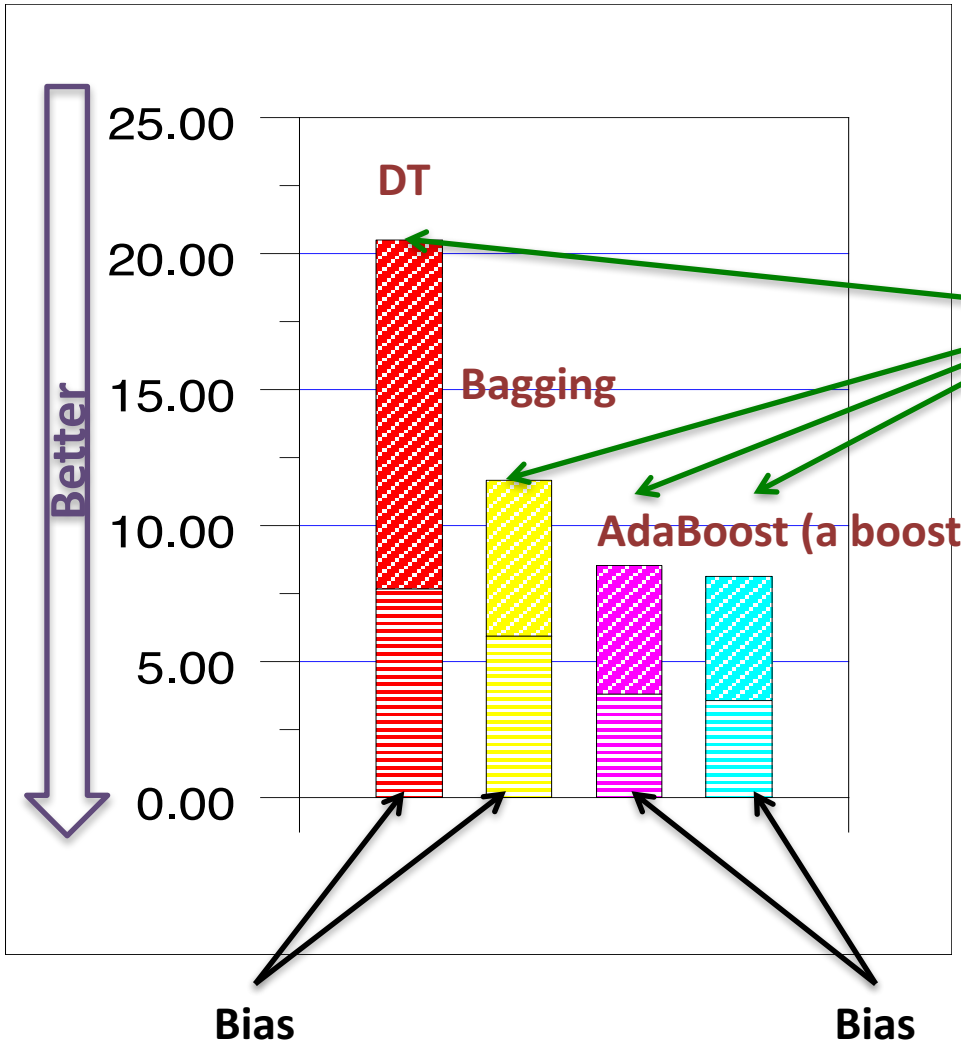


Boosting: Data re-weighting + Sequential training + voting

Advantages

- Boosting reduces both bias and variance!
- Reduce bias: boosting re-assign weights at each iteration → next model pays more attention to previous “wrong” training data.
- Reduce variance: boosting uses averaging, same as bagging & random forest.

Boosting Reduces Bias and Variance



Boosting often uses weak models
E.g, "shallow" decision trees
Weak models have lower variance

"An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants"
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

Ensemble Learning Summary

- Mix different models: Use multiple models to complement each other.

Method	Train sequentially or in parallel?	How to generate different models?	Reduces bias or variance?
Bagging	Parallel	Bootstrap data	Variance
Random Forest	Parallel	Bootstrap + random subset of features at splitting	Variance
Boosting	Sequential	Reweight training data	Bias and variance

Wrap-up

- How do models generalize:
  - Underfitting vs overfitting
- Why do underfitting & overfitting happen?
  - Bias-variance tradeoff
- Select models? Cross validation
- Improve models? Regularization, ensemble learning
- Next lecture: unsupervised learning methods