Data Mining & Machine Learning

CS37300
Purdue University

Oct 2, 2023

Today's topics

- Finish Linear regression
- What is "error"?
- Overfitting
- Model Selection / hyper-parameter tuning

Setup

- Data $\{x_i, y_i\}$ Total *n* data points, $x_i \in R^d, y \in R$.
- $y \sim w^\top x$
- Cost function:

$$\min_w \sum_i |y_i - w^\top x_i|^2$$

- Interpretation: Minimize the "residuals" $(y_i - w^\top x_i)$ i.e. leftover after fitting the model

Last time

- Closed form solution
- Calculating inverses to get the closed form solution
- Extension to GLMs
- Interpretability
- Evaluation

A taste of evaluation

- How do we check whether the learnt model is reasonable?
- MSE
- Coefficient of determination or R^2 statistic = $\frac{\sum_i (y_i - w^T x_i)\text{^}2.}{\sum_i (y_i - \bar{y})\text{^}2}$
  - Intuitively, ratio of variance explained and total variance

Ridge regression

- Alternate cost function using regularization:
  - $\min_w \sum_i |y_i - w^\top x_i|^2 + \lambda ||w||^2$
- "Want to fit the data but don't want w to be too big"
- Also has closed form solution: $w^\star = (X^T X + \lambda I)^{-1} X^T y$
  - The inner matrix is always invertible
- Bias-variance tradeoff (more on this later!)

Sparse modeling

- Instead of using all the features, start with using a few
- Iteratively add or remove features that improve “performance” (e.g. MSE or R^2)
- Alternatively, use LASSO

$$\min_w \sum_i |y_i - w^\top x_i|^2 + \lambda \left|\left|w\right|\right|_1 \quad ,$$

where $\left|\left|w\right|\right|_1 = \sum_i |w|$ is the L1 norm of w.

- Commonly used: Elastic net

$$\min_w \sum_i |y_i - w^\top x_i|^2 + \lambda_1 \left|\left|w\right|\right|^2 + \lambda_2 \left|\left|w\right|\right|_1$$

Error

- Accuracy: Correct predictions / total predictions
  - Problem: Base Rate
  - Suppose a parts quality control is 99.9% accurate at separating good from defective parts

Error

- Accuracy: Correct predictions / total predictions
  - Problem: Base Rate
  - Suppose a parts quality control is 99.9% accurate at separating good from defective parts
  - But what if 99.99% of parts are good?

Error

- Accuracy: Correct predictions / total predictions
  - Problem: Base Rate
  - Suppose a parts quality control is 99.9% accurate at separating good from defective parts
  - But what if 99.99% of parts are good?
- Would you use this classifier?

Error

- Accuracy: Correct predictions / total predictions
  - Problem: Base Rate
  - Suppose a parts quality control is 99.9% accurate at separating good from defective parts
  - But what if 99.99% of parts are good?
- Would you use this classifier?
- Confusion Matrix:

	Classified Good	Classified Defective
Actually good	990 (True Positive)	9 (False Negative)
Actually defective	1 (False Positive)	0 (True Negative)

  - Good: TP, TN
  - Bad: FP, FN

Some typical measures

- Accuracy: (TP+TN) / (TP+TN+FP+FN)
  - Error (essentially inverse): (FP+FN) / (TP+TN+FP+FN)

Some typical measures

- Accuracy: (TP+TN) / (TP+TN+FP+FN)
  - Error (essentially inverse): (FP+FN) / (TP+TN+FP+FN)
- Precision/Recall
  - Precision: TP / (TP + FP) = % correctly predicted positive
  - Recall: TP / (TP + FN) = % positives captured correctly

Some typical measures

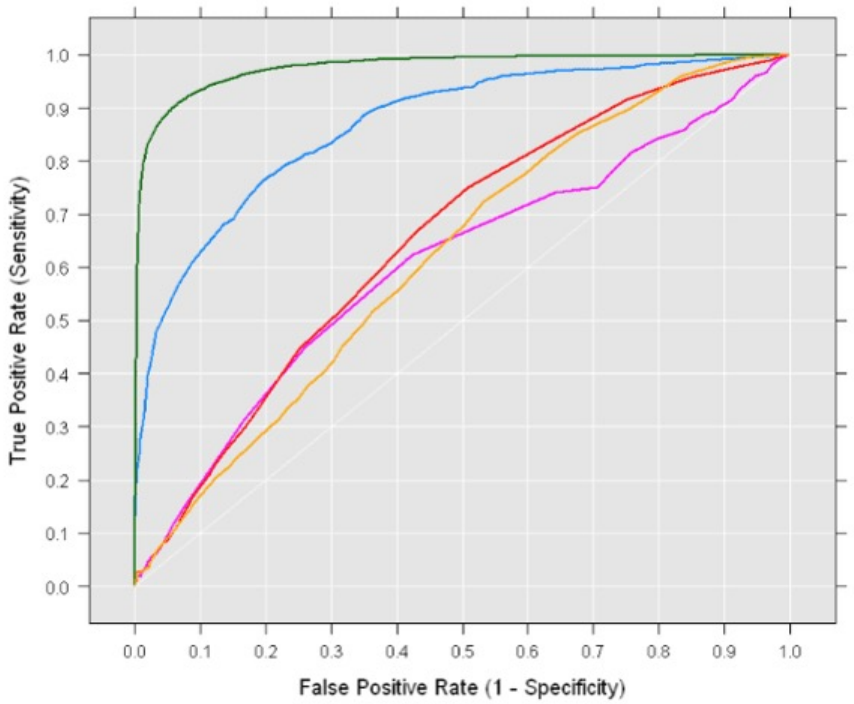
- Accuracy: (TP+TN) / (TP+TN+FP+FN)
  - Error (essentially inverse): (FP+FN) / (TP+TN+FP+FN)
- Precision/Recall
  - Precision: TP / (TP + FP) = % correctly predicted positive
  - Recall: TP / (TP + FN) = % positives captured correctly
  - F1 Score: $\frac{2}{\frac{1}{precision}+\frac{1}{recall}}$

Some typical measures

- Accuracy: (TP+TN) / (TP+TN+FP+FN)
  - Error (essentially inverse): (FP+FN) / (TP+TN+FP+FN)
- Precision/Recall
  - Precision: TP / (TP + FP) = % correctly predicted positive
  - Recall: TP / (TP + FN) = % positives captured correctly
  - F1 Score: $\frac{2}{\frac{1}{precision}+\frac{1}{recall}}$
- Regression Error Measures:
  - Mean Absolute Error: $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$
  - Mean Squared Error: $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

Some typical measures

- Area Under the Curve (AUC): Used to measure a tradeoff between true positives and false positives
  - True Positive Rate / Recall / Sensitivity: TP / (TP+FN)
  - False Positive Rate (1- specificity): FP / (FP + TN)
  - Receiver Operating Curve (ROC): Tradeoff between TPR and FPR
- Better classifiers have a higher "area under the curve"



Bias/Variance trade-off

Bias/Variance

- **Easy to define very powerful learners**
  - Move to a higher dimension
  - Pick small K in KNN
  - Large decision trees
- ***Is it always a good idea?***

Bias/Variance

- **Easy to define very powerful learners**
  - Move to a higher dimension
  - Pick small K in KNN
  - Large decision trees
- ***Is it always a good idea?***
- Simple (simplistic) approach:
- **Option 1:** Learning doesn't work– *our model is not powerful enough!*
  - *Move to a richer representation*

Bias/Variance

- **Easy to define very powerful learners**
  - Move to a higher dimension
  - Pick small K in KNN
  - Large decision trees
- ***Is it always a good idea?***
- Simple (simplistic) approach:
- **Option 1:** Learning doesn't work– *our model is not powerful enough!*
  - *Move to a richer representation*
- **Option 2:** Learning doesn't work – *our model overfits!*
  - *Get more data! Remove features! Regularize!*

Bias/Variance

- **Easy to define very powerful learners**
  - Move to a higher dimension
  - Pick small K in KNN
  - Large decision trees
- ***Is it always a good idea?***
- Simple (simplistic) approach:
- **Option 1:** Learning doesn't work– *our model is not powerful enough!*
  - *Move to a richer representation*
- **Option 2:** Learning doesn't work – *our model overfits!*
  - *Get more data! Remove features! Regularize!*

How can we tell?

Overfitting

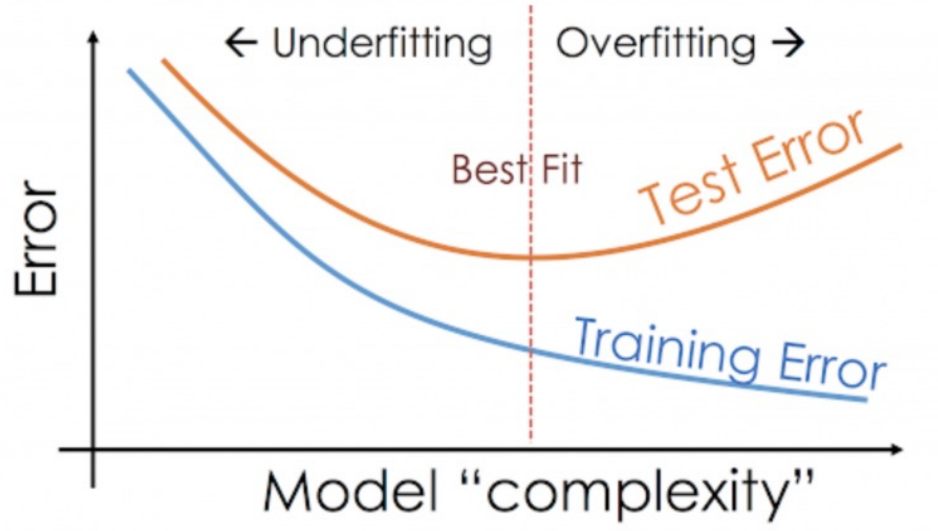
- Overfitting is when the training error rate is small, but the true error rate is not nearly as small

Overfitting

- Overfitting is when the training error rate is small, but the true error rate is not nearly as small
- Overfitting can occur when:
  - Your model is too expressive
  - Your data set is not very large

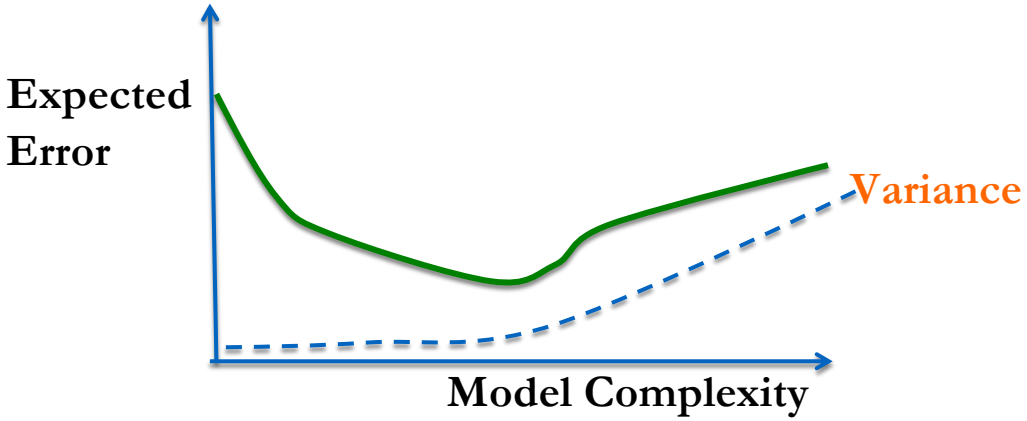
Overfitting

- Overfitting is when the training error rate is small, but the true error rate is not nearly as small
- Overfitting can occur when:
  - Your model is too expressive
  - Your data set is not very large



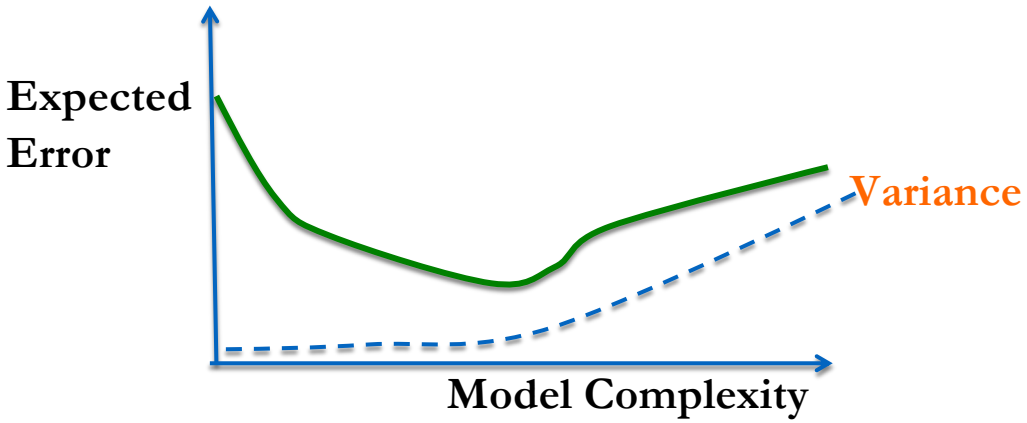
- For a given data set size, there is a trade-off between
  - Model expressiveness (more expressive → smaller training error rate)
  - Overfitting (more expressive → true error rate > training error rate)

The "Variance" of the Learner



How susceptible is the learner to changes in the particular training data?
(i.e., if we draw a new training set from P(x,y), would the learner output a similar solution?)

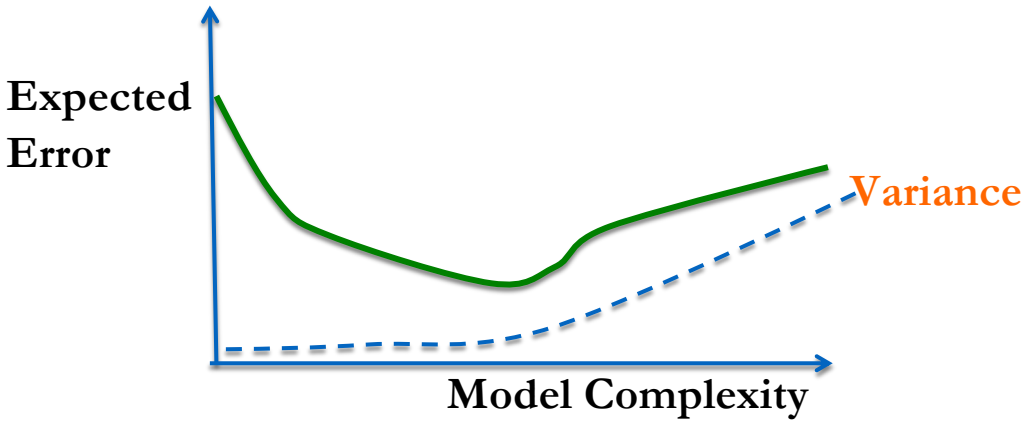
The "Variance" of the Learner



How susceptible is the learner to changes in the particular training data? (i.e., if we draw a new training set from P(x,y), would the learner output a similar solution?)

- You can think about the testing data as another sample from P(x,y)

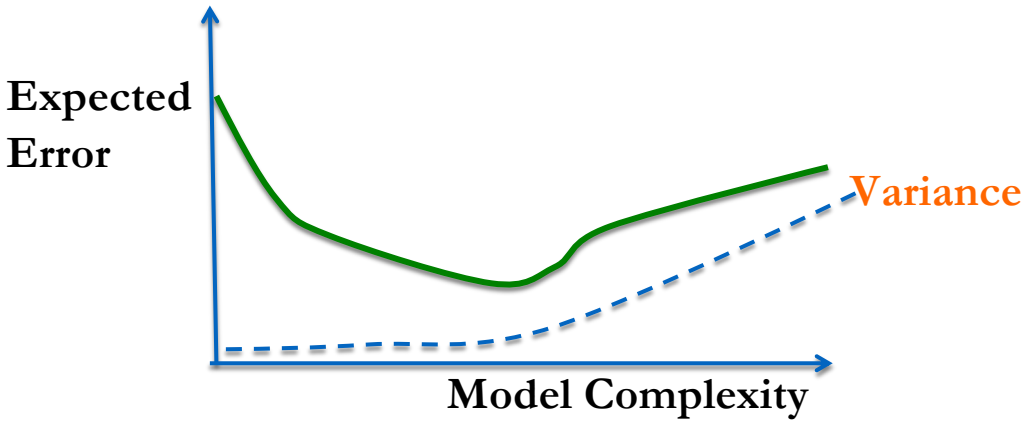
The "Variance" of the Learner



How susceptible is the learner to changes in the particular training data?
(i.e., if we draw a new training set from P(x,y), would the learner output a similar solution?)

- You can think about the testing data as another sample from P(x,y)
- Low training error but high test error indicates high "variance" of solutions

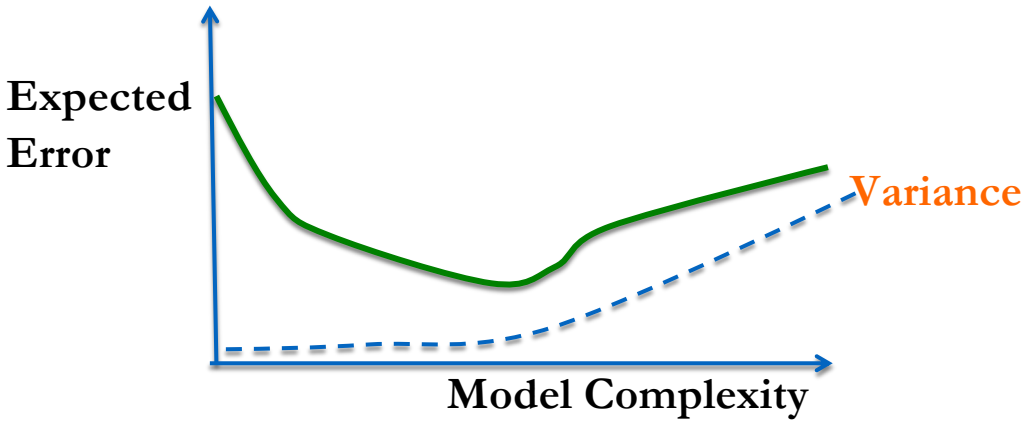
The "Variance" of the Learner



How susceptible is the learner to changes in the particular training data?
(i.e., if we draw a new training set from P(x,y), would the learner output a similar solution?)

- You can think about the testing data as another sample from P(x,y)
- Low training error but high test error indicates high "variance" of solutions
  - *Overfitting*

The "Variance" of the Learner

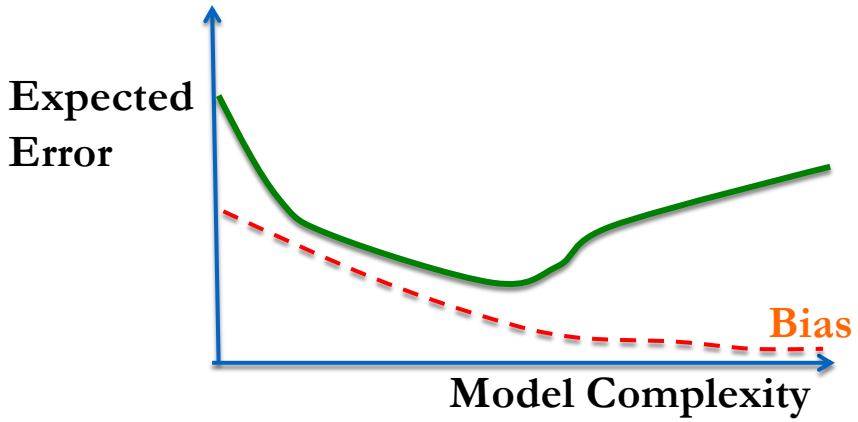


How susceptible is the learner to changes in the particular training data?
(i.e., if we draw a new training set from P(x,y), would the learner output a similar solution?)

- You can think about the testing data as another sample from P(x,y)
- Low training error but high test error indicates high "variance" of solutions
  - *Overfitting*

Variance increases with model complexity!

Bias of the Learner

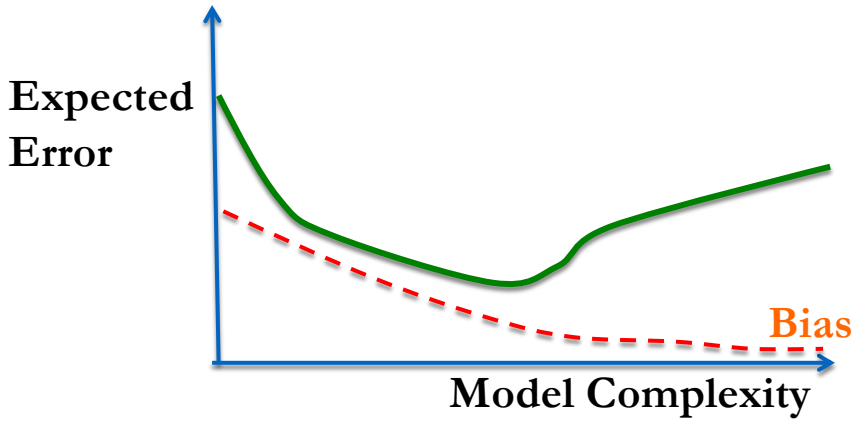


Simpler models have higher "bias": "Trust data less"
meaning they strongly favor specific types of solutions

How likely is the learner to identify the target hypothesis?

- What will happen if we test on a different sample?
  - Training error and testing error will be similar
  - If both training error and testing error are high: **Underfitting**

Bias of the Learner



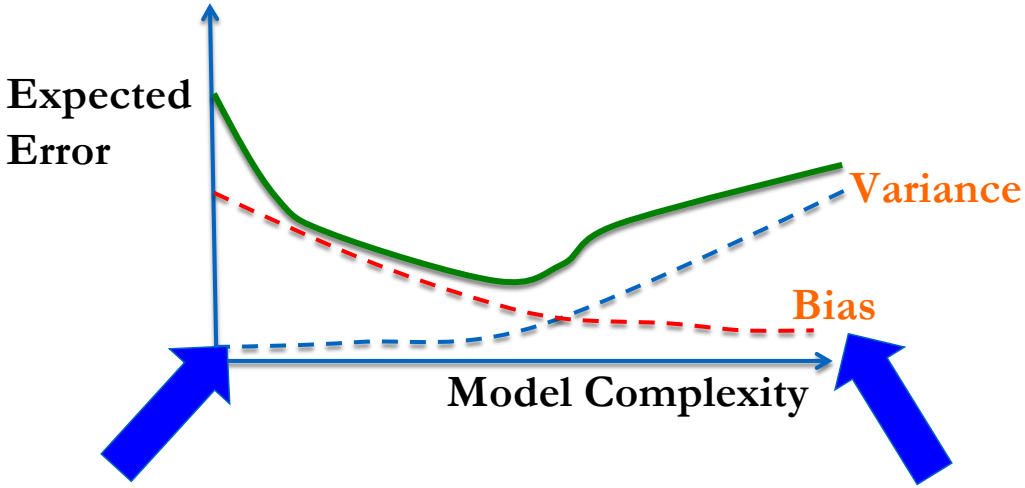
Simpler models have higher "bias": "Trust data less"
meaning they strongly favor specific types of solutions

How likely is the learner to identify the target hypothesis?

- What will happen if we test on a different sample?
  - Training error and testing error will be similar
  - If both training error and testing error are high: **Underfitting**
- **What will happen if we train on a different sample?**

Bias is high when the model is too simple

Model Complexity

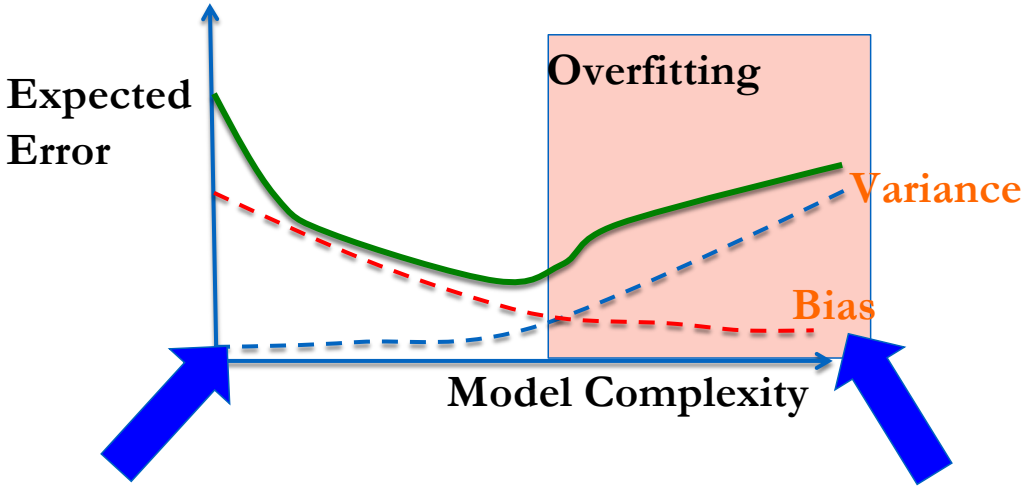


Simple models:
High bias and low variance

Complex models:
High variance and low bias

Expected Error ≈ Bias + Variance

Model Complexity

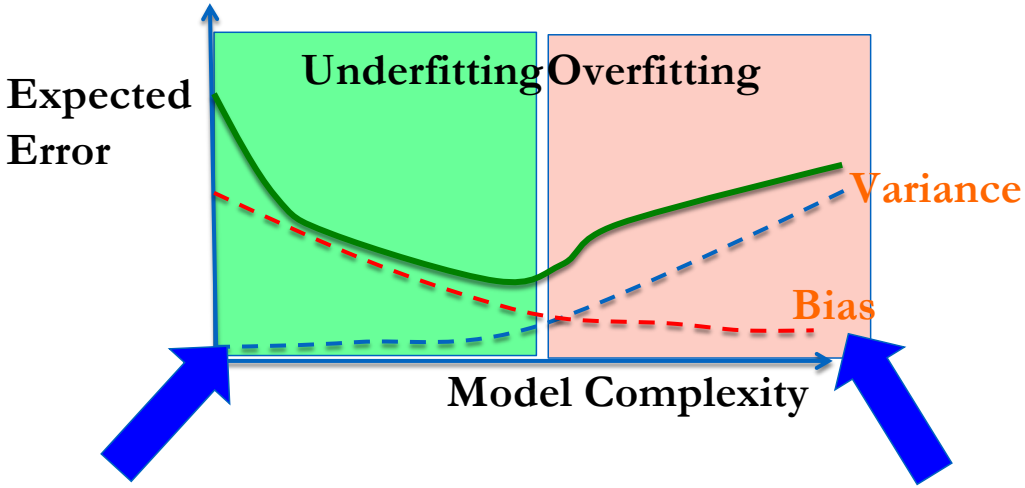


Simple models:
High bias and low variance

Complex models:
High variance and low bias

Expected Error ≈ Bias + Variance

Model Complexity



Simple models:
High bias and low variance

Complex models:
High variance and low bias

Expected Error ≈ Bias + Variance

Bias-Variance Tradeoff in Practice

- We saw that the classification error can be (informally) expressed in terms of bias and variance

Bias-Variance Tradeoff in Practice

- We saw that the classification error can be (informally) expressed in terms of bias and variance
- Reducing the bias and variance can reduce expected error!

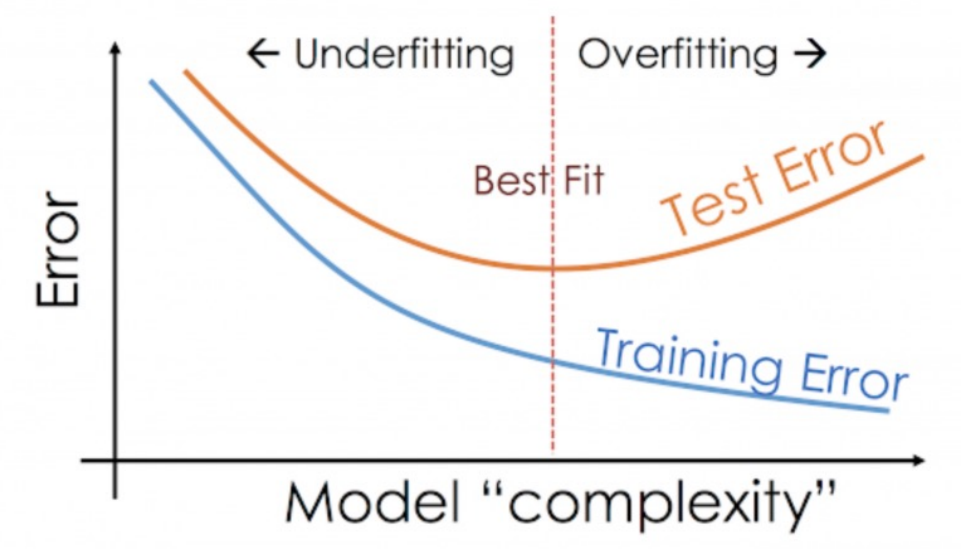
Bias-Variance Tradeoff in Practice

- We saw that the classification error can be (informally) expressed in terms of bias and variance
- Reducing the bias and variance can reduce expected error!
- Different scenarios can lead to different actions for reducing the error
  - **High bias**: add more features
  - **High variance**: simplify the model, add more examples

Bias-Variance Tradeoff in Practice

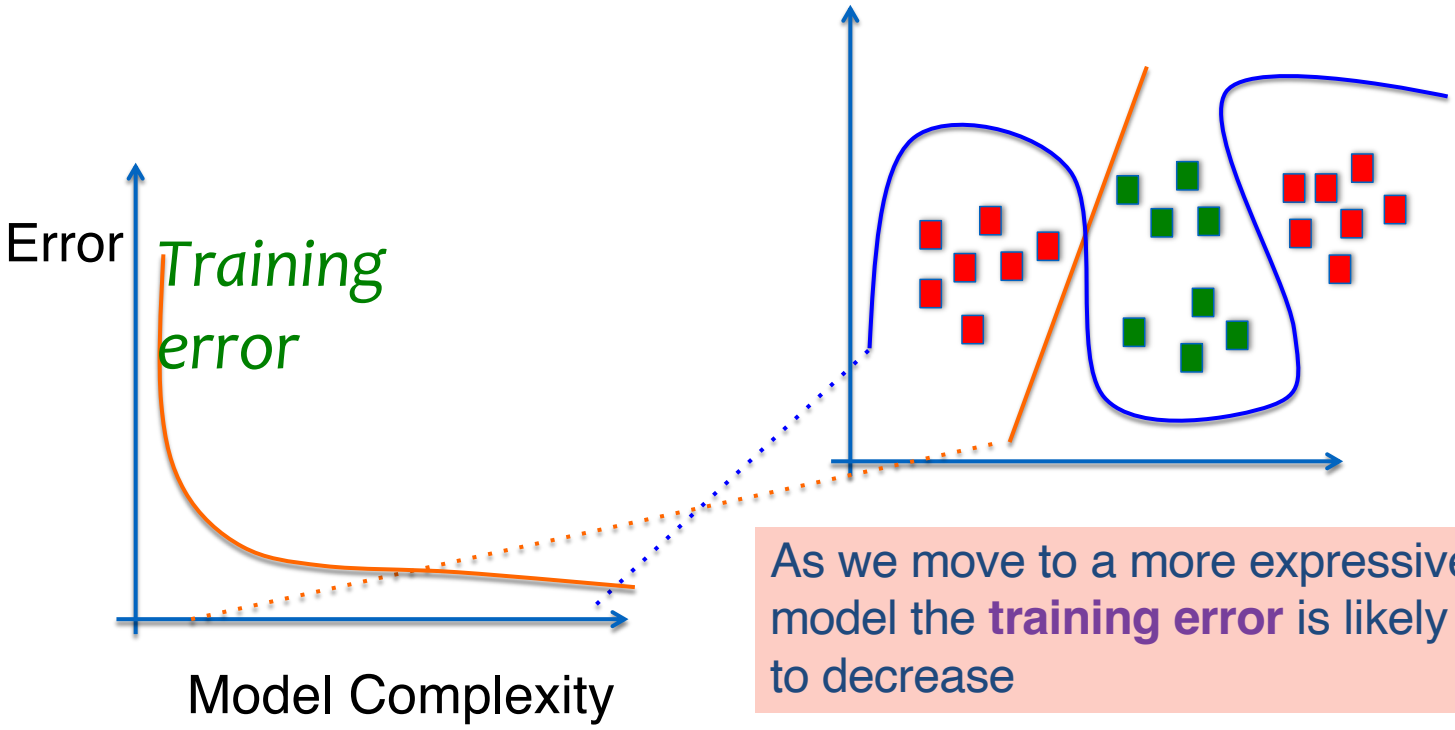
- We saw that the classification error can be (informally) expressed in terms of bias and variance
- Reducing the bias and variance can reduce expected error!
- Different scenarios can lead to different actions for reducing the error
  - **High bias**: add more features
  - **High variance**: simplify the model, add more examples
- *How can we diagnose each one of these scenarios?*

Overfitting



- Question 1: How do we detect overfitting?
- Question 2: How do we prevent / correct overfitting?

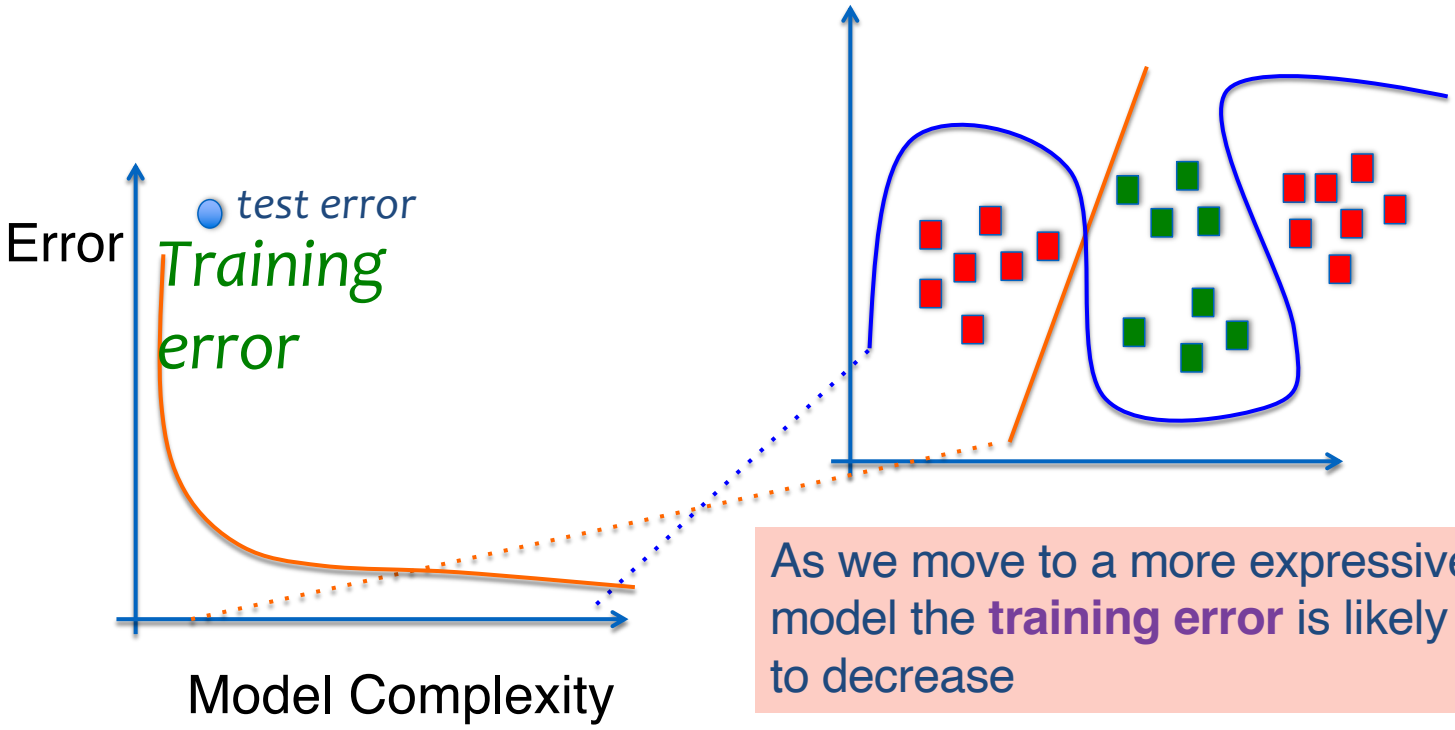
Bias-Variance Analysis



As we move to a more expressive model the **training error** is likely to decrease

Bias-Variance Analysis

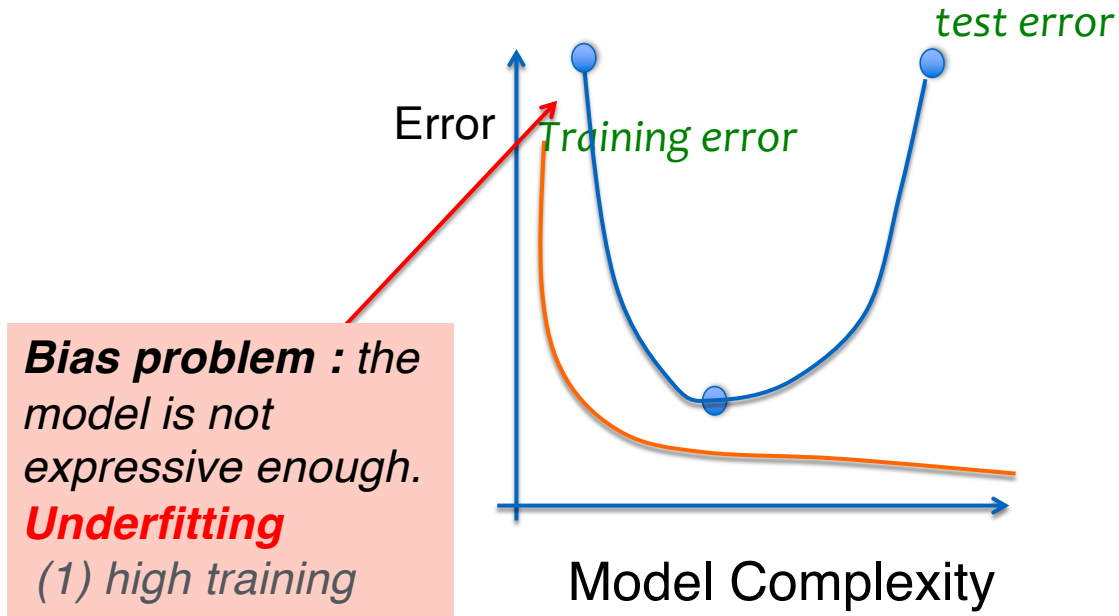
*When we are using a **simple model**, it's very likely we'll underfit resulting in a **high test** error*



As we move to a more expressive model the **training error** is likely to decrease

Bias-Variance Analysis

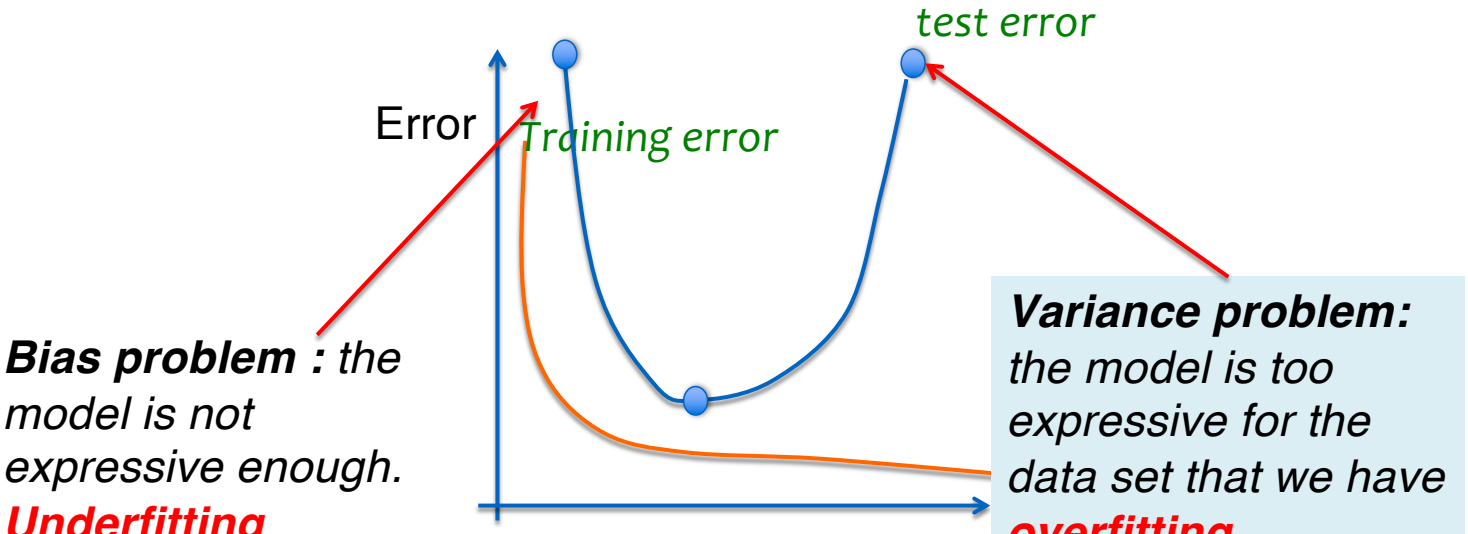
Interpolating over the points: two curves that we can use for ***diagnosis***



Bias problem : *the model is not expressive enough.*
Underfitting
(1) high training error
(2) test error ≈ training error

Bias-Variance Analysis

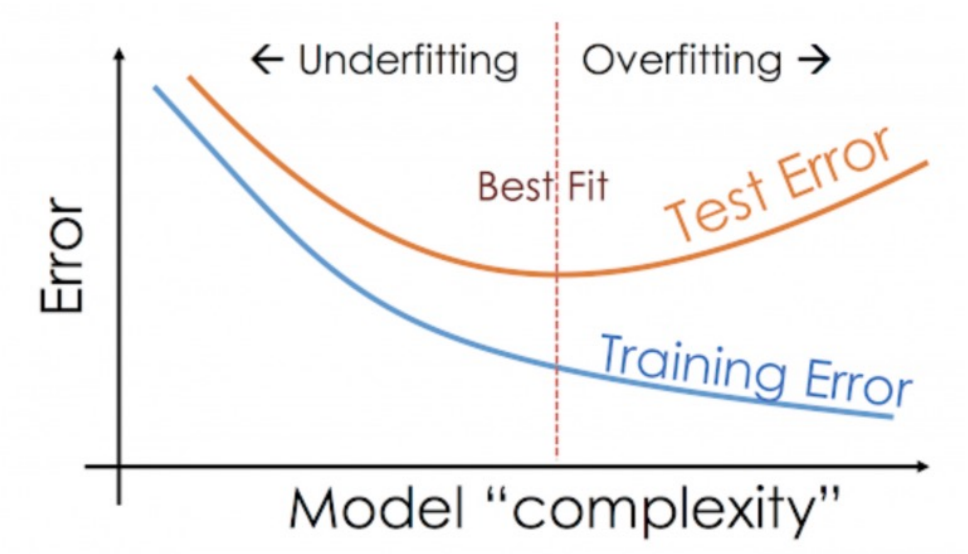
Interpolating over the points: two curves that we can use for ***diagnosis***



Bias problem : *the model is not expressive enough.*
Underfitting
(1) high training error
(2) test error ≈ training error

Variance problem: *the model is too expressive for the data set that we have*
overfitting
(1) low training error,
(2) test error > training error.

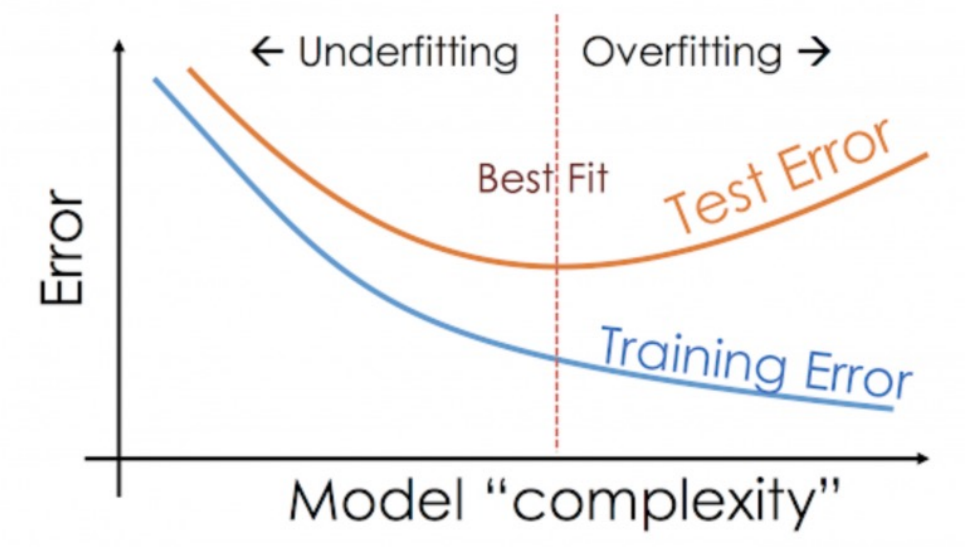
Overfitting



- Question 1: How do we detect overfitting?

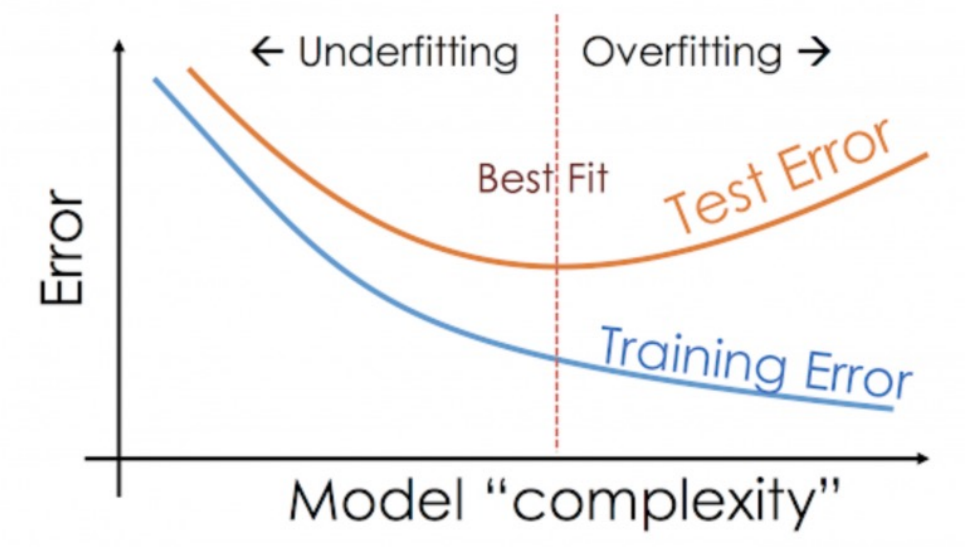
- **Question 2: How do we prevent / correct overfitting?**

Preventing / Correcting Overfitting



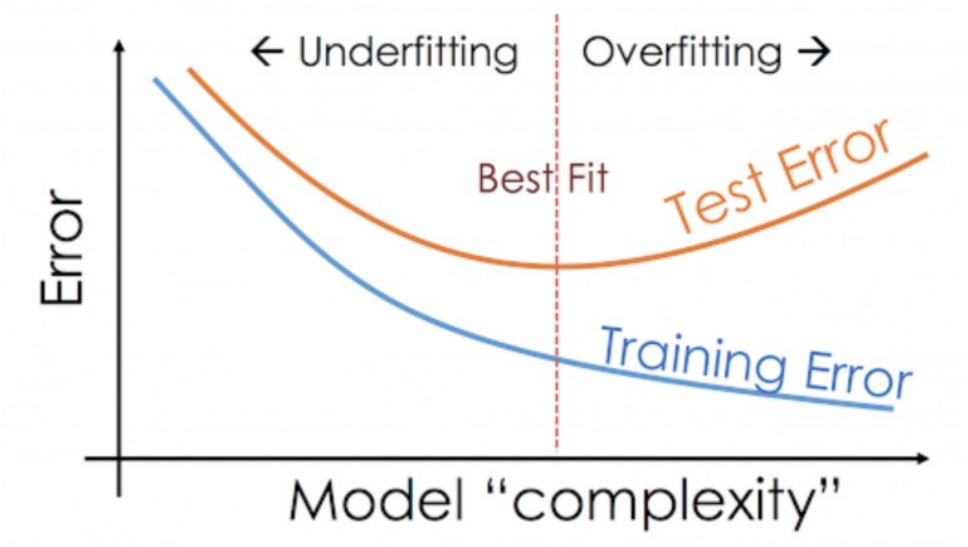
- Model Selection: Find a model that gives good test error

Preventing / Correcting Overfitting



- Model Selection: Find a model that gives good test error
  - Often this is just tuning a "hyper-parameter"
    (e.g., k in kNN, bandwidth in kernel regression, C in Soft-SVM)
  - Sometimes more involved: e.g., post-pruning in decision trees

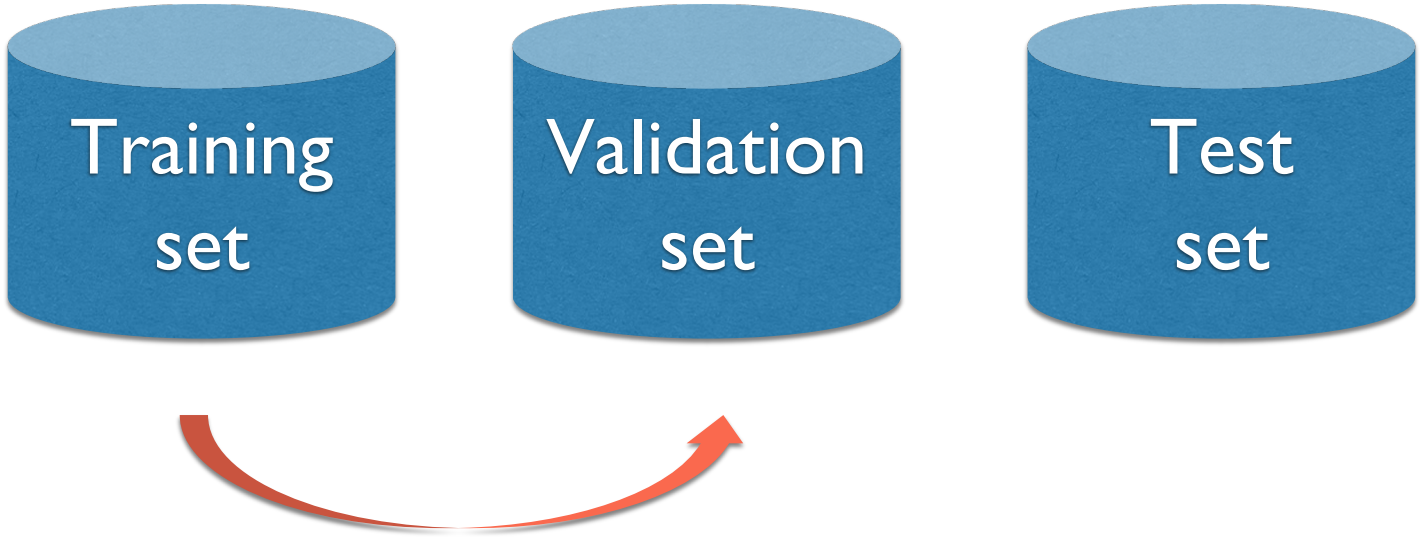
Preventing / Correcting Overfitting



- Model Selection: Find a model that gives good test error
  - Often this is just tuning a "hyper-parameter"
    (e.g., k in kNN, bandwidth in kernel regression, C in Soft-SVM)
  - Sometimes more involved: e.g., post-pruning in decision trees
- A few ways to do this:
  - Validation set (or cross-validation)
  - Theoretical approaches [Structural Risk Minimization]

Training, Validation, Testing

- Split data set into **three** data sets: **training**, **validation**, **test**

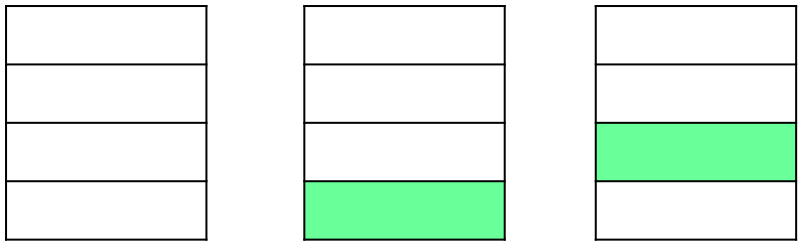


Try different hyper-parameters
(for instance: C=0.1, C=1, C=10 for SVM, or k=1,2,3,…n for kNN)

Report test error rate for the hyper-parameter value
that gave smallest validation error rate

Cross-validation

- Problem: What if the training set is an "unlucky" distribution
  - Error on the test set doesn't match real data
- Solution: Use *all* of the data as test data
  - But then we don't have any data to train on!
- Instead, cross-validation
  - Multiple training/test runs
  - Each uses a different subset as test data



Cross-validation

- **K-fold cross-validation**:
  - Randomly partition the training data into K equal-size subsets $S_1,\ldots,S_K$
  - For each i
    - Train on $S_1 \cup \cdots \cup S_{i-1} \cup S_{i+1} \cup \cdots \cup S_K$ (all the data except S_i)
    - Call this classifier $\hat{h}_i$
    - Evaluate error rate on S_i : $\text{error}_{S_i}(\hat{h}_i)$
  - Return the average: Cross-validation error = $\frac{1}{K}\sum_{i=1}^{K} \text{error}_{S_i}(\hat{h}_i)$

Cross-validation

- **K-fold cross-validation**:
  - Randomly partition the training data into K equal-size subsets $S_1,\ldots,S_K$
  - For each i
    - Train on $S_1 \cup \cdots \cup S_{i-1} \cup S_{i+1} \cup \cdots \cup S_K$ (all the data except S_i)
    - Call this classifier $\hat{h}_i$
    - Evaluate error rate on S_i : $\text{error}_{S_i}(\hat{h}_i)$
  - Return the average: Cross-validation error = $\frac{1}{K}\sum_{i=1}^{K} \text{error}_{S_i}(\hat{h}_i)$
- We can use this to tune hyper-parameters
  - For each setting of the hyper-parameters
  - Run K-fold cross validation
  - Choose the hyper-parameter values with lowest cross-validation error
  - Retrain on the **entire** training set, using the chosen hyper-parameter values

Cross-validation

- How do we pick K?

Cross-validation

- How do we pick K?

- Most popular in practice: K=10

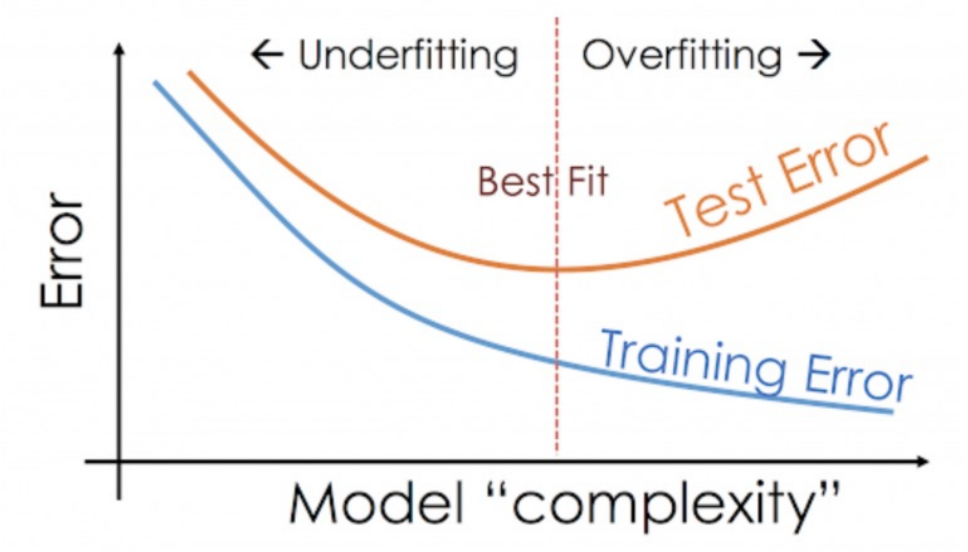
Cross-validation

- How do we pick K?
- Most popular in practice: K=10
- If K=n, it's called "leave one out" cross-validation:
  - Every training point (x_i,y_i) gets its own fold $S_i=\{(x_i,y_i)\}$
  - But this is computationally expensive
  - Also can sometimes have higher variance

Cross-validation

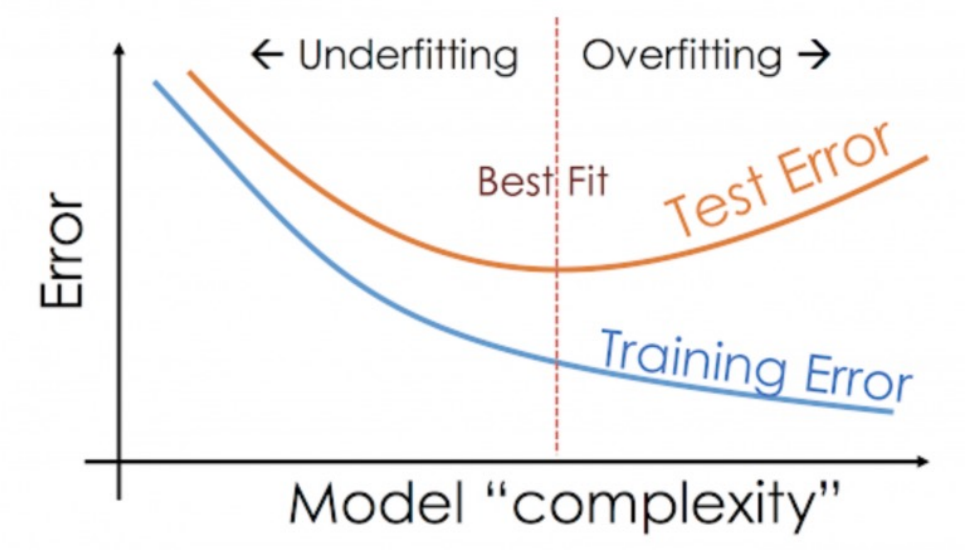
- How do we pick K?
- Most popular in practice: K=10
- If K=n, it's called "leave one out" cross-validation:
  - Every training point (x_i,y_i) gets its own fold $S_i=\{(x_i,y_i)\}$
  - But this is computationally expensive
  - Also can sometimes have higher variance
- To estimate the error rate in the end,
  we would still need a separate held-out test set

Preventing / Correcting Overfitting



- Other approaches:
  - Regularization (e.g., minimizing ||w|| in SVM optimization)
  - Dimensionality reduction / feature selection

Preventing / Correcting Overfitting



- Other approaches:
  - Regularization (e.g., minimizing ||w|| in SVM optimization)
  - Dimensionality reduction / feature selection
- Note: sometimes the appropriate units for x-axis aren't easy to identify.
  - Sometimes large neural networks overfit less than smaller
  - Possibly because the optimization finds good solutions more easily