Bias/Variance and Cross-Validation

Miles Erickson

Original Slides: Ryan Henning

galvanize

- Overfitting and Underfitting
- The Bias/Variance Tradeoff
- Cross-Validation
- K-fold Cross-Validation
- Subset Selection of Predictors

Quick Review: Regression vs. Classification
(in machine learning)

What is regression?

Use features to predict real valued targets.   E.g. predict future sales/revenue

What is classification?

Use features to predict categorical targets.   E.g. predict yes/no, male/female, 0-9

<u>Stepping back:</u> One Goal of Data Science: Make Future Predictions

galvanize

One goal is to make accurate ***<u>predictions</u>*** on future (unseen) data.

1. Define a business goal.
    e.g. make Tesla cars the most dependable vehicles on the market

2. Collect training data.
    e.g. Tesla cars' event logs + historical record of parts replaced

3. Train a model.
    e.g. **features:** event statistics, **target:** time until failure

4. Deploy the model.
    e.g. monitor cars' events in real time, send mechanics to replace parts that will soon fail

Questions!

Underfitting and Overfitting

Underfitting: The model doesn’t fully capture the relationship between predictors and the target. The model has *not* learned the data’s signal.

→ What should we do if our model underfits the data? (assume using kNN)

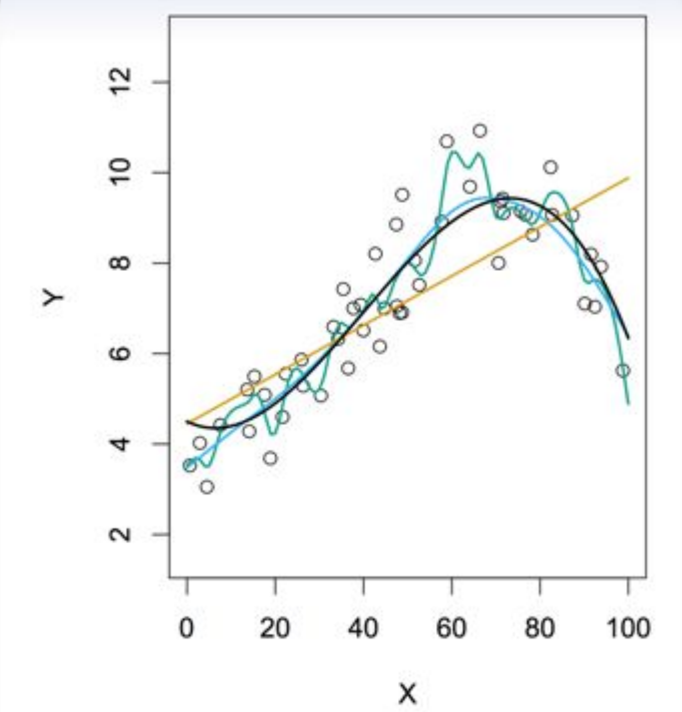
Overfitting: The model has tried to capture the sampling error. The model has learned the data’s signal *and* the noise.

→ What should we do if our model overfits the data?

The Bias/Variance Tradeoff

Let's get an intuitive feel for the bias and the variance of a model… we'll see more math on the next slide.

Note: **Bias** and **Variance** are terms you will use A TON as a data scientist! Exciting times!



We assume the true predictor/target relationship is given by an unknown function plus some sampling error:

$$Y = f(X) + \epsilon$$

We estimate the true (unknown) function by fitting a model over the training set.

$$\hat{Y} = \hat{f}(X)$$

Let's evaluate this model using a test observation **(x_0, y_0)** drawn from the population. What is the model's expected squared prediction error on this test observation?

$$E[(y_o - \hat{f}(x_0))^2] = \ldots$$

galvanize

Our model's expected squared prediction error will depend on (1) the variability of $\mathbf{y_0}$ and (2) the variability of the training set used to train our model. We can break this into three pieces:

$$E[(y_o - \hat{f}(x_0))^2] = \ldots = \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\epsilon)$$

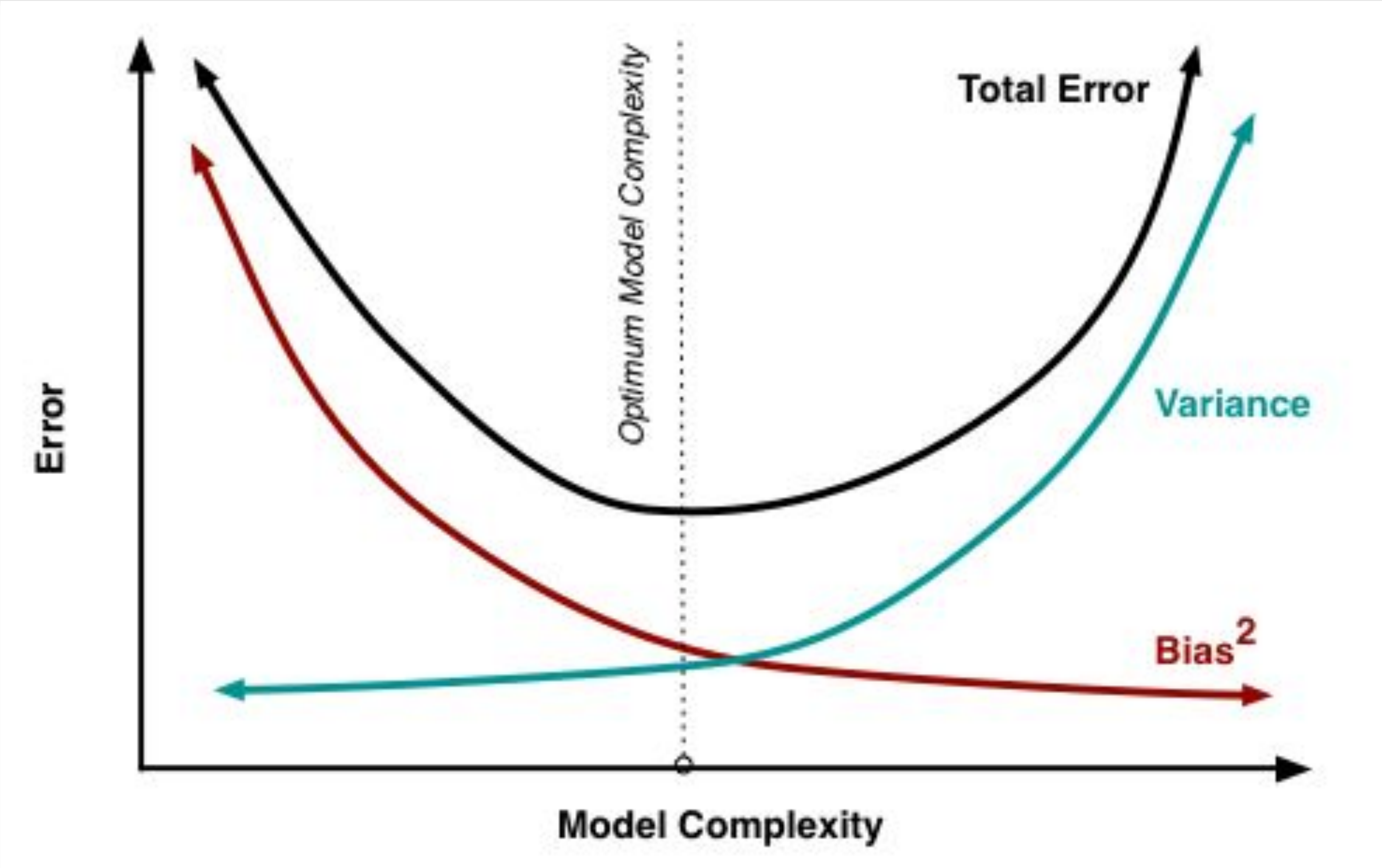
The variance of our model's prediction of $\mathbf{x_0}$ over all possible training sets

The difference between the true target and our model's average prediction over all possible training sets

The variance of the irreducible error.

$$\mathrm{Bias}(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$$

galvanize



How is the **bias/variance tradeoff** related to **underfitting** and **overfitting**?

How can we find the best tradeoff point? I.e. The optimum model complexity

Cross-Validation

Main idea: **Don't use all your data for training.**

Instead: **Split your data into a "training set" and a "validation set".**

randomize!
(why?)

training set

where to split?

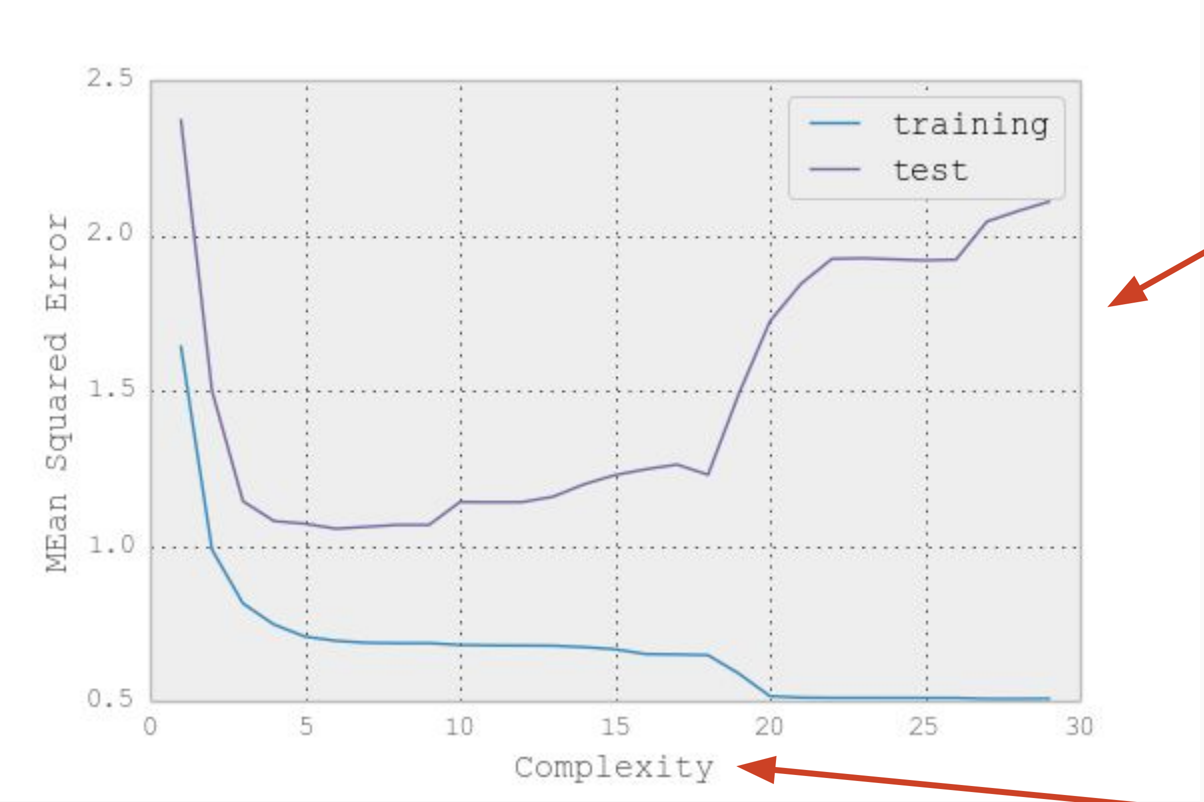
validation set

Cross-Validation

1. Split your data into training/validation sets.
   70/30, 80/20, 90/10 splits are commonly used
2. Use the training set to train several models of varying complexity.
   How do we adjust model complexity?
3. Evaluate each model using the validation set.
   Calculate MSE, log loss, or whatever error metric is best for the problem domain.
4. Keep the model that performs best over the **validation** set.

Let’s predict MPG from horsepower

Cross-Validation Example



You will see this shape all the time!

You will wrestle with the bias/variance tradeoff constantly...

E.g. linear regression w/ varying degree of polynomial

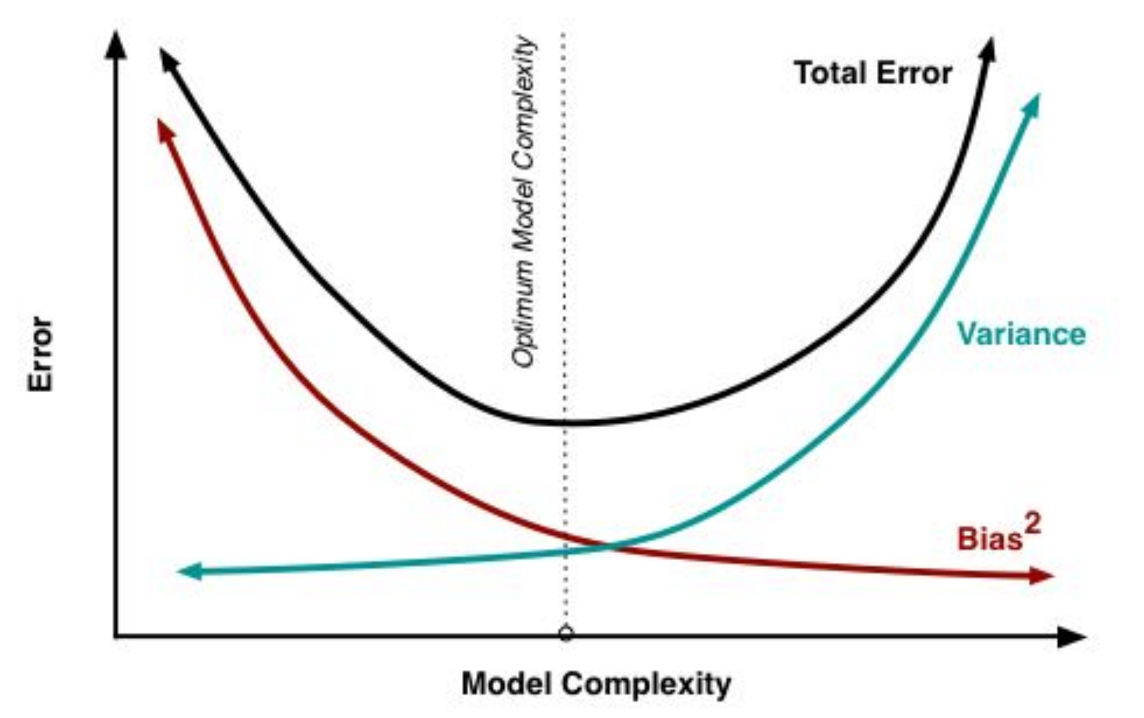
Recall our goal: Making accurate future predictions

galvanize

Fitting the training set perfectly is *easy*.
How?

Fitting future (unseen) data is *not easy*.

Cross validation helps us choose a model that performs well on unseen data.

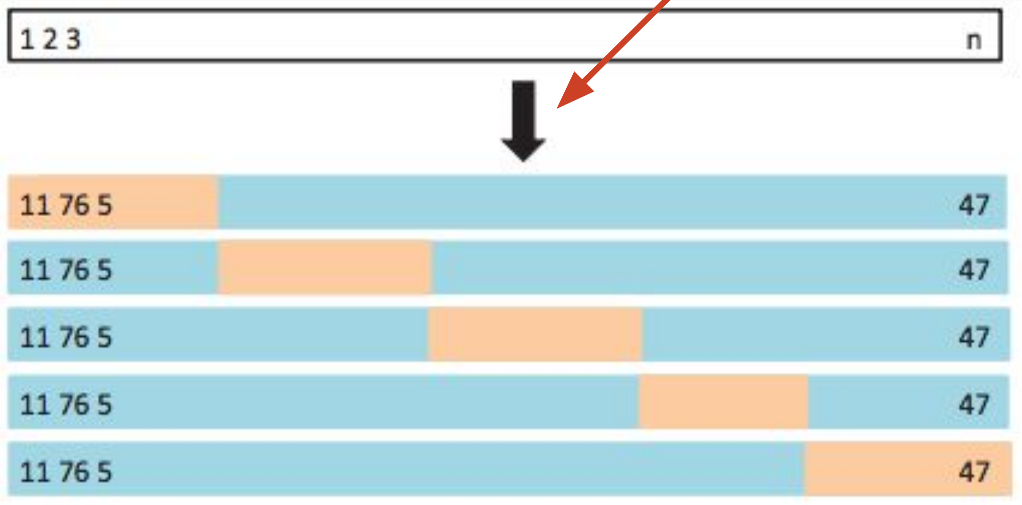


k-Fold Cross-Validation

Commonly, k=5 or k=10

randomize!

1. Split the dataset into k “folds”.
2. Train using (k-1) folds. Validate using the one “leave out” fold. Record a validation metric such as RSS or accuracy.
3. Train *k* models, leaving out a different fold for each one.
4. Average the validation results.

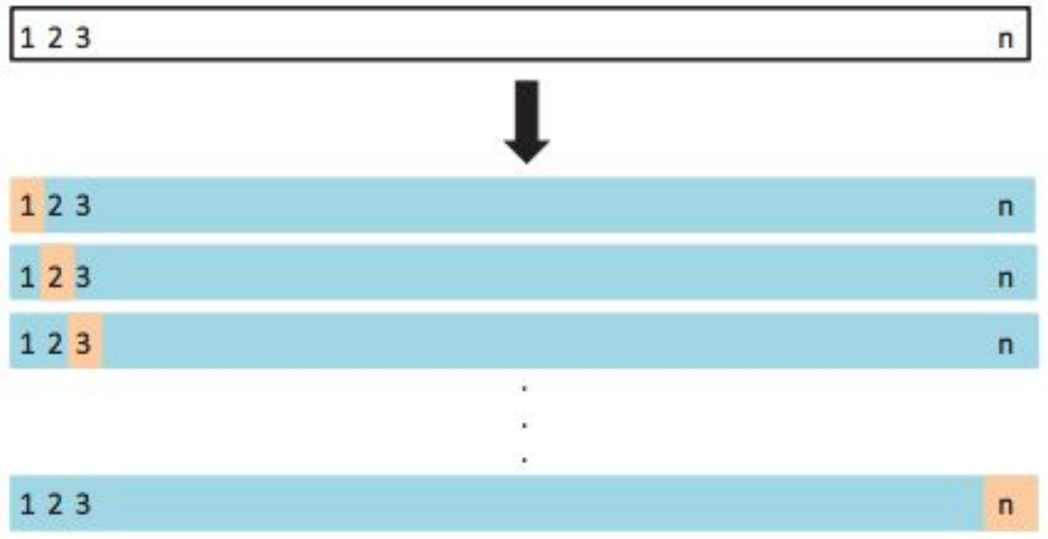


Leave-one-out Cross-Validation

Assume we have *n* training examples.

A special case of k-fold CV is when k=n. This is called *leave-one-out cross-validation*.

Useful (only) if you have a tiny dataset where you can't afford a large validation set.

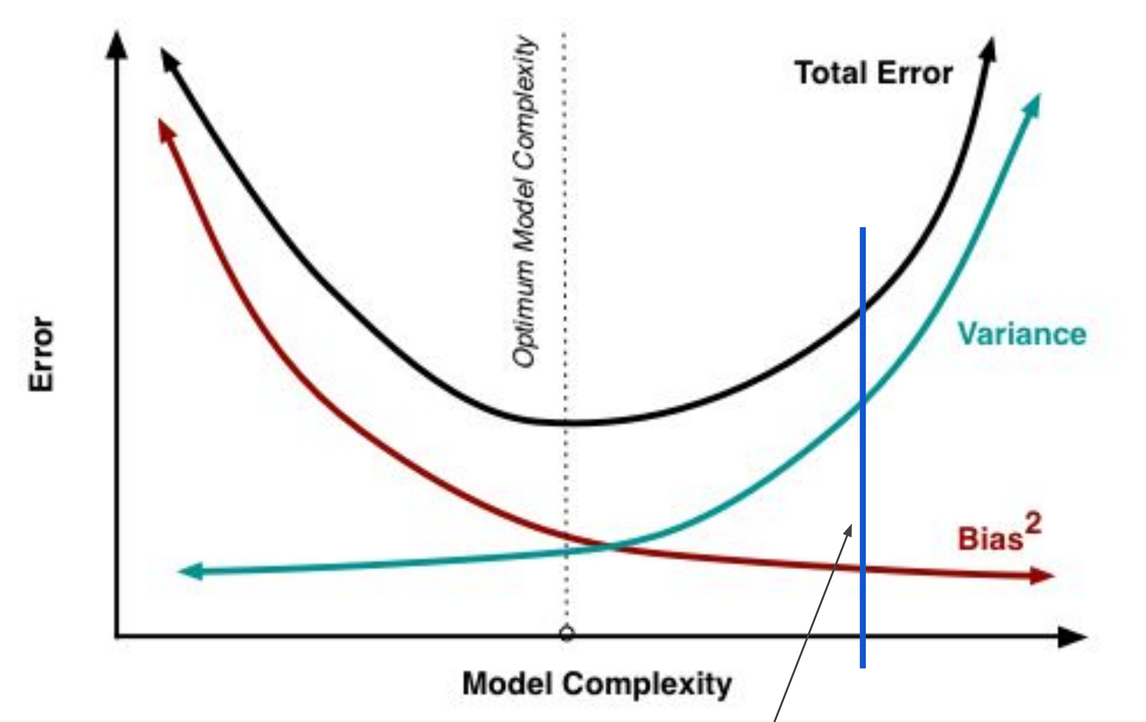


Overfitting in high dimensions is easy, even with simple models.

galvanize

If our data has high dimensionality (many predictors), then it becomes *easy* to overfit to the training data.

This is one result of the **Curse of Dimensionality**.



If p is large, you start over here

“HELP, my model is overfitting!”

You have a few options.

1. Get more data… (not usually possible/practical)
2. $\boxed{\textbf{Subset Selection:}}$ keep only a subset of your predictors (i.e, dimensions)
3. **Regularization:** restrict your model’s parameter space
4. **Dimensionality Reduction:** project the data into a lower dimensional space

Subset Selection

Best subset: Try every model. Every possible combination of *p* predictors

- Computationally intensive. 2^p possible subsets of *p* predictors
- High chance of finding a "good" model by random chance.

... A sort-of monkeys-Shakespeare situation ...

Stepwise: Iteratively pick predictors to be in/out of the final model.

- Forward, backward, forward-backward strategies

scikit-learn

Classes:

- sklearn.linear_model.**KNeighborsClassifier**(n_neighbors=k)
- sklearn.linear_model.**KNeighborsRegressor**(n_neighbors=k)
- sklearn.linear_model.**LinearRegression**(...)

All have these methods:

- fit(X, y)
- predict(X)
- predict_proba(X) -- *classifiers only*