Bias/Variance and Cross-Validation

Matt Drury

Based on Prior Work of RH and JFO

- Explain the bias/variance tradeoff
- Define overfitting and underfitting
- Relate flexibility and complexity of a model to bias / variance, to overfitting / underfitting, and to training / testing error
- Define and implement hold-out and cross-validation on a dataset

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

Stepping back: One Goal of Data Science: Make Future Predictions

One goal is to make accurate ***predictions*** 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!

Review: Linear Regression

We assume the world is built on linear relationships. Under that assumption, we can model the relationship between *features* and a *target* like this:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$$

target
(aka, outcome)

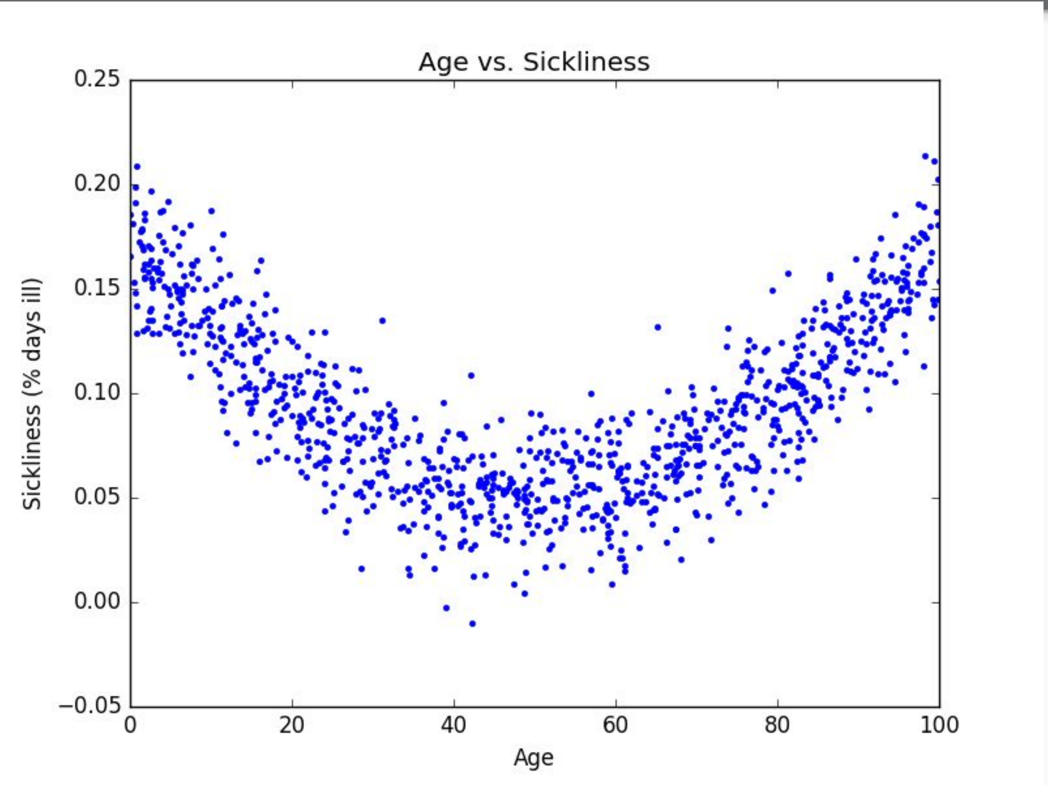
model parameters
(aka, coefficients)

features
(aka, predictors)

sampling error

Review: Linear Regression

We can make linear regression non-linear by inserting extra “interaction” features or higher-order features.



Example:

$$Y = \beta_0 + \beta_1 * \text{age}$$

$$Y = \beta_0 + \beta_1 * \text{age} + \beta_2 * \text{age}^2$$

Is R^2 all that matters?

We *could* just keep inserting interaction features until $R^2 = 1$.

Boom. I solved data science. Here's my idea:

```
def train_super_awesome_perfect_model (X, y):
    while True:
        model = LinearRegression()
        model.fit(X, y)
        if calculate_r2(model, X, y) >= 0.999:
            return model
        else:
            X = insert_new_feature(X)
```

Why is this a bad idea?

Oh the woes of overfitting...

Play with the app at

http://madrury.github.io/smoothers/

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 lin. reg.)

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

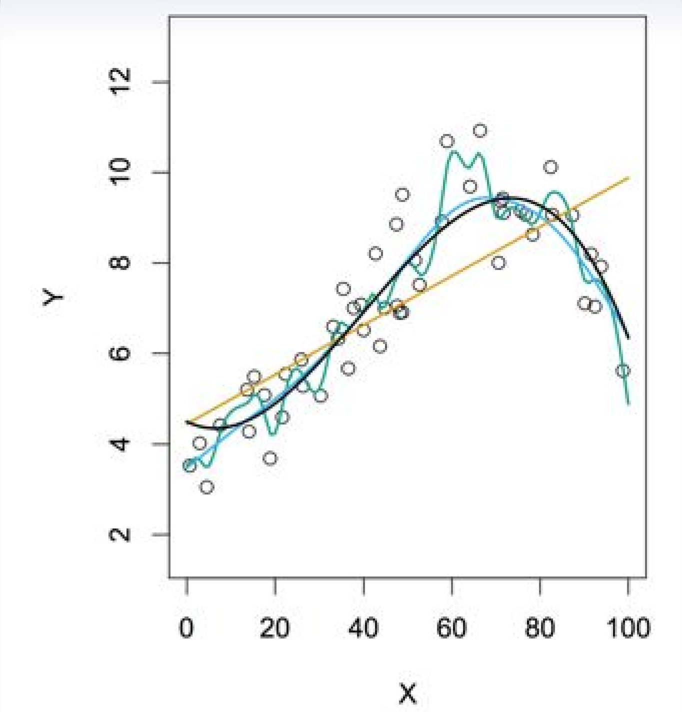
→ What should we do if our model overfits the data? (harder… any guesses?)

The Bias/Variance Tradeoff

Boardwork… build intuition...

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 error we cannot hope to capture:

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

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?

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

In this expectation, we are considering **x_0 as a fixed quantity**, and **y_0 as a random quantity.** So the averaging above is over the **randomness in y_0**

The Bias/Variance Tradeoff

The **bias-varaince** tradeoff breaks this quantity up into **sources of varaition:**

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

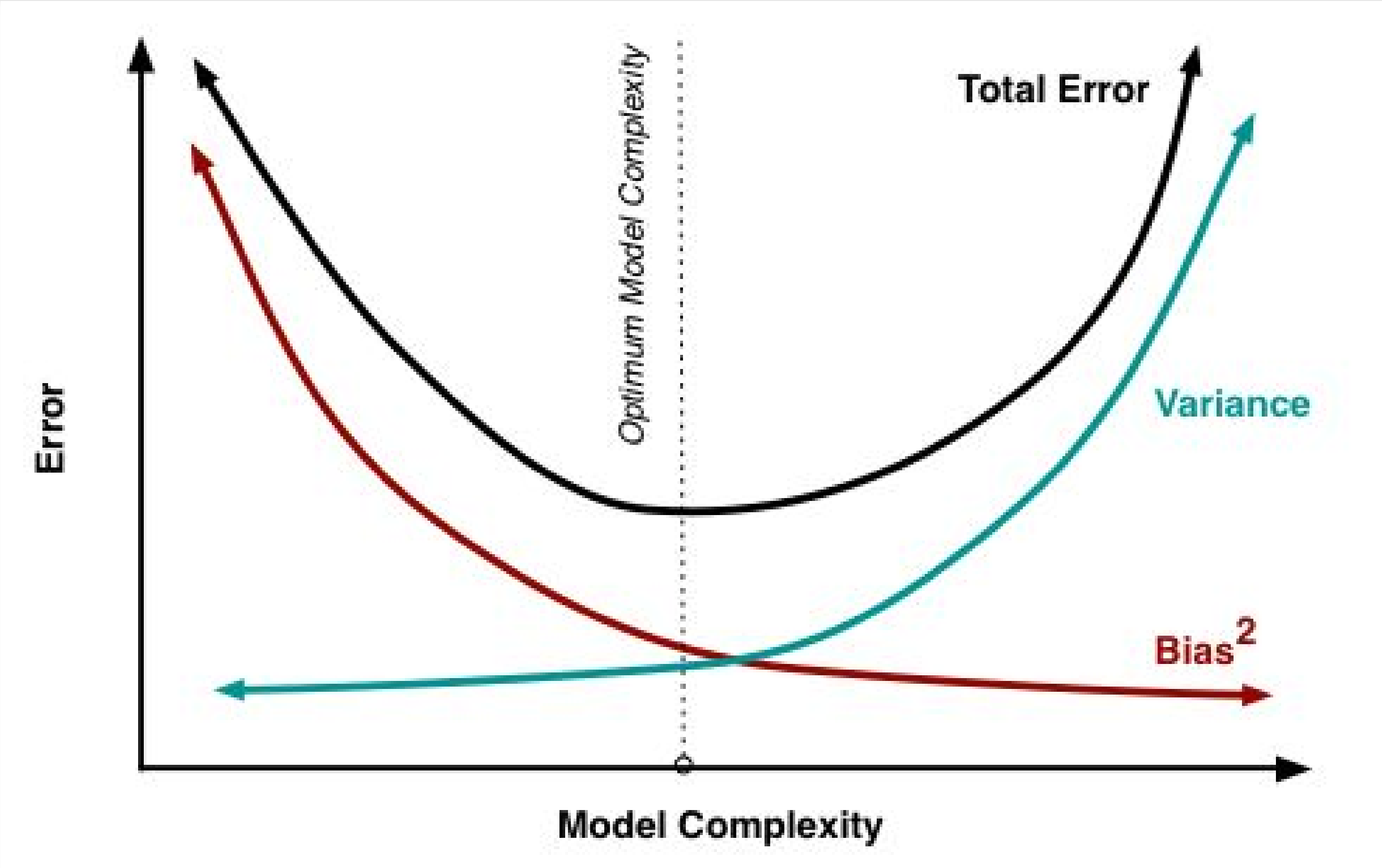
The variance of our model's prediction of $\mathbf{x_0}$ over all possible values of $\mathbf{y_0}$

The difference between the truth and our model's average prediction over all possible values of $\mathbf{y_0}$

The variance of the irreducible error.

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

The Bias/Variance Tradeoff



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

QUESTION TIME !

1. What are the ways we can increase complexity of a model?

2. Why does variance increase and bias decrease
   as model complexity increases?

3. How do you determine the optimal complexity to use?

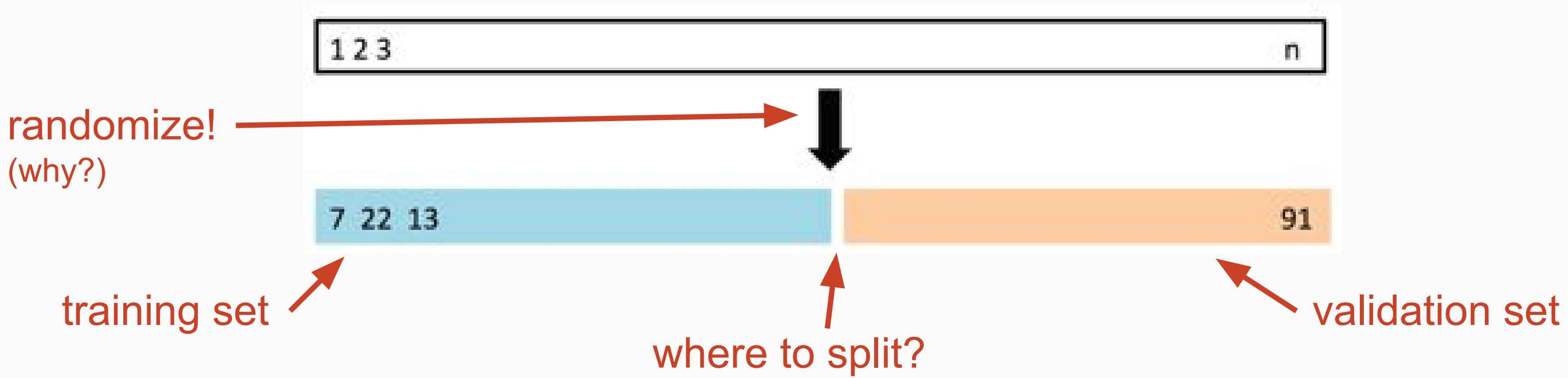
An Explicit Example

Switch to other slide deck...

Hold Out Validation

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

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



Hold-Out Validation

1. Split your data into training/validation sets.

   70/30 or 90/10 splits are commonly used

2. Use the training set to train several models of varying complexity.

   e.g. linear regression (w/ and w/out interaction features), neural nets, decision trees, etc.
   (we'll talk about hyperparameter tuning, grid search, and feature engineering later)

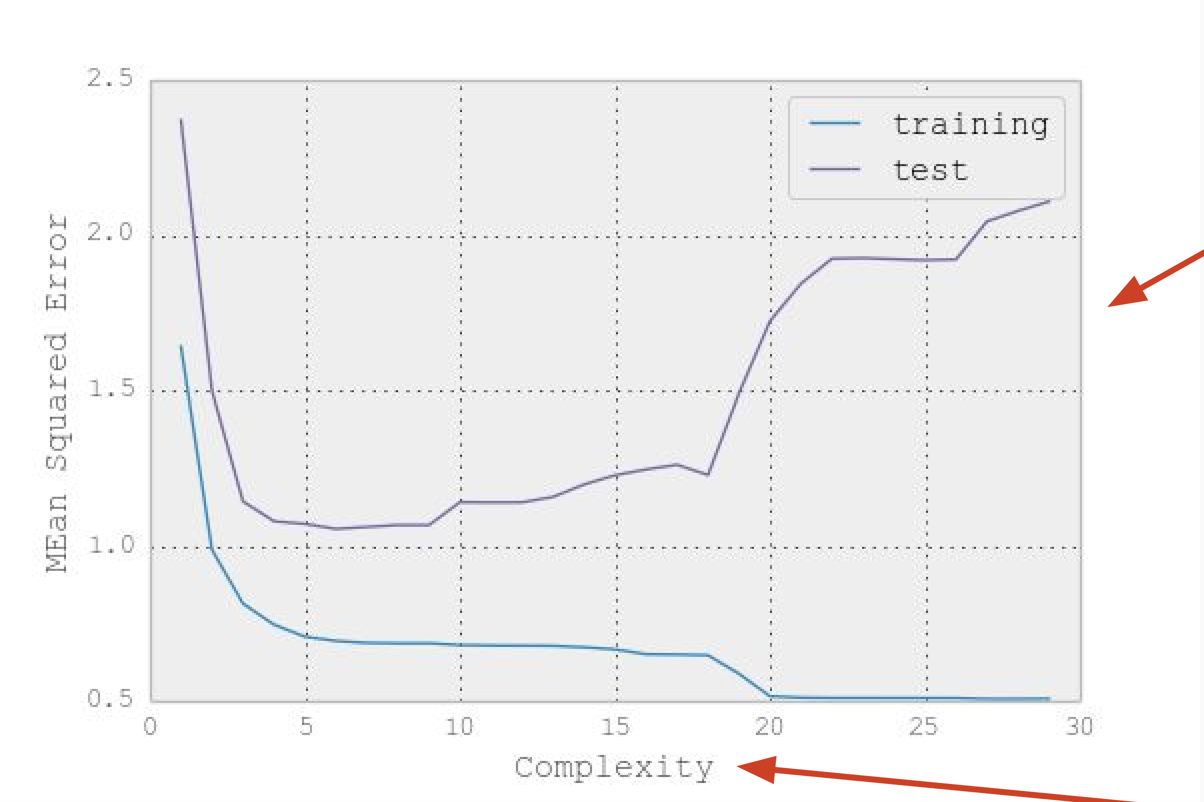
3. Evaluate each model using the validation set.

   calculate R^2, MSE, likelihood, or whatever you think is best

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...

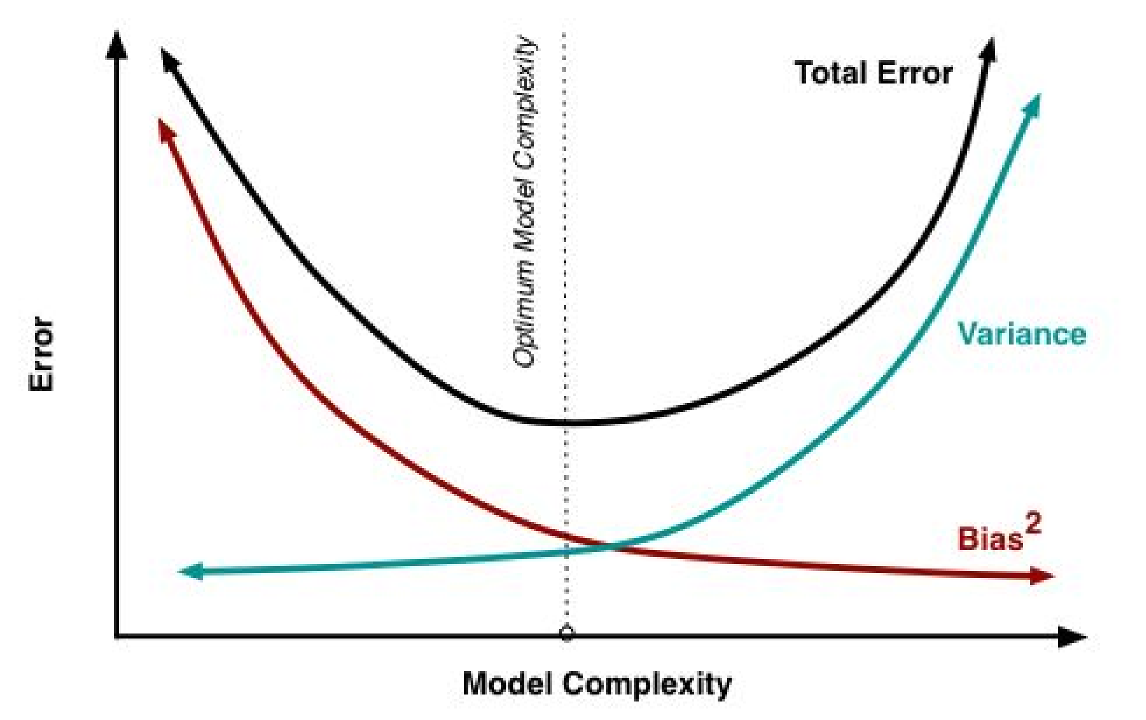
E.g. linear regression w/ varying degree of polynomial

Recall our goal: Making accurate future predictions

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.

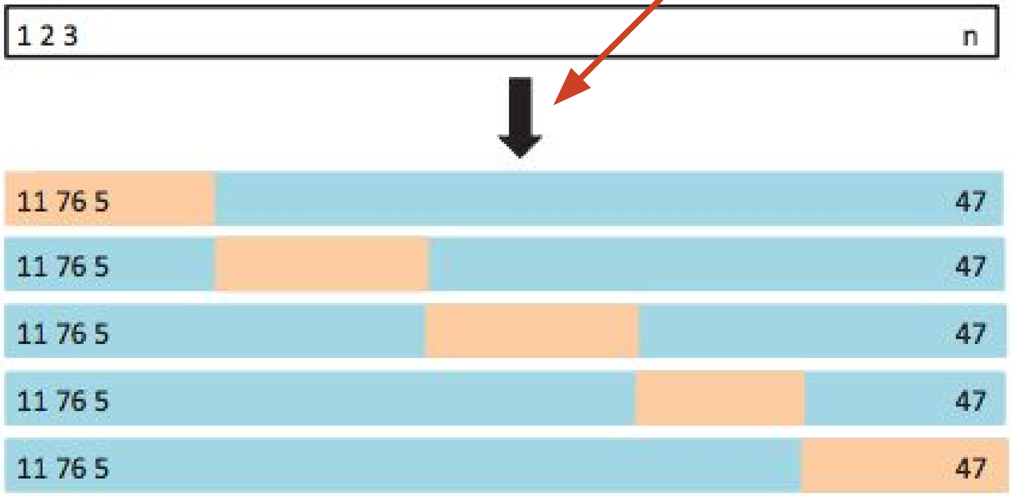


k-Fold Cross-Validation

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.

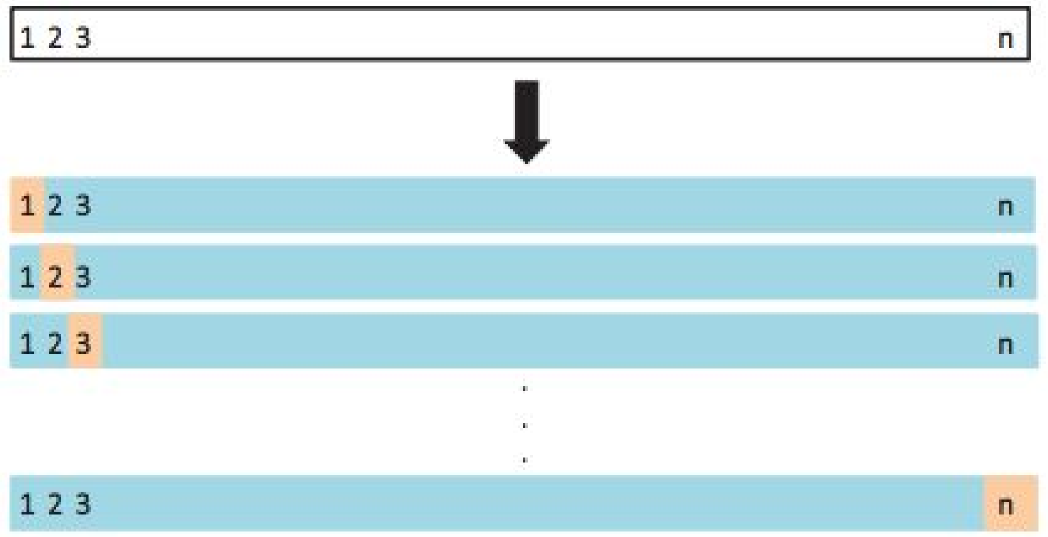


Leave-one-out Cross-Validation

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.

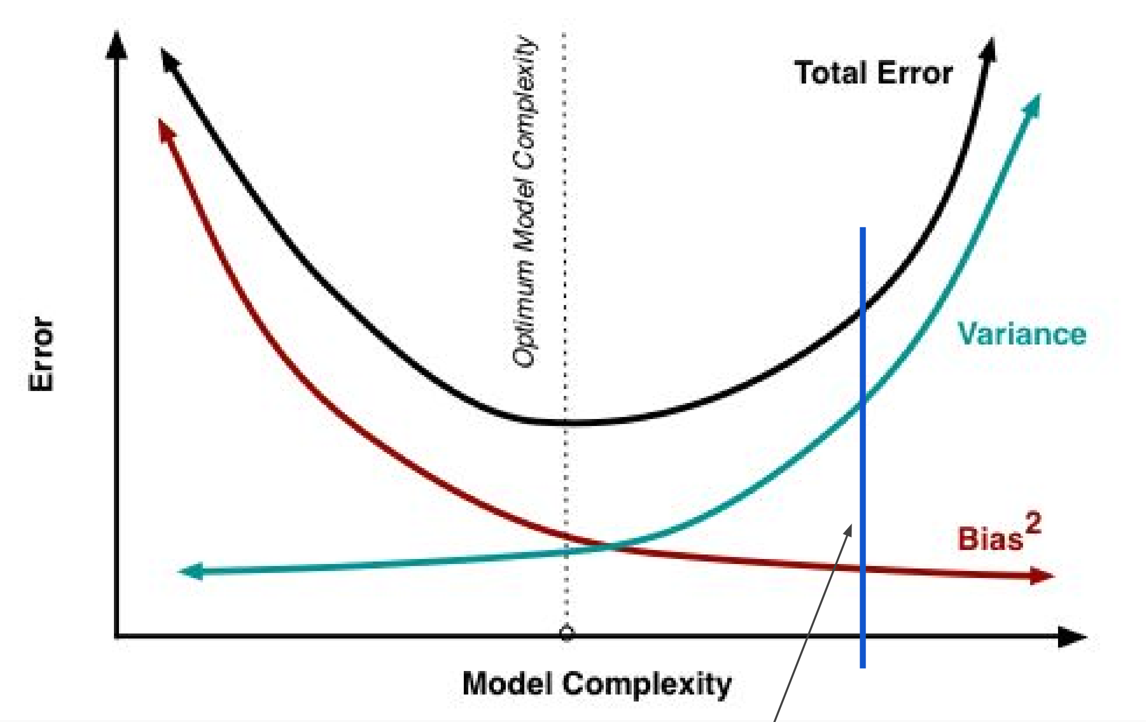


Overfitting in high dimensions is easy, even with simple models.

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

This is one result of the so-called **Curse of Dimensionality** (look it up).

Even linear regression might be too complex of a model for high dimensional data (and the smaller the dataset, the worse this problem is).



Linear regression in high dimensions

QUESTION TIME !

1. Why do we often prefer to use K-Fold Cross Validation instead of a simple train test split?

2. How many models are created in a 5-fold cross validation to compare two models (model A and model B)?

3. What do we do with all the models we created and how do we determine what single model to keep?

“HELP, my model is overfitting!”

You have a couple of options...

1. Get more data… (not usually possible/practical)
2. **Regularization:** restrict your model’s parameter space

TO BE CONTINUED...