CSE/STAT 416

Assessing Performance

Amal Nanavati
University of Washington
June 27, 2022

Adapted from Hunter Schafer's slides

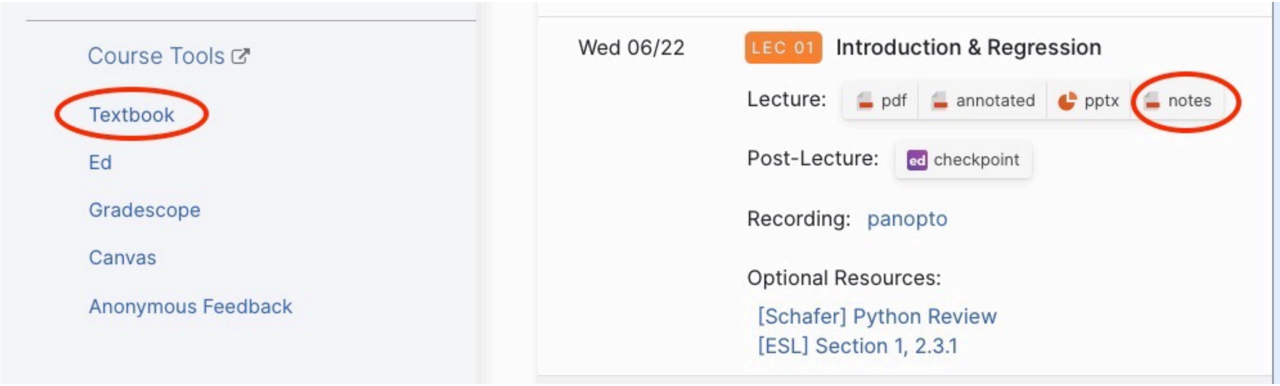
Today's Agenda

Administrivia & Recap (20 mins)

Main Lecture: Assessing Performance (1 hr 30 mins)

Administrivia

We have lecture notes!



Notes on OH

Check EdStem for announcements or clarifications on logistics

Great job on Learning Reflection 1!

Upcoming Timeline:

- HW 1 released Wed 6/29, Due <u>**Tues 7/5, 11:59PM**</u>
- Checkpoint 2 <u>**Due Wed 6/29 1:50PM**</u>
- Learning Reflection 2 <u>**Due Fri 7/1 11:59PM**</u>

Lecture 1 Recap

Linear Regression Model

Assume we have a simple model with ***one feature***, where we establish a linear relationship between ***the area of a house*** i and ***its price***:

$$y_i = f(x_i) + \varepsilon_i$$

$$y_i = w_0 + w_1 x_i + \epsilon_i$$

$$b + mx$$

w_0, w_1 are the **parameters** of our model that need to be learned

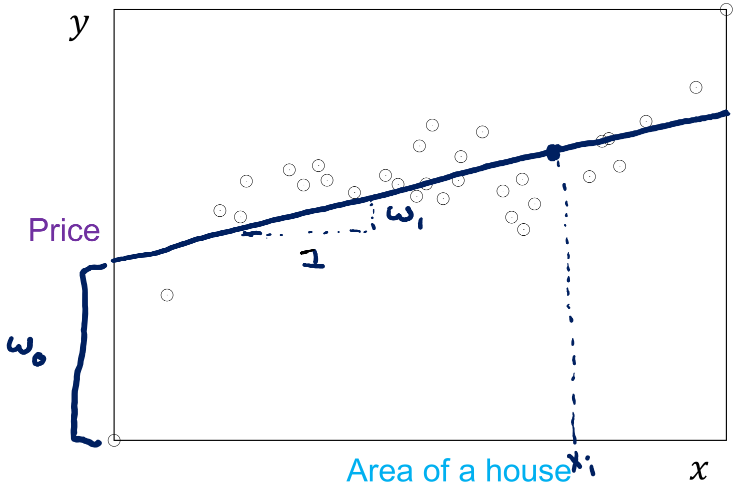
- w_0 is the intercept / **bias**, representing the starting price of a house
- w_1 is the slope / **weight** associated with **feature** ”area of a house”

Learn estimates of these parameters $\hat{w}_1$, $\hat{w}_0$ and use them to predict new value for any input x!

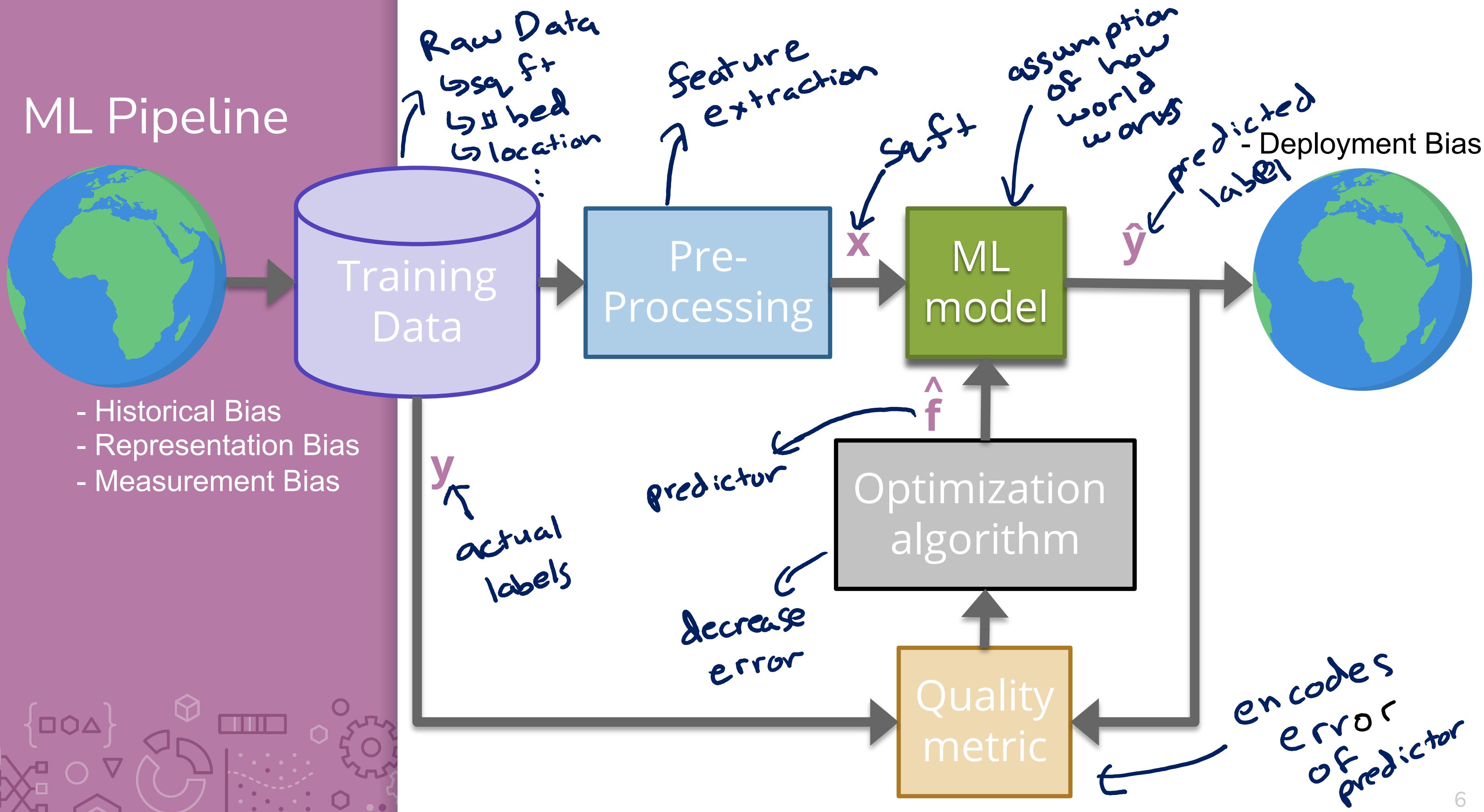
$$\hat{f}(x)$$

$$\hat{y} = \hat{w}_1 x + \hat{w}_0$$

Why don’t we add ϵ?



ML Pipeline



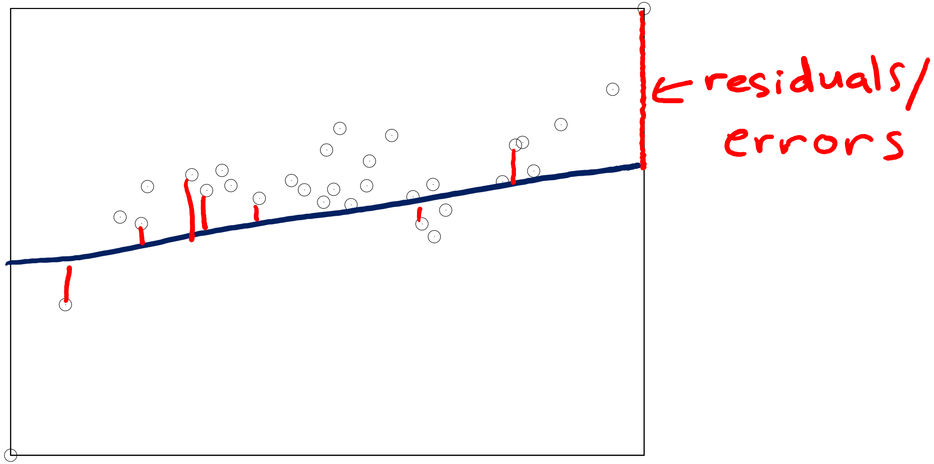
- Historical Bias
- Representation Bias
- Measurement Bias

- Deployment Bias

Mean Squared Error (MSE)

How to define error? **Mean squared error (MSE)**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

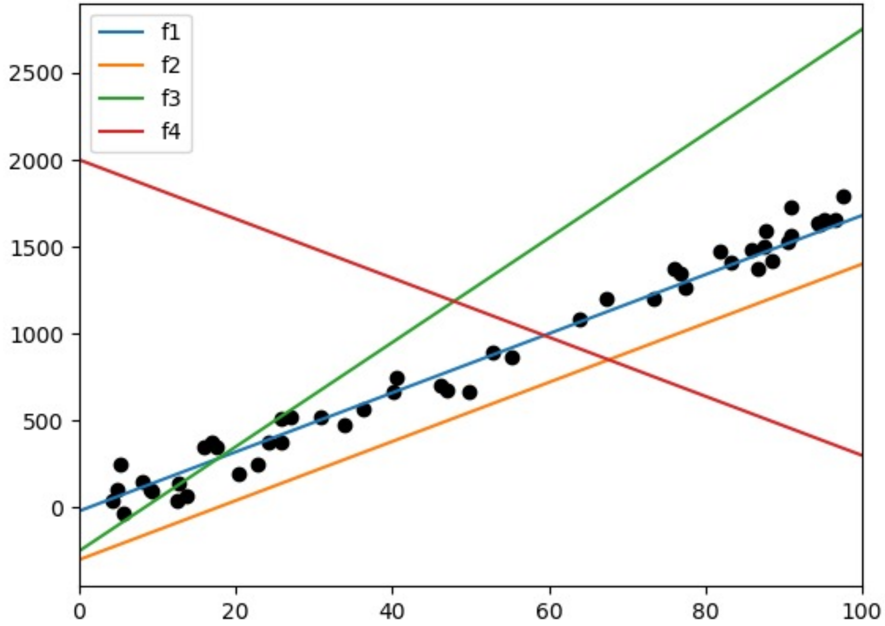


Poll Everywhere

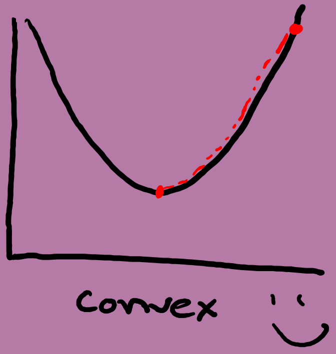
Think

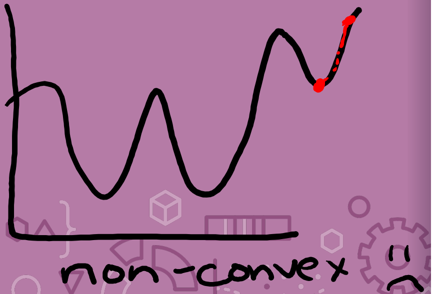
1 min

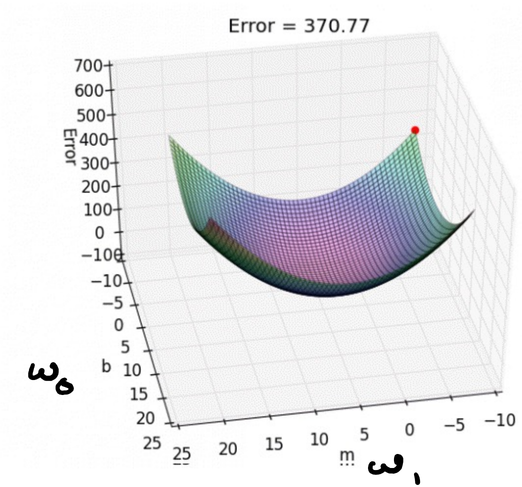
Sort the following lines by their MSE on the data, from smallest to largest. (estimate, don't actually compute)

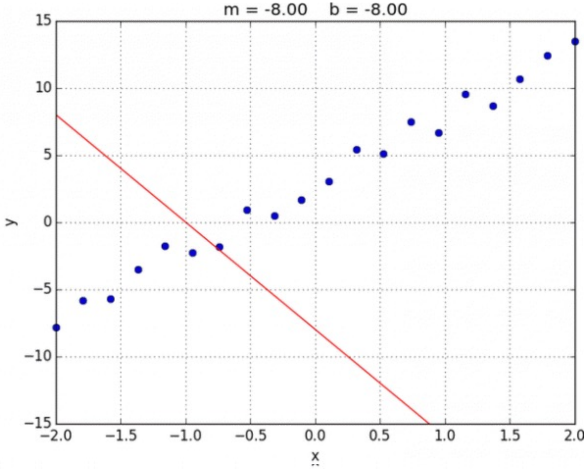


Gradient Descent









Instead of computing all possible points to find the minimum, just start at one point and "roll" down the hill.
Use the gradient (slope) to determine which direction is down.

Start at some (random) weights w
While we haven't converged:

$$w -= \alpha \nabla L(w)$$

gradient = direction of maximum ascent

- α: learning rate
- $\nabla L(w)$: the gradients of loss function L on a set of weights w

Adding Other Features

Generally, we are given a data table of values we might look at that includes more than one feature per house.

- Each row is a data point.
- Each column represents a feature
- One of the columns contains the actual output values

sq. ft.	# bathrooms	owner's age	…	price
1400	3	47	…	70,800
700	3	19	…	65,000
…	…	…	…	…
1250	2	36	…	100,000

- Sometimes we want to extract new features from existing features (e.g., #bath/#bed)

Features

Features are the values we select or compute from the data inputs to put into our model. **Feature extraction** is the process of reduce the number of features in a dataset by creating new features from the existing ones (and then discarding the original features).

Model

$$y = w_0 h_0(x) + w_1 h_1(x) + \ldots + w_D h_D(x)$$

$$= \sum_{j=0}^{D} w_j h_j(x)$$

Feature	Value	Parameter
0	$h_0(x)$ often 1 (constant)	w_0
1	$h_1(x)$	w_1
2	$h_2(x)$	w_2
…	…	…
d	$h_d(x)$	w_d

Linear Regression Recap

Dataset

$\{(X^{(i)}, y^{(i)})\}_{i=1}^{n}$ where $X^{(i)} \in \mathbb{R}^d$, $y \in \mathbb{R}$

Feature Extraction

$h(x)\colon \mathbb{R}^d \to \mathbb{R}^D$

$h(x) = (h_0(x), h_1(x), \dots, h_D(x))$

Regression Model

$y = f(x) + \epsilon$

$= \sum_{j=0}^{D} w_j h_j(x) + \epsilon$ ← sometimes omit ϵ

$= w^T h(x) + \epsilon$

Quality Metric / Loss function

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

Predictor

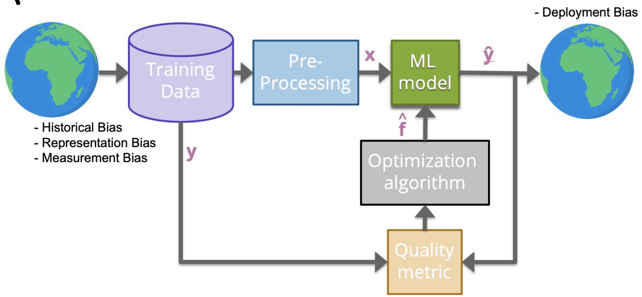
$$\hat{w} = \underset{w}{\operatorname{argmin}}\, MSE(w)$$

Optimization Algorithm

Optimized using Gradient Descent

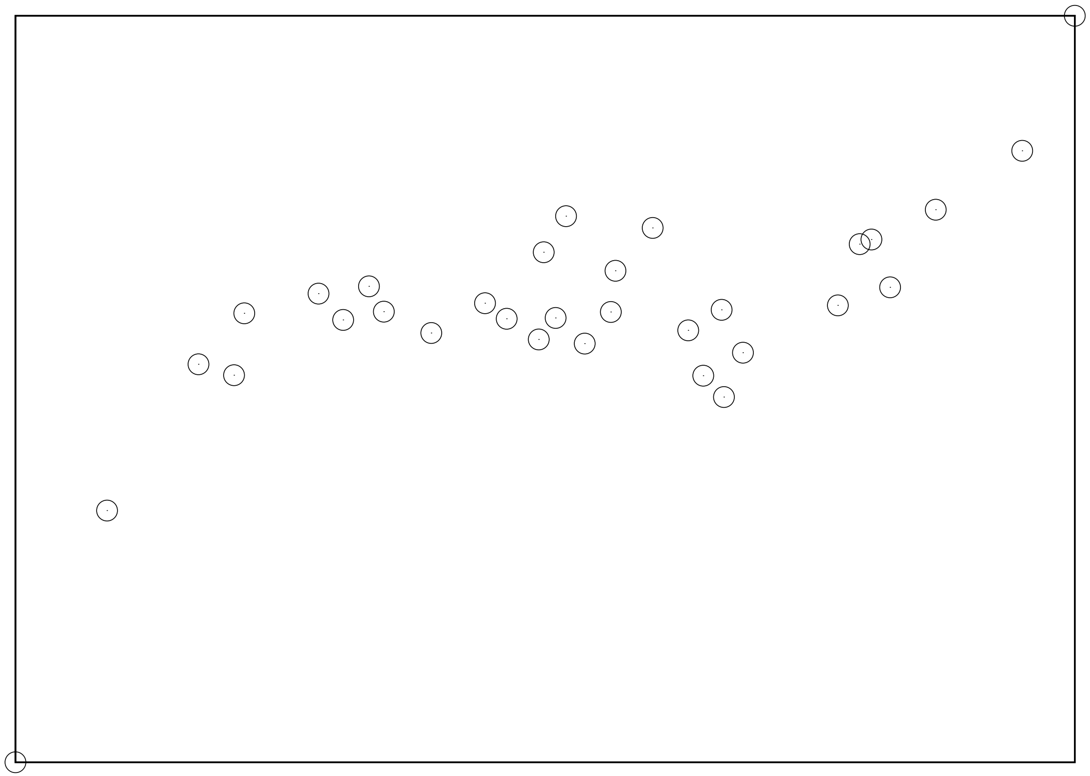
Prediction

$$\hat{y} = \hat{w}^T h(x)$$



Assessing Performance

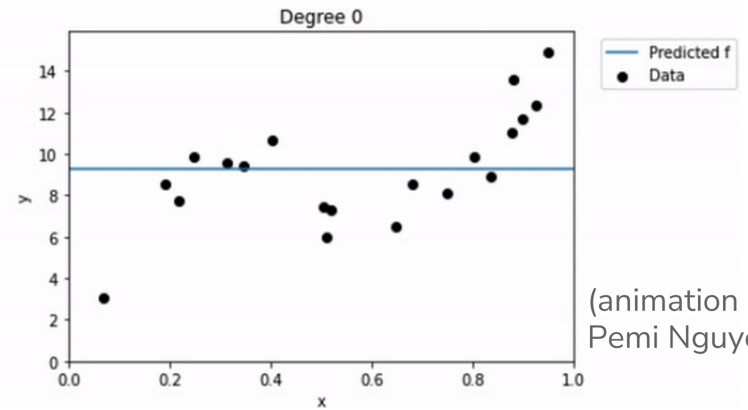
Polynomial Regression



How do we decide what the right choice of p is?

Polynomial Regression

Consider using different degree polynomials on the same training set.



(animation by Pemi Nguyen)

From estimating with your eyes, which one seems to have the lowest MSE on this dataset?

It seems like minimizing the MSE on the training set is not the whole story here …

Performance

Why do we train ML models?

We generally want them to do well on **unseen** data.

If we choose the model that minimizes MSE on the data it learned from, we are just choosing the model that can **memorize**, not the one that **generalizes** well.

Analogy: Just because you can get 100% on a practice exam you've studied for hours, it doesn't mean you will also get 100% on the real test that you haven't seen before.

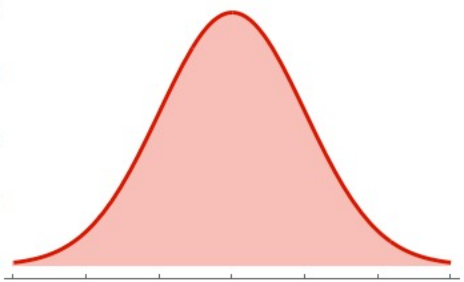
Key Idea: Assessing yourself based on something you *learned from* generally overestimates how well you will do in the future!

Future Performance

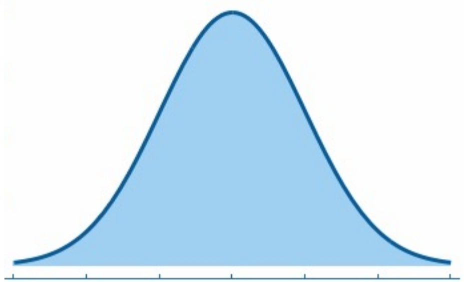
What we care about is how well the model will do on unseen data.

How do we measure this? **True error**

To do this, we need to understand uncertainty in the world



Sq. Ft.



Price | Sq. Ft.

True Error

Model Assessment

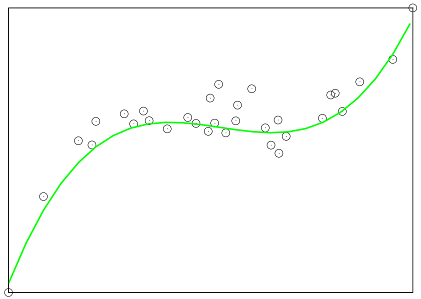
How can we figure out how well a model will do on future data if we don't have any future data?

Estimate it! We can hide data from the model to test it later as an estimate how it will do on future data

We will randomly split our dataset into a **train set** and a **test set**

The train set is to train the model

The test set is to estimate the performance in the future



Test Error

What we really care about is the **true error,** or how well a model perform on unseen data in the wild, but we can't know that without having an infinite amount of data!

We will use the **test set** to estimate the true error.

Note: The train and test set need to be **randomly split** in order for the test set to be truly reflective of data in the real world.

Call the error on the test set the **test error** for a model $\hat{f}$:

$$MSE_{test} = \frac{1}{n} \sum_{i \in Test} \left(y^{(i)} - \hat{f}\left(x^{(i)}\right) \right)^2$$

If the test set is large enough, this can approximate the true error.

Train/Test Split

If we use the test set to estimate future, how big should it be?

This comes at a cost of reducing the size of the training set though (in the absence of being able to just get more data)

In practice people generally do train:test as either

80:20

90:10

Important: Never train your model on data in the test set!

Poll Everywhere

Think

1 minute

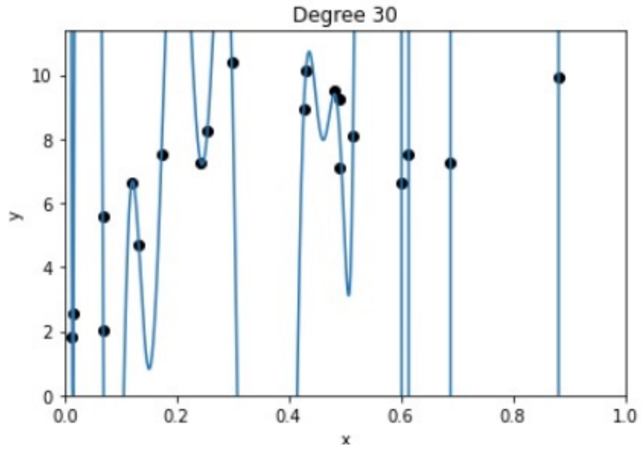
Which of the models do you expect to have the:

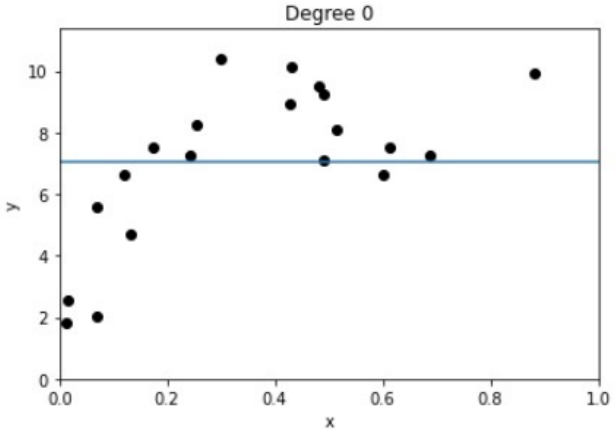
Highest Train Error

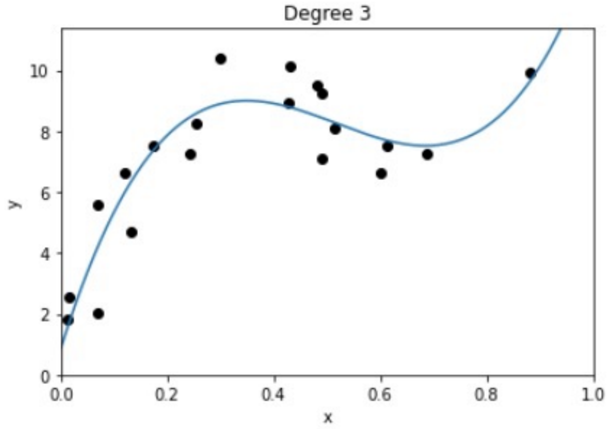
Highest Test Error

Lowest Train Error

Lowest Test Error







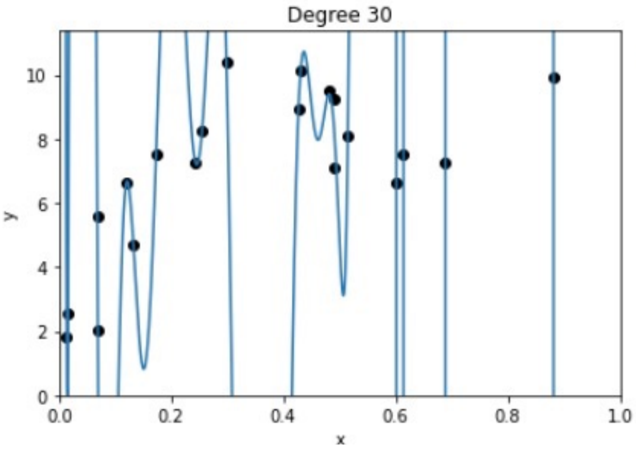
Poll Everywhere

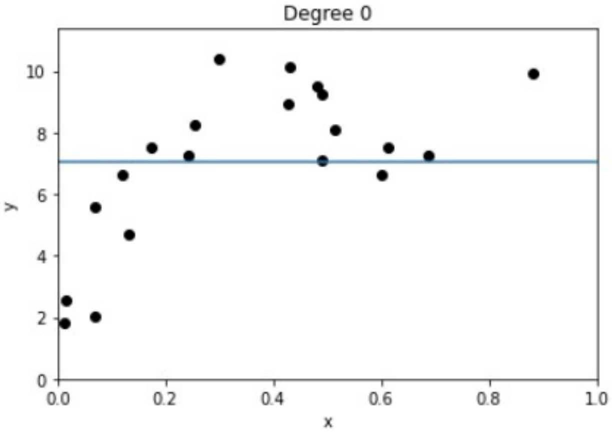
Group

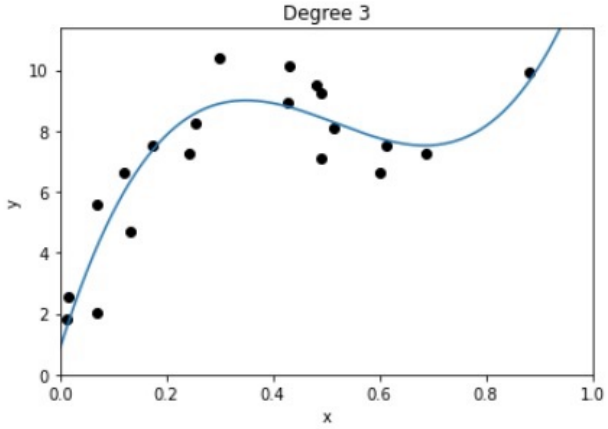
2 minute

Which of the models do yoı

- Highest Train Error
- Highest Test Error
- Lowest Train Error
- Lowest Test Error







Brain Break

Model Complexity

Model Complexity

There is not a well-defined way to measure the complexity of a model. It depends on the nature of the models.

We usually associate it with the number of parameters. A model with more parameters is usually more complex.

Example with polynomial regression:

- Model 1: (2 parameters)
  - $y = w_0 + w_1 x$
- Model 2: (4 parameters)
  - $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

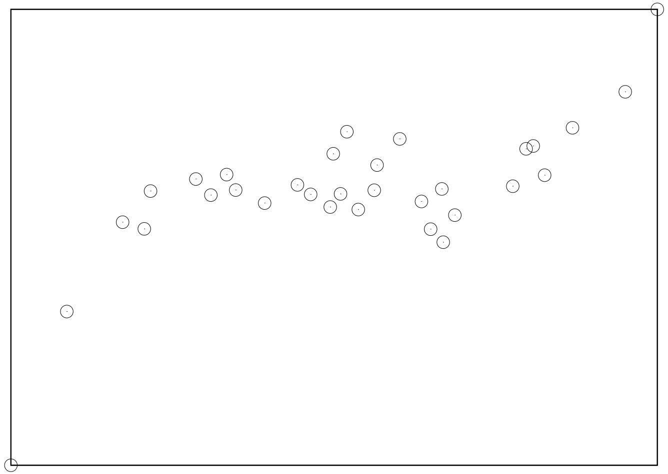
We say that model 2 is more complex than model 1.

Training Error

What happens to **training error** as we increase model complexity?

Start with the simplest model (a constant function)

End with a very high degree polynomial

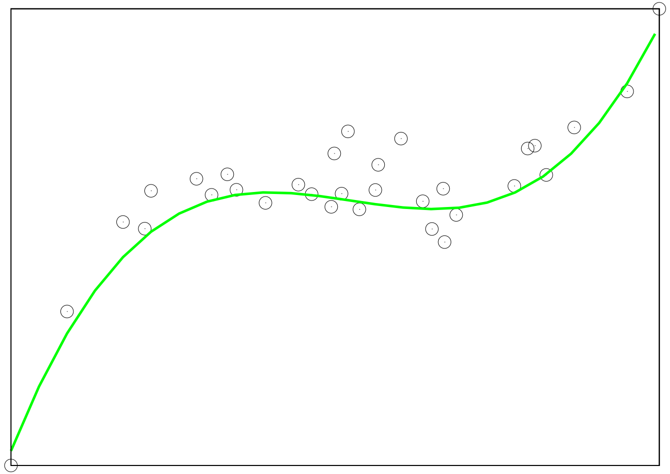


True Error

What happens to **true error** as we increase model complexity?

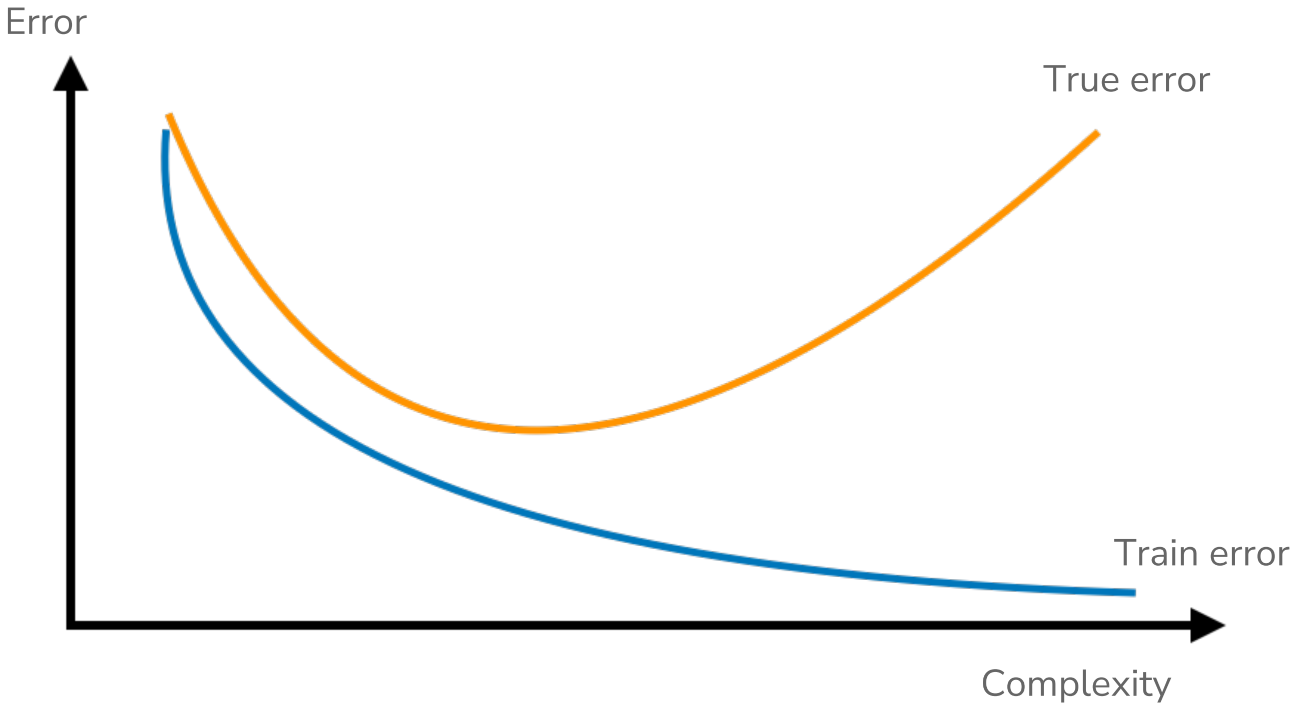
Start with the simplest model (a constant function)

End with a very high degree polynomial



Train/True Error

Compare what happens to train and true error as a function of model complexity

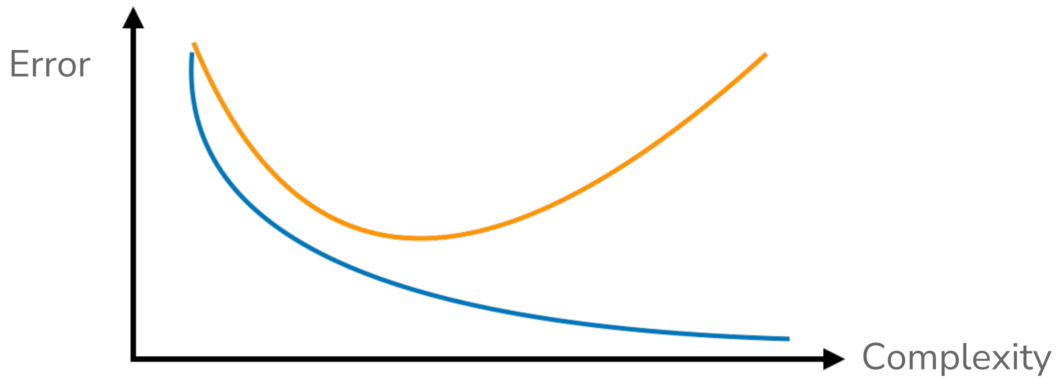


Overfitting

Overfitting happens when we too closely match the training data and fail to generalize.

Overfitting occurs when you train a predictor $\hat{w}$ but there exists another predictor w' from the same model class such that:

$$error_{true}(w') < error_{true}(\hat{w})$$
$$error_{train}(w') > error_{train}(\hat{w})$$

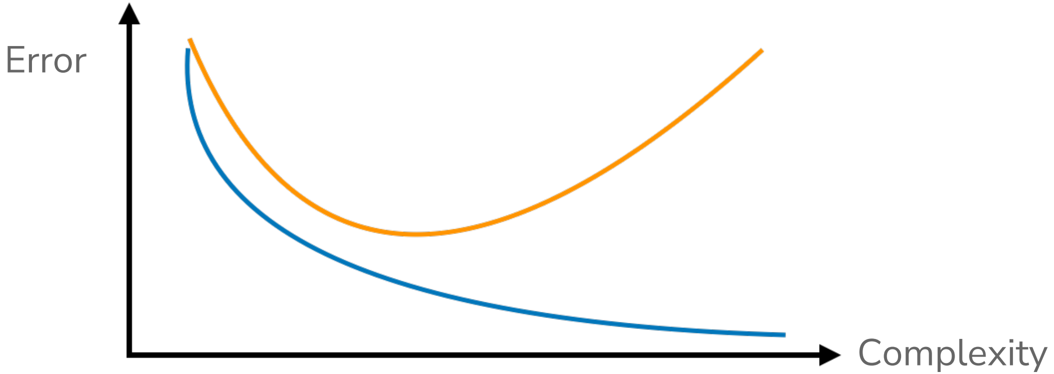


Underfitting

Underfitting happens when a model cannot capture the complex patterns between a training set's features and its output values.

Underfitting occurs when you train a predictor $\hat{w}$ but there exists another predictor w' from the same model class such that:

$$error_{true}(w') < error_{true}(\hat{w})$$
$$error_{train}(w') < error_{train}(\hat{w})$$



Poll Everywhere

Think

1 min

Rank these models from most to least complex.

A. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bathrooms)$

B. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bathrooms) + w_3(school\ rank)$

C. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bed) + w_3(\#bath) + w_4(age)$

D. $y = w_0 + w_1(sq.ft.) + w_2(sq.ft.)^2 + w_3(\#\ bathrooms)$

Poll Everywhere

Group

1.5 min

Rank these models from most to least complex.

A. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bathrooms)$

B. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bathrooms) + w_3(school\ rank)$

C. $y = w_0 + w_1(sq.ft.) + w_2(\#\ bed) + w_3(\#bath) + w_4(age)$

D. $y = w_0 + w_1(sq.ft.) + w_2(sq.ft.)^2 + w_3(\#\ bathrooms)$

Bias-Variance Tradeoff

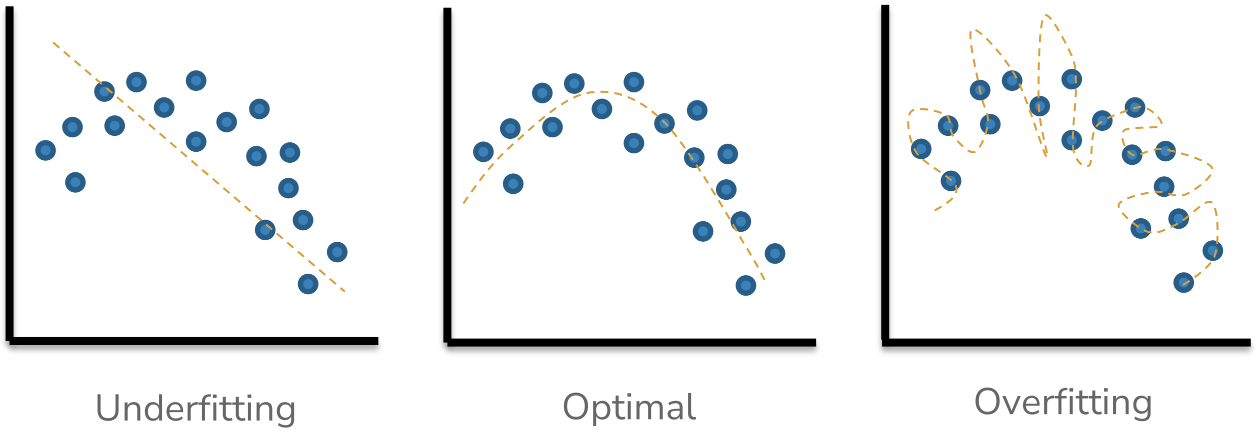
Underfitting / Overfitting

The ability to overfit/underfit is a knob we can turn based on the model complexity.

More complex => easier to overfit

Less complex => easier to underfit

In a bit, we will talk about how to chose the "just right", but now we want to look at this phenomena of overfitting/underfitting from another perspective.



Signal vs. Noise

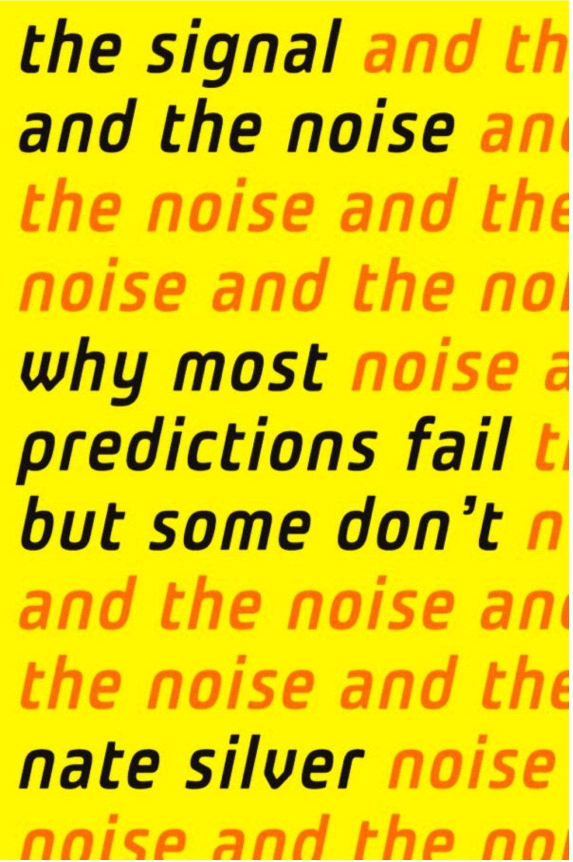
Learning from data relies on balancing two aspects of our data

Signal

Noise

Complex models make it easier to fit too closely to the noise

Simple models have trouble picking up the signal



Source of errors in a model

Total errors for a machine learning model comes from 3 types:

Bias

Variance

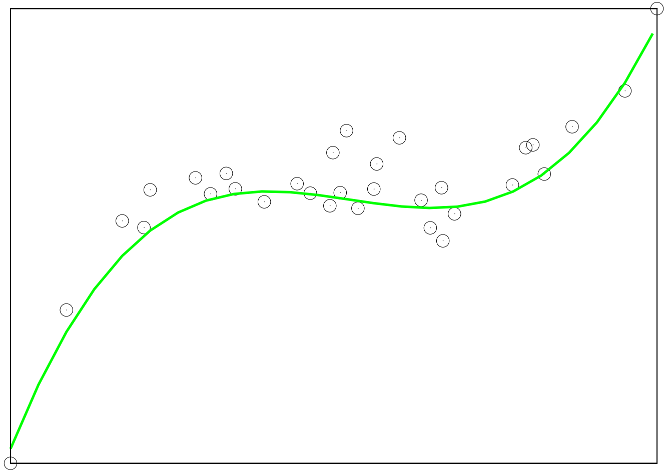
Irreducible Error

Irreducible error is the one that we can't avoid or possibly eliminate. They are caused by elements outside of our control, such as noise from observations.

Bias

A model that is too simple fails to fit the signal. In some sense, this signifies a fundamental limitation of the model we are using to fail to fit the signal. We call this type of error **bias**.

Bias is the difference between the average prediction of our model **and** the expected value which we are trying to predict.

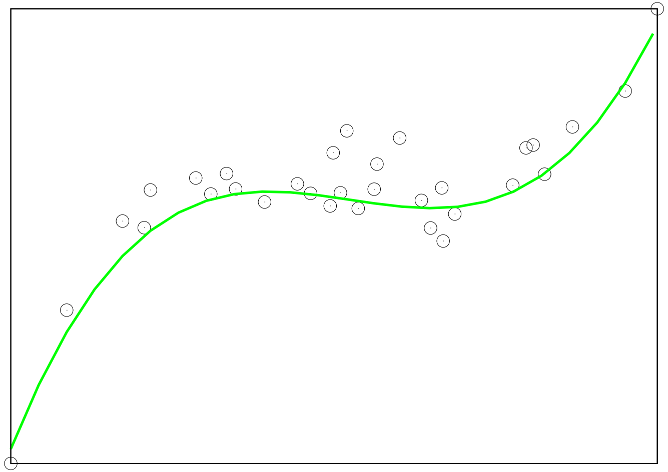


Low complexity (simple) models tend to have high bias.

Variance

A model that is too complicated for the task overly fits to small fluctuations. The flexibility of the complicated model makes it capable of memorizing answers rather than learning general patterns. This contributes to the error as **variance**.

Variance is the variability in the model prediction, meaning how much the predictions will change if a different training dataset is used.



High complexity models tend to have high variance.

Poll Everywhere

Think

1 min

What are some real-world / human analogies for each of these concepts?

- Overfitting / Underfitting
- Train Set / Test Set
- Bias
- Variance

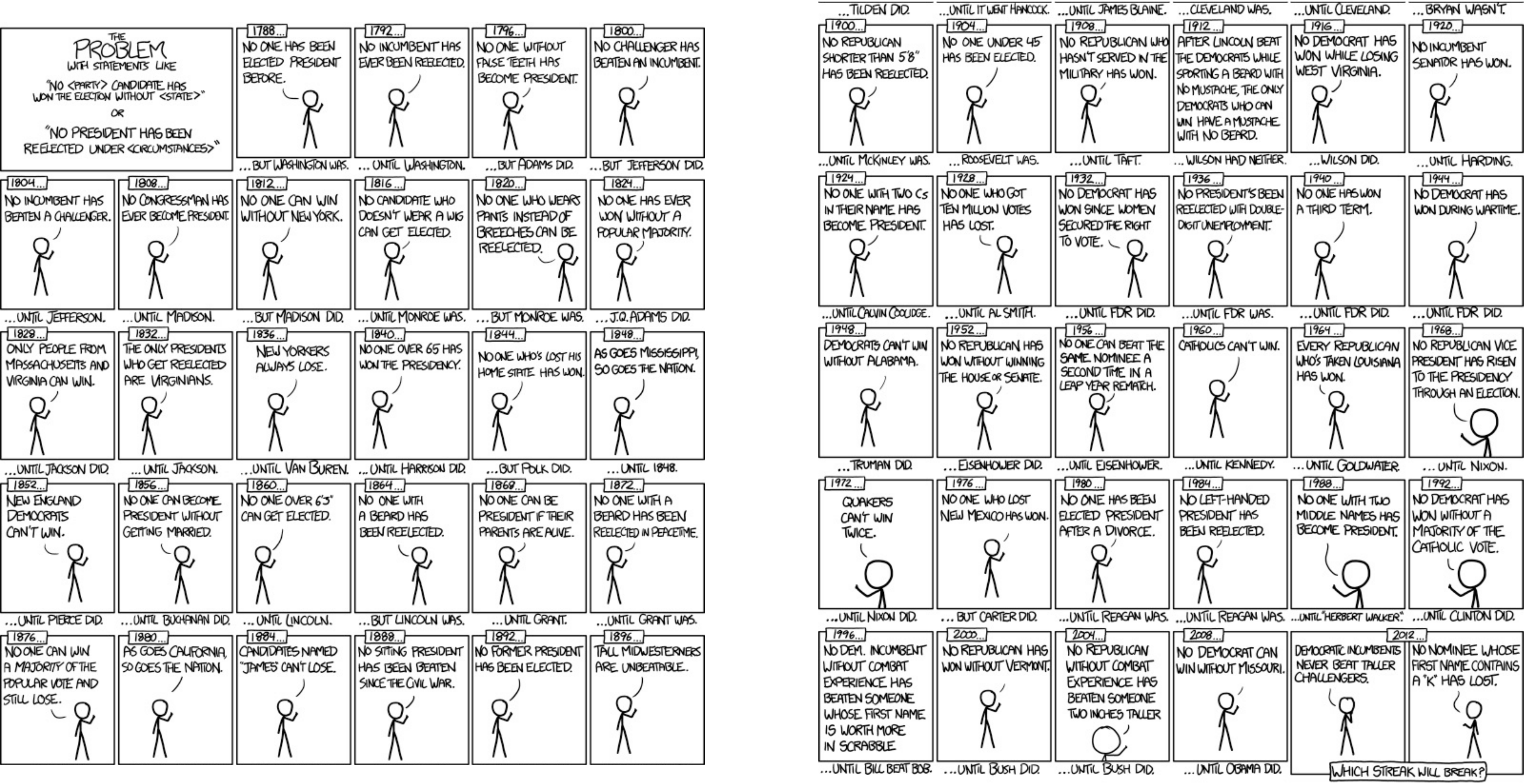
Poll Everywhere

Group

2 mins

What are some real-world / human analogies for each of these concepts?

- Overfitting / Underfitting
- Train Set / Test Set
- Bias
- Variance


THE PROBLEM WITH STATEMENTS LIKE
"NO <PARTY> CANDIDATE HAS WON THE ELECTION WITHOUT <STATE>"
OR
"NO PRESIDENT HAS BEEN REELECTED UNDER <CIRCUMSTANCES>"
1788...
NO ONE HAS BEEN ELECTED PRESIDENT BEFORE.
...BUT WASHINGTON WAS.
1792...
NO INCUMBENT HAS EVER BEEN REELECTED.
...UNTIL WASHINGTON.
1796...
NO ONE WITHOUT FALSE TEETH HAS BECOME PRESIDENT.
...BUT ADAMS DID.
1800...
NO CHALLENGER HAS BEATEN AN INCUMBENT.
...BUT JEFFERSON DID.
1804...
NO INCUMBENT HAS BEATEN A CHALLENGER.
...UNTIL JEFFERSON.
1808...
NO CONGRESSMAN HAS EVER BECOME PRESIDENT.
...UNTIL MADISON.
1812...
NO ONE CAN WIN WITHOUT NEW YORK.
...BUT MADISON DID.
1816...
NO CANDIDATE WHO DOESN'T WEAR A WIG CAN GET ELECTED.
...UNTIL MONROE WAS.
1820...
NO ONE WHO WEARS PANTS INSTEAD OF BREECHES CAN BE REELECTED.
...BUT MONROE WAS.
1824...
NO ONE HAS EVER WON WITHOUT A POPULAR MAJORITY.
...J.Q. ADAMS DID.
1828...
ONLY PEOPLE FROM MASSACHUSETTS AND VIRGINIA CAN WIN.
...UNTIL JACKSON DID.
1832...
THE ONLY PRESIDENTS WHO GET REELECTED ARE VIRGINIANS.
...UNTIL JACKSON.
1836...
NEW YORKERS ALWAYS LOSE.
...UNTIL VAN BUREN.
1840...
NO ONE OVER 65 HAS WON THE PRESIDENCY.
...UNTIL HARRISON DID.
1844...
NO ONE WHO'S LOST HIS HOME STATE HAS WON.
...BUT POLK DID.
1848...
AS GOES MISSISSIPPI, SO GOES THE NATION.
...UNTIL 1848.
1852...
NEW ENGLAND DEMOCRATS CAN'T WIN.
...UNTIL PIERCE DID.
1856...
NO ONE CAN BECOME PRESIDENT WITHOUT GETTING MARRIED.
...UNTIL BUCHANAN DID.
1860...
NO ONE OVER 6'3" CAN GET ELECTED.
...UNTIL LINCOLN.
1864...
NO ONE WITH A BEARD HAS BEEN REELECTED.
...BUT LINCOLN WAS.
1868...
NO ONE CAN BE PRESIDENT IF THEIR PARENTS ARE ALIVE.
...UNTIL GRANT.
1872...
NO ONE WITH A BEARD HAS BEEN REELECTED IN PEACETIME.
...UNTIL GRANT WAS.
1876...
NO ONE CAN WIN A MAJORITY OF THE POPULAR VOTE AND STILL LOSE.
...TILDEN DID.
1880...
AS GOES CALIFORNIA, SO GOES THE NATION.
...UNTIL IT WENT HANCOCK.
1884...
CANDIDATES NAMED "JAMES" CAN'T LOSE.
...UNTIL JAMES BLAINE.
1888...
NO SITTING PRESIDENT HAS BEEN BEATEN SINCE THE CIVIL WAR.
...CLEVELAND WAS.
1892...
NO FORMER PRESIDENT HAS BEEN ELECTED.
...UNTIL CLEVELAND.
1896...
TALL MIDWESTERNERS ARE UNBEATABLE.
...BRYAN WASN'T.
1900...
NO REPUBLICAN SHORTER THAN 5'8" HAS BEEN REELECTED.
...UNTIL McKINLEY WAS.
1904...
NO ONE UNDER 45 HAS BEEN ELECTED.
...ROOSEVELT WAS.
1908...
NO REPUBLICAN WHO HASN'T SERVED IN THE MILITARY HAS WON.
...UNTIL TAFT.
1912...
AFTER LINCOLN BEAT THE DEMOCRATS WHILE SPORTING A BEARD WITH NO MUSTACHE, THE ONLY DEMOCRATS WHO CAN WIN HAVE A MUSTACHE WITH NO BEARD.
...WILSON HAD NEITHER.
1916...
NO DEMOCRAT HAS WON WHILE LOSING WEST VIRGINIA.
...WILSON DID.
1920...
NO INCUMBENT SENATOR HAS WON.
...UNTIL HARDING.
1924...
NO ONE WITH TWO Cs IN THEIR NAME HAS BECOME PRESIDENT.
...UNTIL CALVIN COOLIDGE.
1928...
NO ONE WHO GOT TEN MILLION VOTES HAS LOST.
...UNTIL AL SMITH.
1932...
NO DEMOCRAT HAS WON SINCE WOMEN SECURED THE RIGHT TO VOTE.
...UNTIL FDR DID.
1936...
NO PRESIDENT'S BEEN REELECTED WITH DOUBLE-DIGIT UNEMPLOYMENT.
...UNTIL FDR WAS.
1940...
NO ONE HAS WON A THIRD TERM.
...UNTIL FDR DID.
1944...
NO DEMOCRAT HAS WON DURING WARTIME.
...UNTIL FDR DID.
1948...
DEMOCRATS CAN'T WIN WITHOUT ALABAMA.
...TRUMAN DID.
1952...
NO REPUBLICAN HAS WON WITHOUT WINNING THE HOUSE OR SENATE.
...EISENHOWER DID.
1956...
NO ONE CAN BEAT THE SAME NOMINEE A SECOND TIME IN A LEAP YEAR REMATCH.
...UNTIL EISENHOWER.
1960...
CATHOLICS CAN'T WIN.
...UNTIL KENNEDY.
1964...
EVERY REPUBLICAN WHO'S TAKEN LOUISIANA HAS WON.
...UNTIL GOLDWATER.
1968...
NO REPUBLICAN VICE PRESIDENT HAS RISEN TO THE PRESIDENCY THROUGH AN ELECTION.
...UNTIL NIXON.
1972...
QUAKERS CAN'T WIN TWICE.
...UNTIL NIXON DID.
1976...
NO ONE WHO LOST NEW MEXICO HAS WON.
...BUT CARTER DID.
1980...
NO ONE HAS BEEN ELECTED PRESIDENT AFTER A DIVORCE.
...UNTIL REAGAN WAS.
1984...
NO LEFT-HANDED PRESIDENT HAS BEEN REELECTED.
...UNTIL REAGAN WAS.
1988...
NO ONE WITH TWO MIDDLE NAMES HAS BECOME PRESIDENT.
...UNTIL "HERBERT WALKER."
1992...
NO DEMOCRAT HAS WON WITHOUT A MAJORITY OF THE CATHOLIC VOTE.
...UNTIL CLINTON DID.
1996...
NO DEM. INCUMBENT WITHOUT COMBAT EXPERIENCE HAS BEATEN SOMEONE WHOSE FIRST NAME IS WORTH MORE IN SCRABBLE.
...UNTIL BILL BEAT BOB.
2000...
NO REPUBLICAN HAS WON WITHOUT VERMONT.
...UNTIL BUSH DID.
2004...
NO REPUBLICAN WITHOUT COMBAT EXPERIENCE HAS BEATEN SOMEONE TWO INCHES TALLER
...UNTIL BUSH DID.
2008...
NO DEMOCRAT CAN WIN WITHOUT MISSOURI.
...UNTIL OBAMA DID.
2012...
DEMOCRATIC INCUMBENTS NEVER BEAT TALLER CHALLENGERS.
NO NOMINEE WHOSE FIRST NAME CONTAINS A "K" HAS LOST.
WHICH STREAK WILL BREAK?

Bias-Variance Tradeoff

Tradeoff between bias and variance:

Simple models: High bias + Low variance

Complex models: Low bias + High variance

Source of errors for a particular model $\hat{f}$ using MSE loss function:

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{bias}[\hat{f}(x)]^2 + \text{var}(\hat{f}(x)) + \sigma_\epsilon^2$$

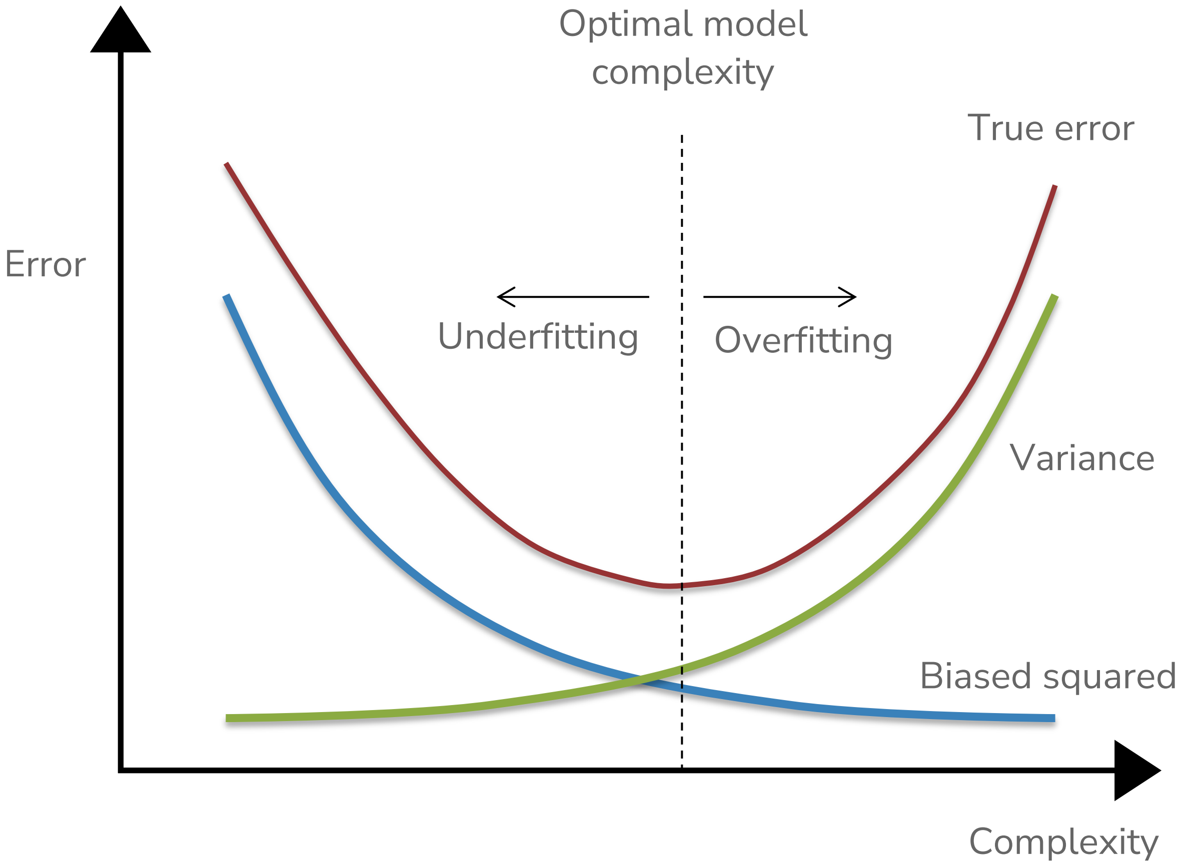
Error = Biased squared + Variance + Irreducible Error

Bias-Variance Tradeoff

Visually, this looks like the following!

$$Error = Bias^2 + Variance + Irredicible\ Error$$

Bias – Variance Tradeoff



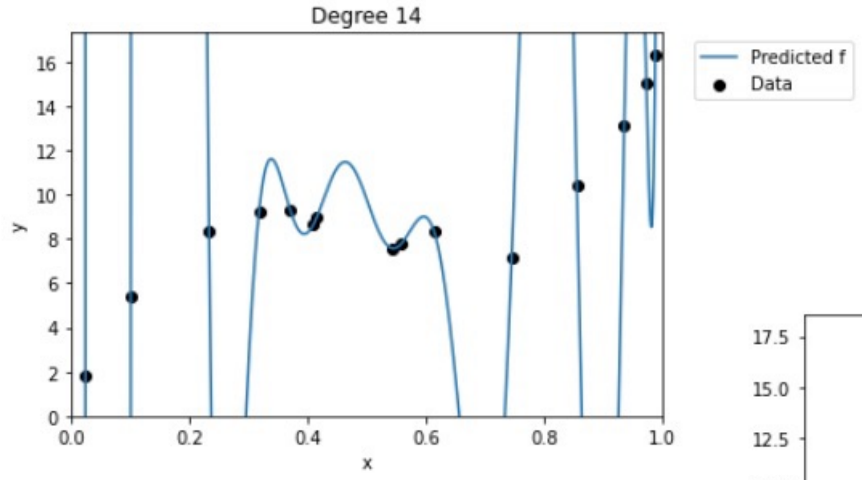
Dataset Size

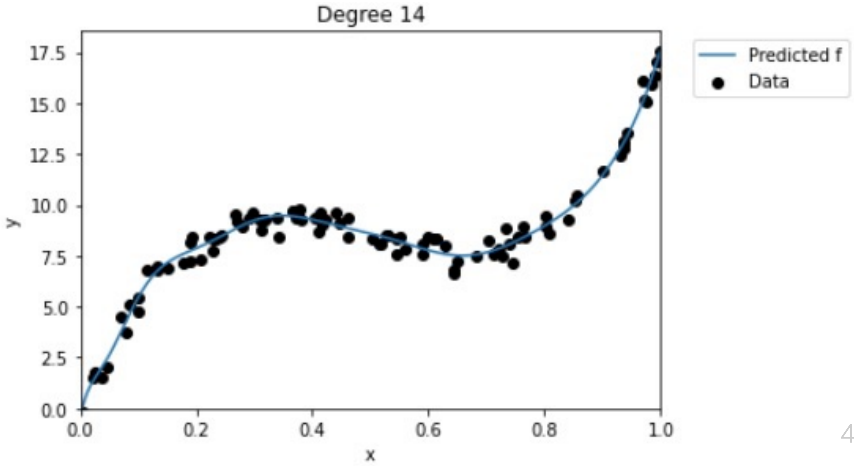
So far our entire discussion of error assumes a fixed amount of data. What happens to our error (true error and training error) as we get more data?

Dataset Size

Model complexity doesn't depend on the size of the training set

The larger the training set, the lower the variance of the model, thus less overfitting





Brain Break

Choosing Complexity

Choosing Complexity

So far we have talked about the affect of using different complexities on our error. Now, how do we choose the right one?

Poll Everywhere

Think

1 min

pollev.com/cs416

Suppose I wanted to figure out the right degree polynomial for my dataset (we'll try p from 1 to 20). What procedure should I use to do this? Pick the best option

For each possible degree polynomial p:

- Train a model with degree p on the training set, pick p that has the lowest test error
- Train a model with degree p on the training set, pick p that has the highest test error
- Train a model with degree p on the test set, pick p that has the lowest test error
- Train a model with degree p on the test set, pick p that has the highest test error
- None of the above

Poll Everywhere

Think

2 min

pollev.com/cs416

Suppose I wanted to figure out the right degree polynomial for my dataset (we'll try p from 1 to 20). What procedure should I use to do this? Pick the best option

For each possible degree polynomial p:

- Train a model with degree p on the training set, pick p that has the lowest test error
- Train a model with degree p on the training set, pick p that has the highest test error
- Train a model with degree p on the test set, pick p that has the lowest test error
- Train a model with degree p on the test set, pick p that has the highest test error
- None of the above

Choosing Complexity

We can't just choose the model that has the lowest **train** error because that will favor models that overfit!

It then seems like our only other choice is to choose the model that has the lowest **test** error (since that is our approximation of the true error)

This is almost right. However, the test set has been **tampered**, thus is no longer is an unbiased estimate of the true error.

We didn't technically train the model on the test set (that's good), but we chose **which model** to use based on the performance of the test set.

- It's no longer a stand in for "the unknown" since we probed it many times to figure out which model would be best.

NEVER EVER EVER touch the test set until the end. You only use it ONCE to evaluate the performance of the best model you have selected during training.

Choosing Complexity

We will talk about two ways to pick the model complexity without ruining our test set.

- Using a validation set
- Doing (k-fold) cross validation

Validation Set

So far we have divided our dataset into train and test

We can't use Test to choose our model complexity, so instead, break up Train into ANOTHER dataset

We will pick the model that does best on validation. Note that this now makes the validation error of the "best" model a biased estimate of true error. The test error will be an unbiased estimate though since we never looked at it!

Validation Set

The process generally goes

```
train, validation, test = random_split(dataset)
for each model complexity p:
    model = train_model(model_p, train)
    val_err = error(model, validation)
    keep track of p and model with smallest val_err
return best p & error(model, test)
```

Validation Set

Pros

Easy to describe and implement

Pretty fast

- Only requires training a model and predicting on the validation set for each complexity of interest

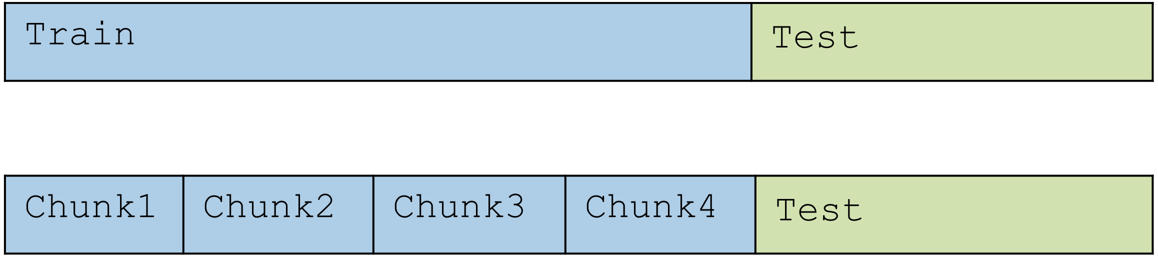
Cons

- Have to sacrifice even more training data
- Prone to overfitting*

Cross-Validation

Clever idea: Use many small validation sets without losing too much training data.

Still need to break off our test set like before. After doing so, break the training set into k chunks.



For a given model complexity, train it k times. Each time use all but one chunk and use that left out chunk to determine the validation error.

Cross Validation

For a set of hyperparameters, perform Cross Validation on k folds

Cross-Validation

The process generally goes

```
chunk_1, …, chunk_k, test = random_split(dataset)
for each model complexity p:
    for i in [1, k]:
        model = train_model(model_p, chunks - i)
        val_err = error(model, chunk_i)
    avg_val_err = average val_err over chunks
    keep track of p with smallest avg_val_err
return model trained on train (all chunks) with
best p & error(model, test)
```

Cross-Validation

Pros

- Prevent overfitting: By training the model on multiple folds instead of only 1 training set, this learns the model with the best generalization capabilities.
- Don't have to actually get rid of any training data!

Cons

- Slow. For each model selection, we have to train k times
- Very computationally expensive

Cross-Validation

For best results, need to make k really big

- Theoretical best estimator is to use $k = n$
  - Called "Leave One Out Cross Validation"
- In practice, people use $k = 5$ to 10

Recap

Theme: Assess the performance of our models

Ideas:

Model complexity

Train vs. Test vs. True error

Overfitting and Underfitting

Bias-Variance Tradeoff

Error as a function of train set size

Choosing best model complexity

- Validation set
- Cross Validation