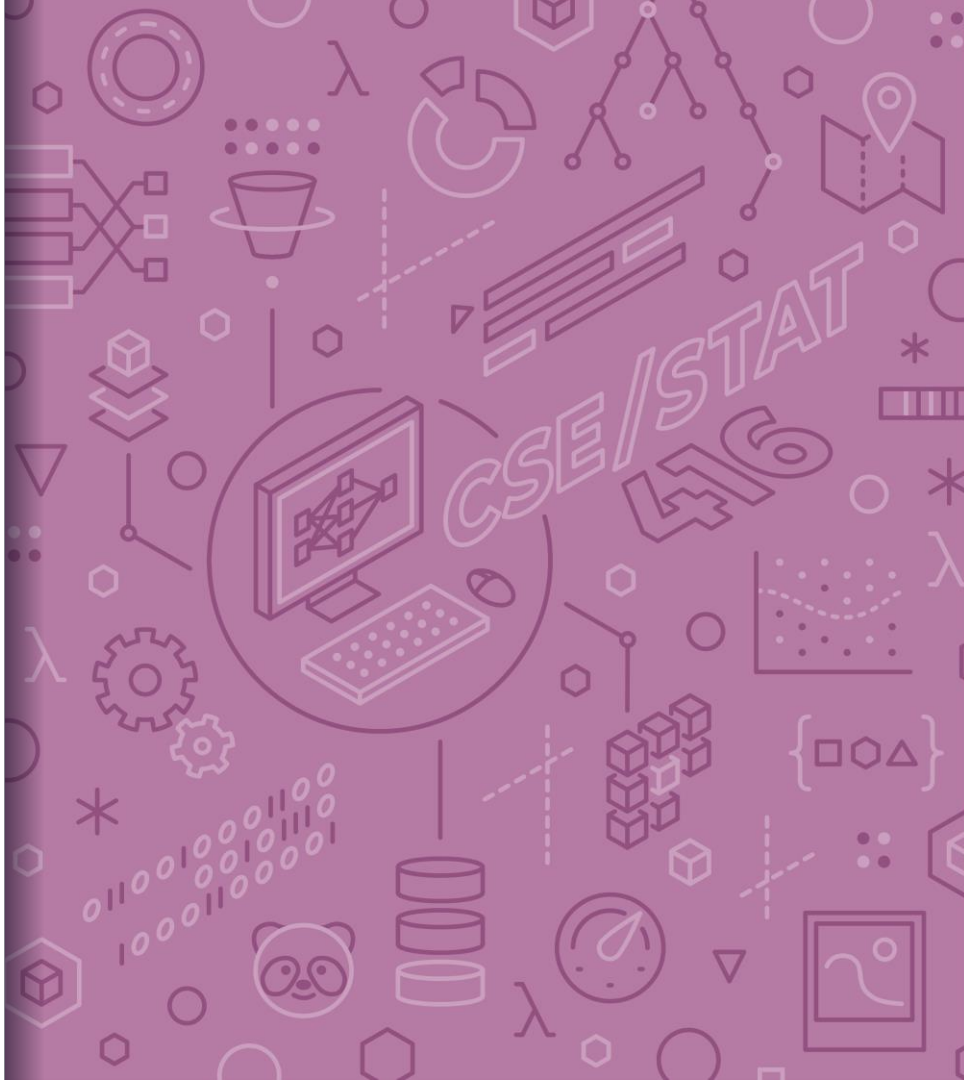
Questions? Raise hand or Sli.do #cs416

CSE/STAT 416

Introduction + Regression

Hunter Schafer
University of Washington
March 27, 2023

Before Class: Introduce yourself to your neighbors! What are you excited to learn about this quarter?
Listening to: The Golf Club



Machine Learning is changing the world.

Google Trends Home Explore Trending Now Sign in

machine learning
Search term

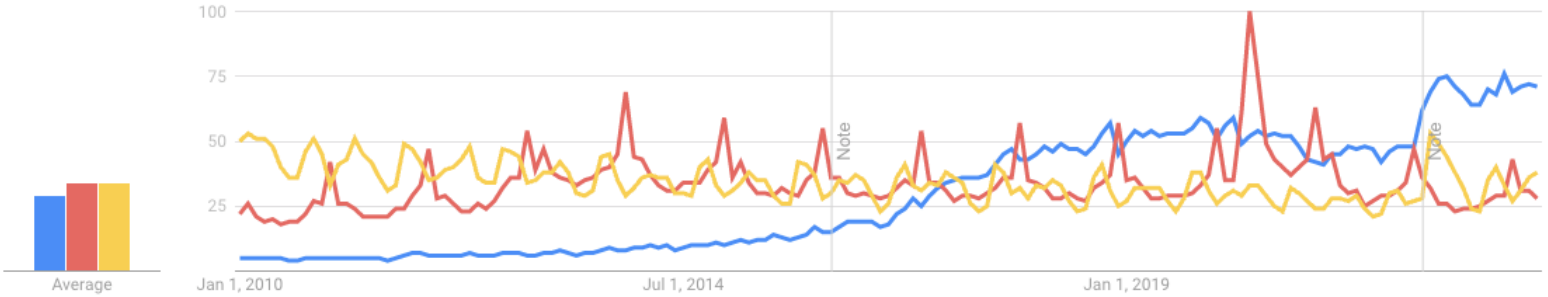
chocolate chip co...
Search term

united nations
Search term

+ Add comparison

Worldwide | 1/1/10 - 3/26/23 | All categories | Web Search

Interest Over Time



Google Trends Home Explore Trending Now Sign in

machine learning
Search term

chocolate chip co...
Search term

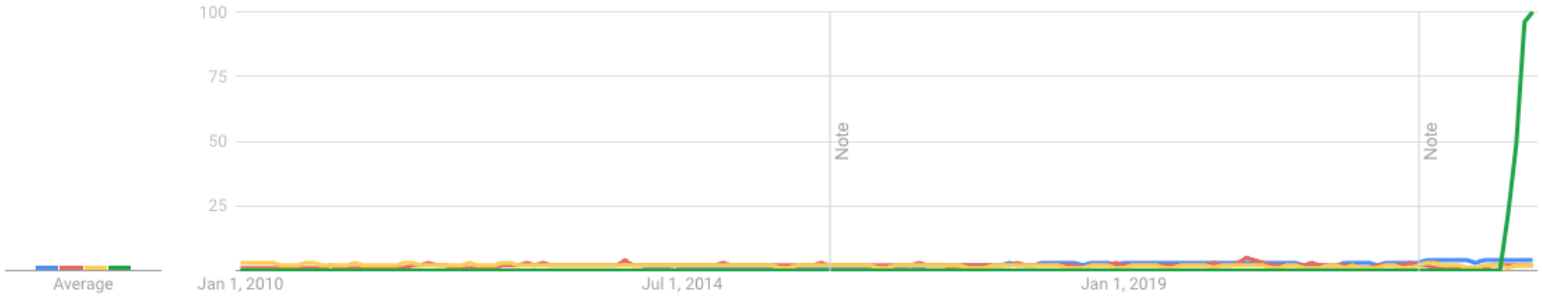
united nations
Search term

chatgpt
Search term

+

Worldwide | 1/1/10 - 3/26/23 | All categories | Web Search

Interest Over Time



It's Everywhere!

amazon
Retail

Google
PageRank
Search

livingsocial
Coupons

NETFLIX
Movie Distribution

Obama'08
Campaigning

Zillow
Real Estate

Avvo
Legal Advice

Google Adsense
Advertising

glassdoor
Human Resources

eHarmony
Dating

LinkedIn
Networking

RelateIQ
CRM

PANDORA
Music

fitbit
Wearables

Disruptive companies differentiated by

INTELLIGENT APPLICATIONS

using

Machine Learning

It’s Everywhere...

It’s Everywhere...

Eddy Dever
@EddyDever

Follow

It’s terrifying that both of these things are true at the same time in this world:

• computers drive cars around

• the state of the art test to check that you’re not a computer is whether you can successful identify stop signs in pictures

12:26 AM - 13 May 2018

5,644 Retweets **12,727** Likes

It’s Everywhere...

Object Detection

It’s Everywhere...

Face Detection

Microsoft Plans to Eliminate Face Analysis Tools in Push for ‘Responsible A.I.’

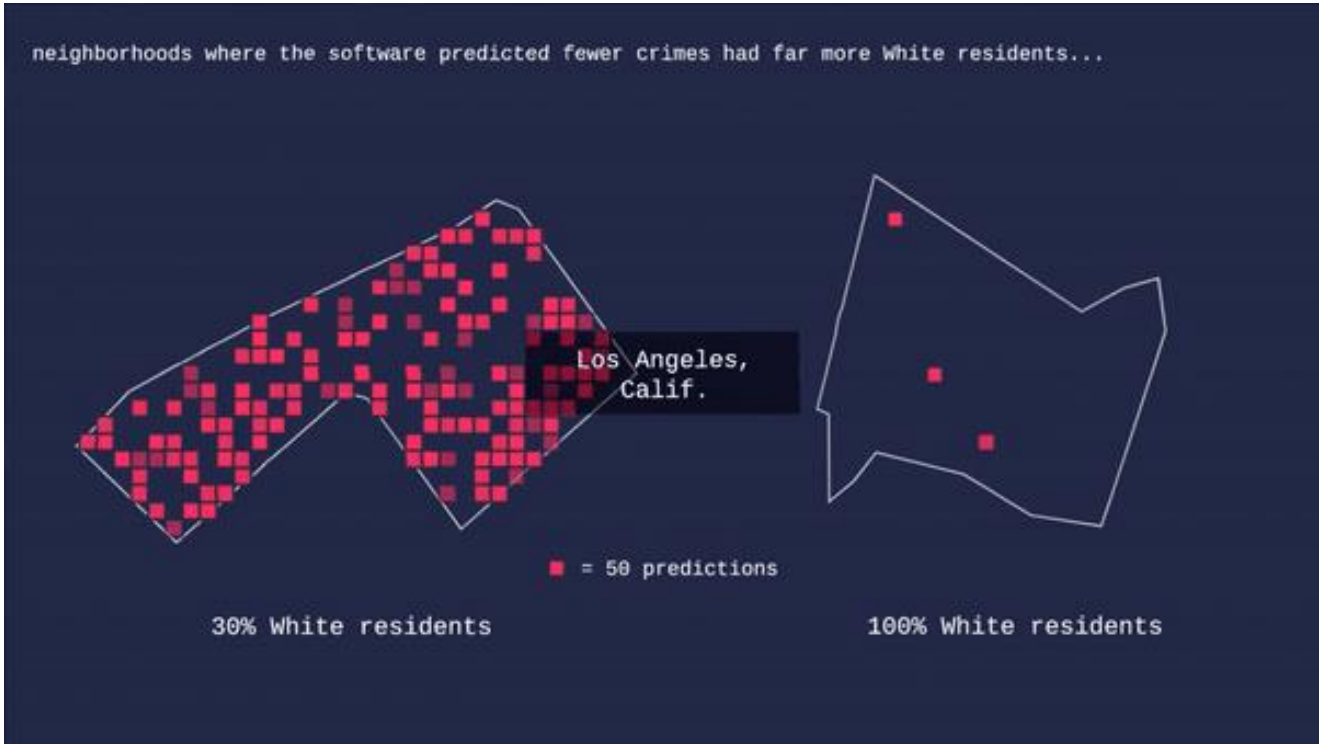
The technology giant will stop offering automated tools that predict a person’s gender, age and emotional state and will restrict the use of its facial recognition tool.

June 21, 2022, 12:02 p.m. ET

Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

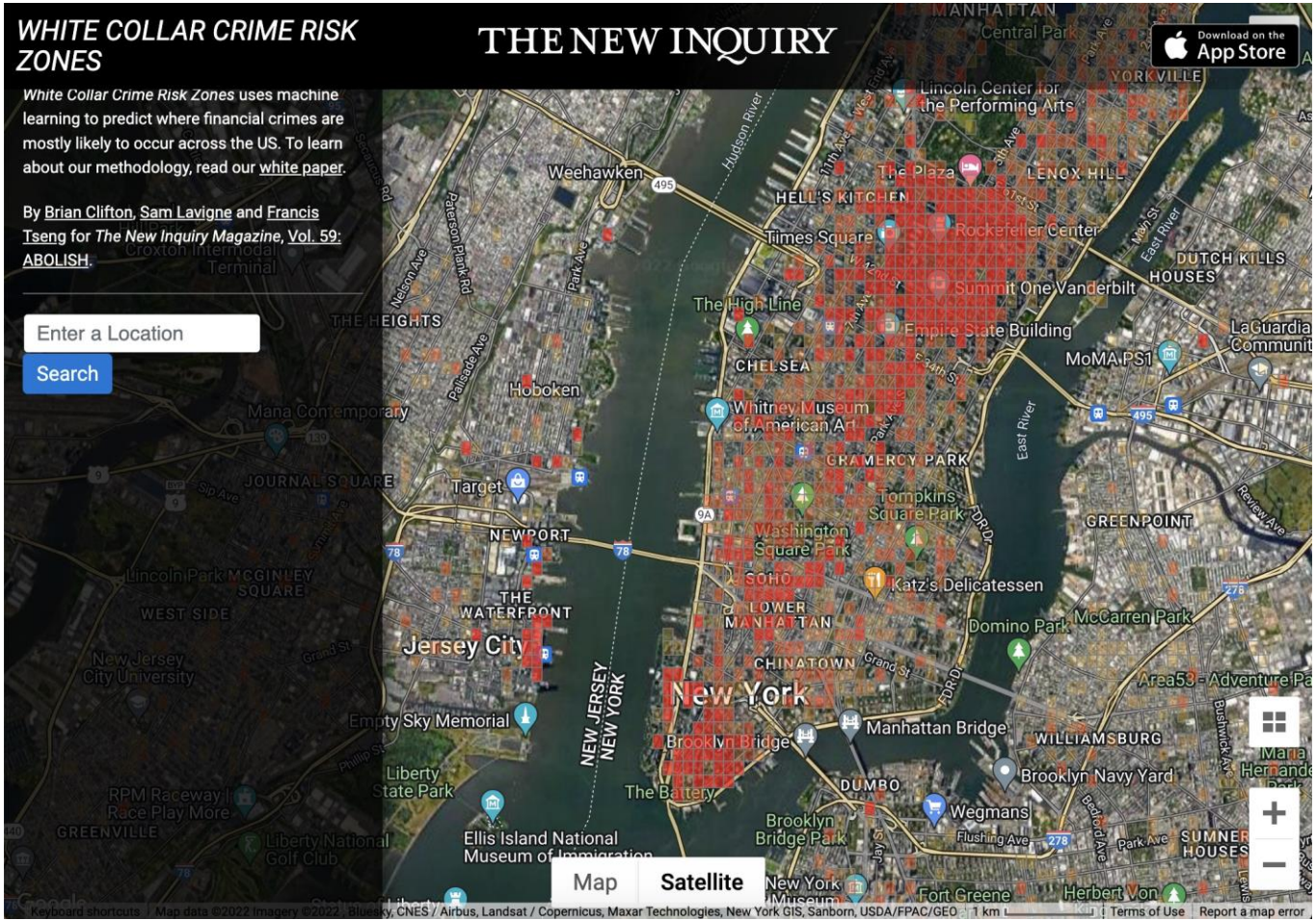
It’s Everywhere...

Predictive Policing



It’s Everywhere...

Predictive Policing



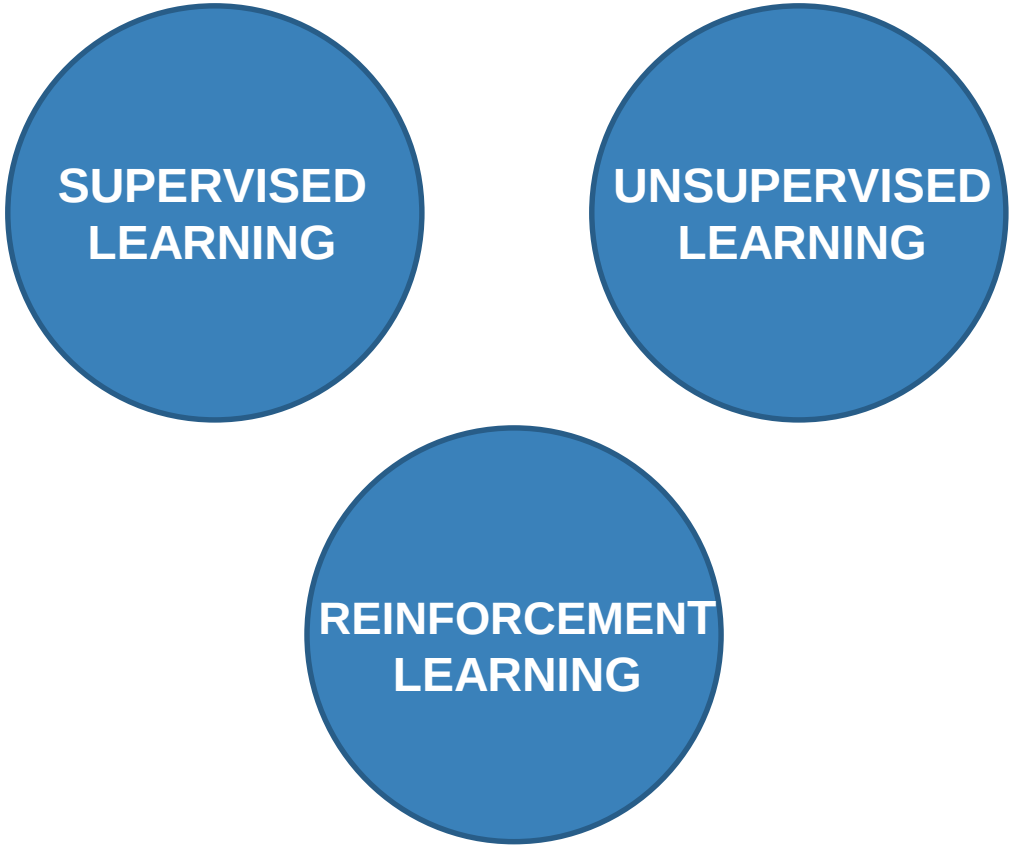
What is Machine Learning?

Generically (and vaguely)

Machine Learning (ML) is the study of algorithms that improve their **performance** at some **task** with **experience.**

Tom Mitchell (1998): a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Taxonomy of Machine Learning (Based on tasks)



Taxonomy of Machine Learning (Based on tasks)

1. **Supervised Learning**
   - Training data is labeled, where inputs are paired with correct outputs
   - Infers a mapping function from the inputs to outputs
   - ***Examples:*** *image classification, stock price predictions*

2. **Unsupervised Learning**
   - Analyze and cluster unlabeled datasets
   - Discover patterns or data categorization without the need for human intervention
   - ***Examples:*** *DNA clustering, anomaly detection*

3. **Reinforcement Learning**
   - Not covered in this class (you can learn this in CSE 415 / 473 (Introduction to Artificial Intelligence)
   - Agents learn the optimal behaviors to obtain maximum reward through interactions with the environment and observations of how they responds.

Course Overview

This course is broken up into 5 main case studies to explore ML in various contexts/applications.

1. Regression
   - Predicting housing prices
2. Classification
   - Positive/Negative reviews (Sentiment analysis)
3. Document Retrieval + Clustering
   - Find similar news articles
4. Recommender Systems
   - Given past purchases, what do we recommend to you?
5. Deep Learning
   - Recognizing objects in images

Course Topics

Models

- Linear regression, regularized approaches (ridge, LASSO)
- Linear classifiers: logistic regression
- Non-linear models: decision trees
- Nearest neighbors, clustering
- Recommender systems
- Deep learning

Algorithms

- *Gradient descent*
- Boosting
- K-means

Concepts

- Point estimation, MLE
- Loss functions, bias-variance tradeoff, cross-validation
- Sparsity, overfitting, model selection
- Decision boundaries

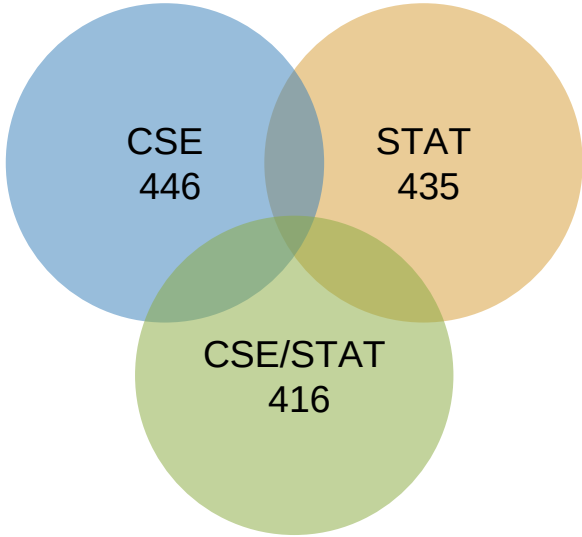
ML Course Landscape

CSE 446

- CSE majors
- Very technical course

STAT 435

- STAT majors
- Very technical course



CSE/STAT 416

- Everyone else!
  - This is a super broad audience!
- Give everyone a strong foundational understanding of ML
  - More breadth than other courses, a little less depth

Level of Course

Our Motto

Everyone should be able to learn machine learning, so our job is to make tough concepts intuitive and applicable.

This means...

- Minimize pre-requisite knowledge
- Focus on important ideas, avoid getting bogged down by math
- Maximize ability to develop and deploy
- Use pre-written libraries to do many tasks
- Learn concepts in case studies

Does not mean course isn't fast paced! There are a lot of concepts to cover!

Course Logistics

Who am I?

- Hunter Schafer
  - Assistant Teaching Professor
  - Paul G. Allen School for Computer Science & Engineering (CSE)
- Office Hours
  - Time: 2:30 – 4:30 on Tuesdays
  - Location: CSE 530
- Contact
  - Course Content + Logistics: EdStem
  - Personal Matters: hschafer@cs.washington.edu

Who are the TAs?

Huong Ngo
she/her
hvn2002@uw

Jake Flynn
he/him
jflynn56@uw

James Cao
he/him
zc68@uw

Lily Shen

lilyshen@uw

Melissa Mitchell
she/her
mcm08@uw

Pavan Kumar Anand
he/him
pka4@uw

Therese Pena Pacio
she/her
tpacio@cs

Learning Reflections

Recap broader context of the past week.

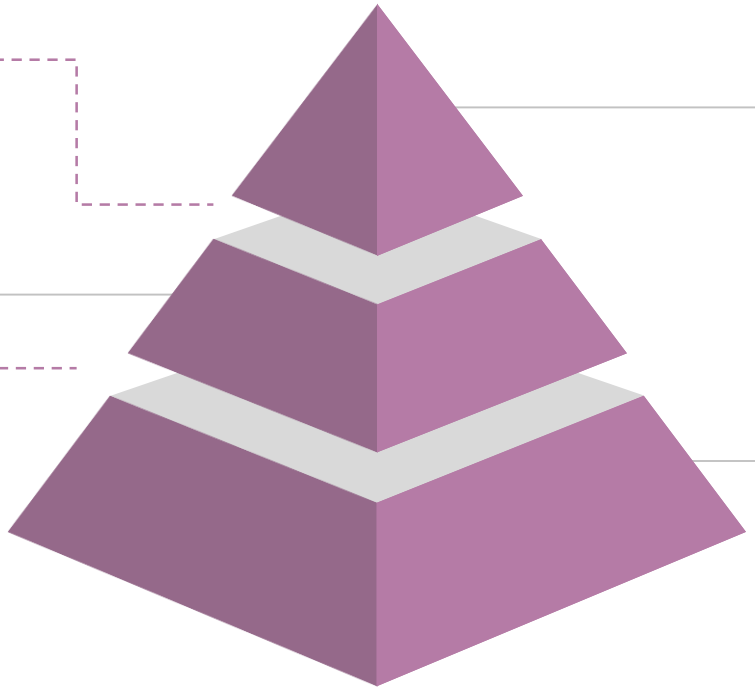
Sections

Practice material covered in **1** in a context where a TA can help you.

The emphasis is still on you l**earning by doing.**

Checkpoints

Test your understanding of the last concept



Homework

With the scaffolding from **1 and 2**, you are probably now capable to tackle the homework. These will be complex and challenging, but you'll continue to **learn by doing.**

Lectures

Introduced to material for the first time. Mixed with activities and demos to give you a chance to **learn by doing.**

No where near mastery yet!

Mon	Tue	Wed	Thur	Fri
Lecture	Nothing	**Lecture**	**Section**	Nothing
Previous CP **Due** Pre-Class **Completed**	HW **Due**	Previous CP **Due** Pre-Class **Completed** Next HW **Released**		Learning Refl. **Due**

- We happen to not record attendance in lectures and section, but attending these sessions is expected
- Panopto for Lecture (on Canvas)

Assessment

- **Weekly Homework Assignments (55%)**
  - **Number**: Approximately 8
  - Each Assignment has two parts that contribute to your grade separately:
    - Programming (35%) - Autograded
    - Conceptual (20%) – Manually Graded
- **Checkpoints (5%)**
  - **Number:** Approximately 20 (each lecture, drop 2)
- **Learning Reflections (10%)**
  - **Number:** Approximately 10 (each week, drop 1)
- **Midterm (10%)**
  - **Format:** Take-Home, online
- **Final Exam (20%)**
  - **Date:** Thursday, June 8 at 8:30 am
  - **Format:** In-person, paper exam. Can bring cheat sheet.

Learning Reflections

- Summary: 2-4 sentences
- List of key concepts (≥ 5)
- Uncertainties

Summary

This week we learned about deep learning and neural networks. Unlike most of the machine learning algorithms that we learned about this quarter, deep learning / neural networks is a bit more new and more complicated in a sense that people have been giving it more attention in recent years and there are not yet any definitive rules to how you have to create models to make them perform the best. At a high level, it gets inputs and can have one or more hidden layers where weights are computed, then spits out the outputs at the end.

Concepts

- Neural networks (NN)
  - Each neuron will have weights and values, and the summation of those values (or with an activation function for non-linear) will go into each neuron in the hidden layer.
  - Each neuron in the hidden layer can have a bias term which is subtracted from its input value.
  - Then based on a certain set of rules for the output, its output will go to the neurons in the next layer, which can be another hidden layer or the output.
  - You keep doing this until you get the outputs.
  - There are no set rules yet as to what is better practice for all cases, so there are a lot of hyperparameters like number of hidden layers or hidden neurons, the activation function, learning rate for gradient descent, batch size, epochs to train, etc.
- Convolutional NN: the idea is to reduce the number of weights that needs to be learned in the NN by using kernels and pools to reduce the number of features.
  - Convolution with kernels
    - A kernel can "slide" across the original input data, generally to find sum of element-wise product between the kernel and overlapping part of the image. The output of this is a smaller version of the original image.
    - You can decide the size of kernel, padding size and values, and stride values
  - Pools
    - Similar to a kernel, but instead of using all the values to be part of the output (through summation, products, etc), just pick a similar value like the min, max, average, median, etc. Typical to use the max pool.
  - Then use a combination of kernel convolution and pooling to get a smaller input for the NN.

Uncertainties

It seems neural networks is a less developed field compared to the other machine learning algorithms that we have learned about this quarter. One thing I wonder is, how do you know it is ready to implement in the world for real life applications, especially when it can have high error rates with even the slightest change to the input or model?

CS 416 Learning Reflection- Week 9

Summary: This week's lectures are the introduction to neural networks. The importance and strength of deep learning for different applications are discussed. The convolutional neural networks (CNN) are detailed with help of object recognition in an image. The regression and classification methods using neural networks and hyper-parameter tuning for these networks are explained.

Concepts:

Neural Networks: It is a simulated representation of neurons in brains having a linear expression using the sum of weights multiplied by inputs. It is then followed either by a linear or nonlinear function for activation. A series of layers are formed using these neurons and complex functions are learned. Almost any function could be learned using neural networks but they require sufficient computational resources and can be executed parallelly using GPUs.

Activation Functions: Various activation functions like the sigmoid, hyperbolic tangent (tanh), Rectified linear unit (ReLU) and soft plus are shown with their respective advantages. ReLU being popular has fragile traits during training. Sigmoids are now being used only at outputs for determining class probabilities and are called softmax.

Overfitting: NN's tend to overfit and it can be addressed by regularizing using dropout conditions and sometimes stopping early or using less number of layers when not demanded by the problem.

Backpropagation: It is a popular algorithm to learn coefficients using the predictions obtained from the forward pass. The error using metrics like RSS and cross-entropy loss is used to adjust the weights for better prediction later.

Hyperparameters: They are the most important after forming an architecture for a machine learning problem, especially NN's. The number of epochs, activation functions, layers, batch size, and learning rate are all important in their own aspect. They have to be carefully chosen and later optimized.

CNN: Primarily being used for images, convolutions help to reduce the number of inputs by combing information about local pixels. Typically a predefined kernel is swept over the input using a stride length and the values of the weights are learned in later epochs. Many times a pooling layer is used to downsample channels separately. Only a final fully connected layer is used to form a neural network just before the output. Transfer learning these days can help in faster initialization and reduce training times for common data available publicly. CNN's are sensitive to transformation or external noise added and can get confused.

Uncertainties: How to handle various channels of input and does it always have to be an image for CNN?

Homework Logistics

- **Late Days**
  - 6 Free Late Days for the whole quarter.
  - Can use up to 2 Late Days on any assignment.
  - Each Late Day used after the 6 Free Late Days results in a -10% on that assignment
  - Learning reflections and checkpoints – no late days. Can be turned in up to a week later for 50% credit.
- **Collaboration**
  - You are encouraged to discuss assignments and concepts **at a high level**
    - If you are reading off parts of your solution, it's likely not high level
    - Discuss process, not answers!
  - All code and answers submitted must be your own
- **Turn In**
  - Concept portions and Learning reflections are turned in on Gradescope
  - Everything else (Programming portion and checkpoints) are turned in on EdStem

Getting Help

The best place to get **asynchronous help** is EdStem. You can post questions (publicly or privately) to get help from peers or members of the course staff.

- You're encouraged to respond with your ideas to other posts!

The best place to get **synchronous help** is office hours or to form a study group.

- The synchronous nature of office hours is the best for back-and-forth conversations like clarifying confusing topics or helping you learn the process of debugging your assignment.

Slido #cs416

Case Study 1

Regression:
Housing Prices

Fitting Data

Goal: Predict how much my house is worth

Have data from my neighborhood

n data points

(x_i, y_i)

or

$(x^{(i)}, y^{(i)})$

$$
\begin{aligned}
(x_1, y_1) &= (2318\, sq.ft.,\ \$315k) \\
(x_2, y_2) &= (1985\, sq.ft.,\ \$295k) \\
(x_3, y_3) &= (2861\, sq.ft.,\ \$370k) \\
&\vdots \qquad\qquad \vdots \\
(x_n, y_n) &= (2055\, sq.ft.,\ \$320k)
\end{aligned}
$$

Assumption:

There is a relationship between $y \in \mathbb{R}$ and $x \in \mathbb{R}^d$

notation for d features

(right now d=1)

approx equals

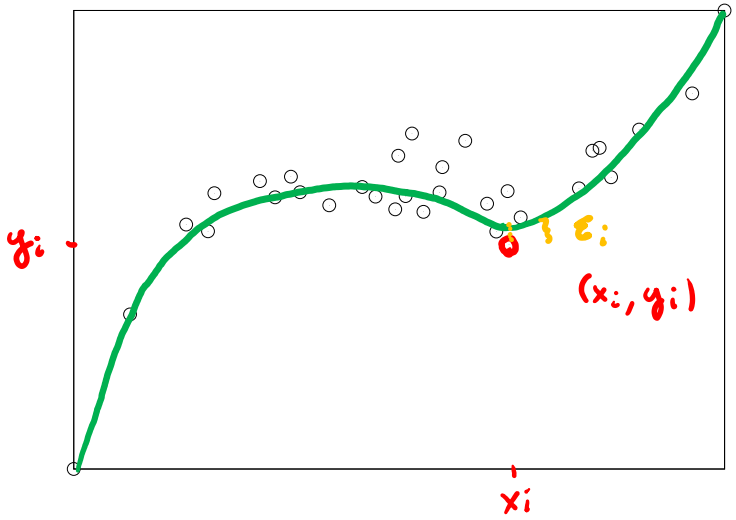
$$y \approx f(x)$$

x is the **input data.** Can potentially have many inputs

y is the **outcome/response/target/label/dependent variable**

Model

A **model** is how we *assume* the world works



Regression model:

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

Assume $E[\epsilon_i] = 0$

"Essentially, all models are wrong, but some are useful."

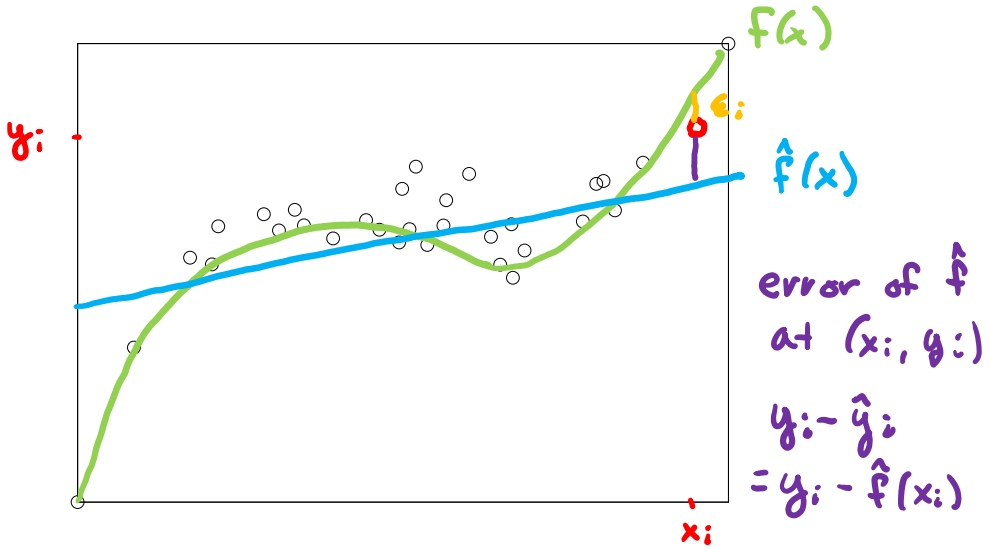
- George Box, 1987

Predictor

We don't know f! We need to learn it from the data!

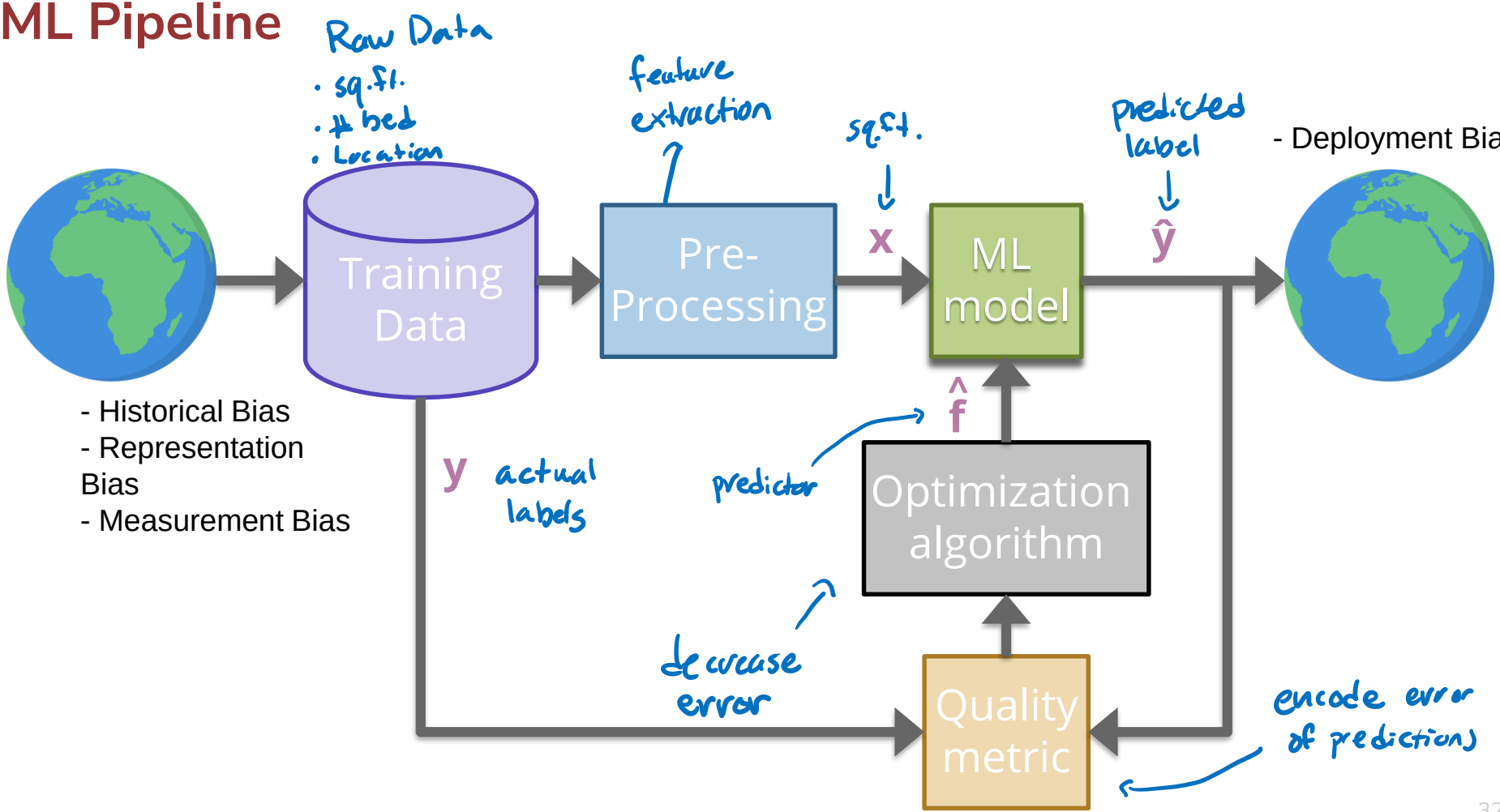
Use machine learning to learn a predictor $\hat{f}$ from the data

For a given input x, predict: $\hat{y} = \hat{f}(x)$



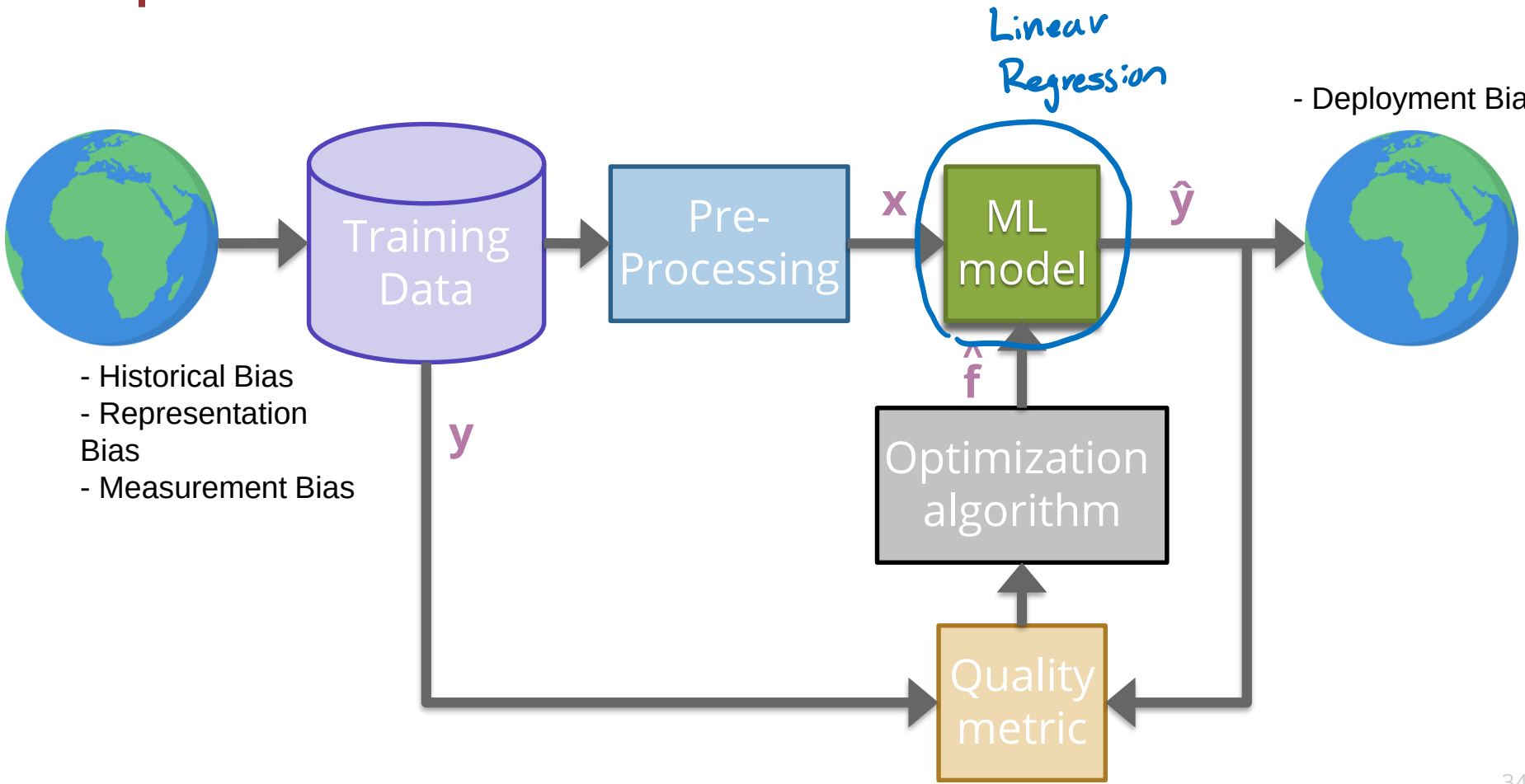
Small error on an example, means we had a good fit *for that point*

ML Pipeline


Raw Data
. sq.ft.
. # bed
. Location
feature extraction
sq.ft.
predicted label
- Deployment Bias
Training Data
Pre-Processing
x
ML model
ŷ
- Historical Bias
- Representation Bias
- Measurement Bias
y actual labels
predictor
f̂
Optimization algorithm
decrease error
Quality metric
encode error of predictions

Linear Regression

ML Pipeline


Linear Regression
- Deployment Bias
Training Data
Pre-Processing
x
ML model
ŷ
- Historical Bias
- Representation Bias
- Measurement Bias
y
f
Optimization algorithm
Quality metric

Linear Regression Model

Linear: $y = mx + b$ (slope, intercept)

Assume the data is produced by a line.

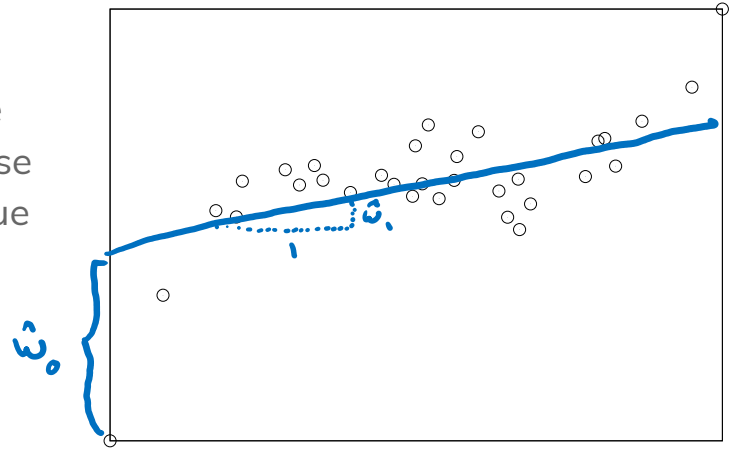
$$y_i = w_0 + w_1 x_i + \epsilon_i$$

Assuming $f(x_i)$ is a linear function $f(x_i) = w_0 + w_1 x_i$

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

- w_0 is the **intercept** / **bias** ($ of the land with no house)
- w_1 is the **slope** / **weight** ($ increase per increase in sq. ft)

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



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

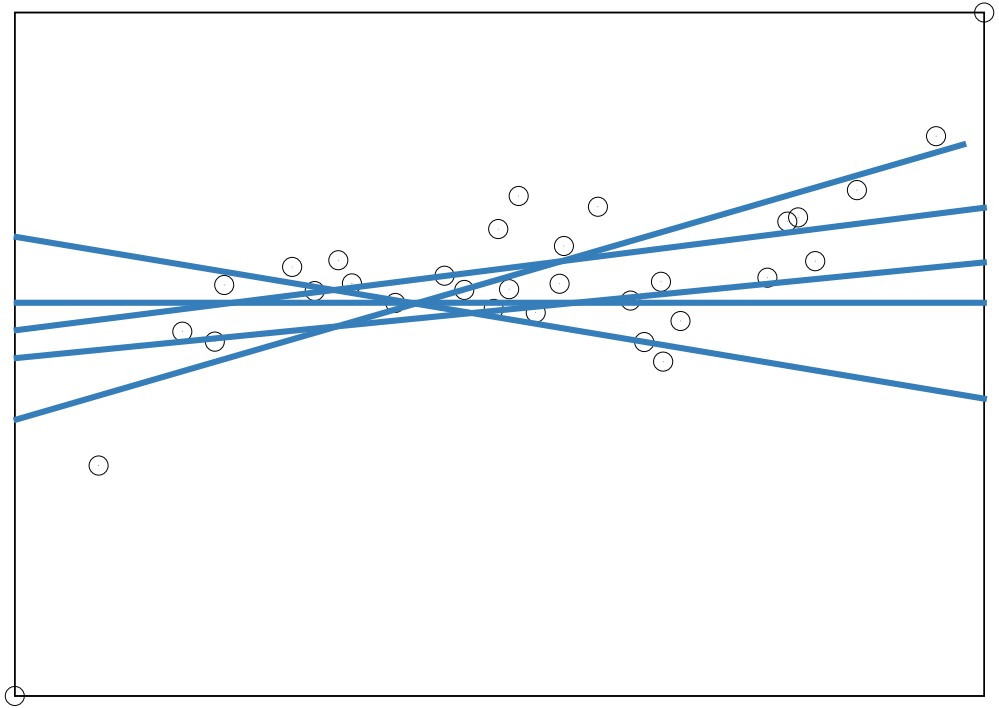
$\hat{f}(x)$

Why don't we add ϵ?

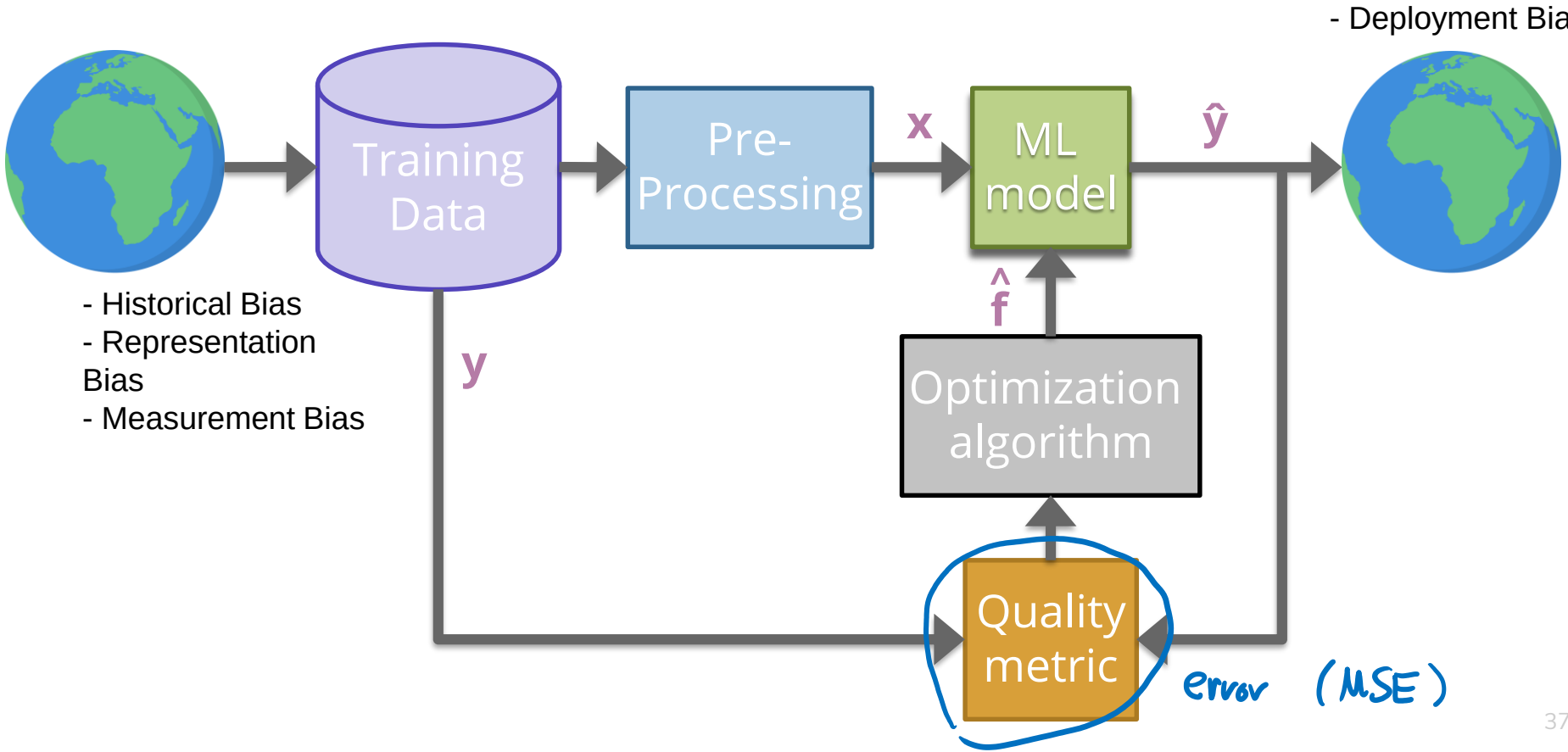
Basic Idea

Try a bunch of different lines and see which one is best!

What does best even mean here?



ML Pipeline


- Deployment Bias
Training Data
Pre-Processing
x
ML model
ŷ
- Historical Bias
- Representation Bias
- Measurement Bias
y
f̂
Optimization algorithm
Quality metric
error (MSE)

“Cost” of predictor

Define a “cost” for a particular setting of parameters

- Low cost → Better fit
- Find settings that minimize the cost
- For regression, we will use the error as the cost.
  - Low error = Low cost = **Better predictor (hopefully)**

Note: There are other ways we can define cost which will result in different “best” predictors. We will see what these other costs are and how they affect the result.

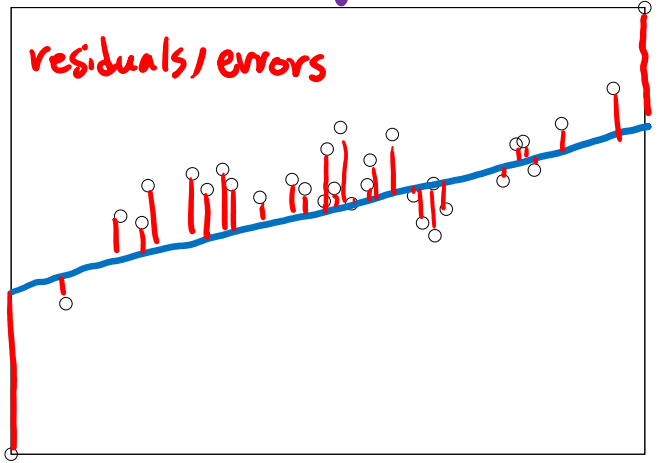
Mean Squared Error (MSE)

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

$$MSE(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i))^2$$

Average of squared errors



slido

sli.do #cs416

Didn't do in class

- **Goal**: Get you actively participating in your learning
- Typical Activity
  - Question is posed
  - **Think** (1 min): Think about the question on your own
  - **Pair** (2 min): Talk with your neighbor to discuss question
    - If you arrive at different conclusions, discuss your logic and figure out why you differ!
    - If you arrived at the same conclusion, discuss why the other answers might be wrong!
  - **Share** (1 min): We discuss the conclusions as a class
- During each of the **Think** and **Pair** stages, you will respond to the question via a sli.do poll
  - Not worth any points, just here to help you learn!

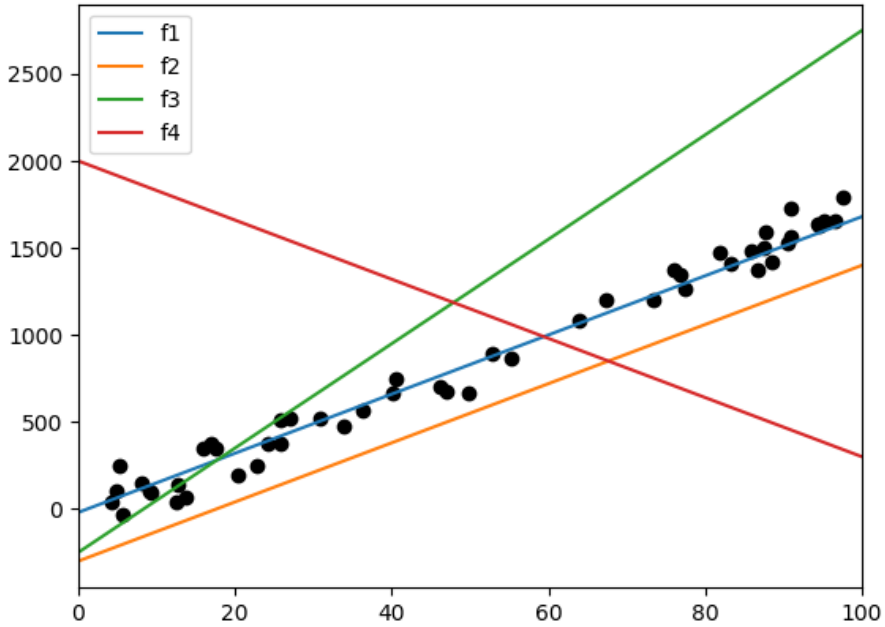
slido

Think

1 min

sli.do #cs416

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



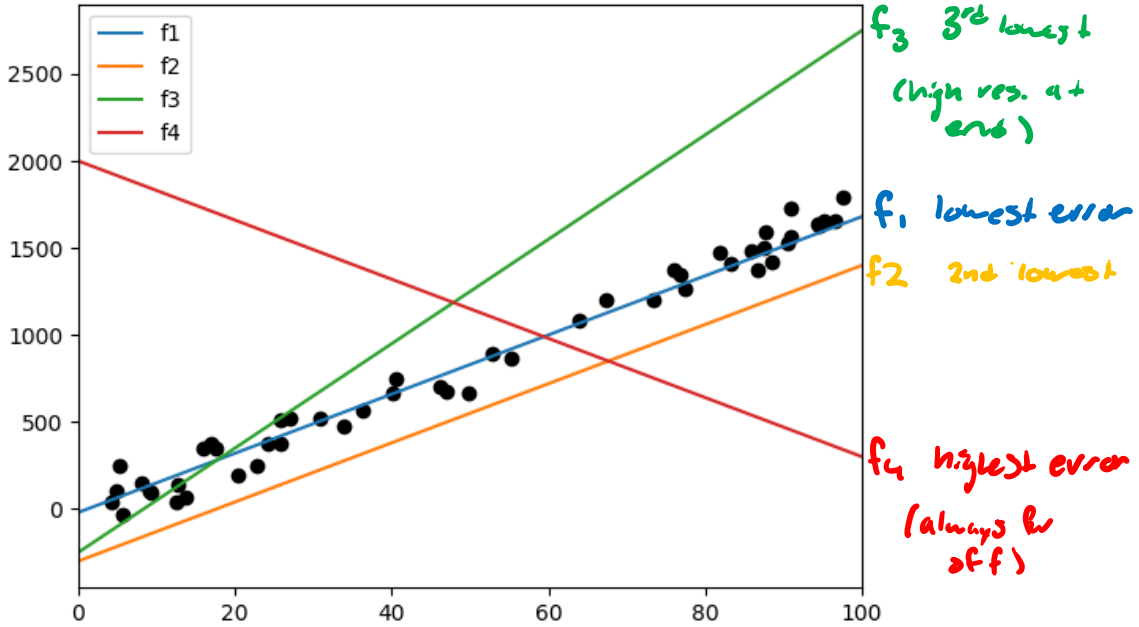
slido

Group

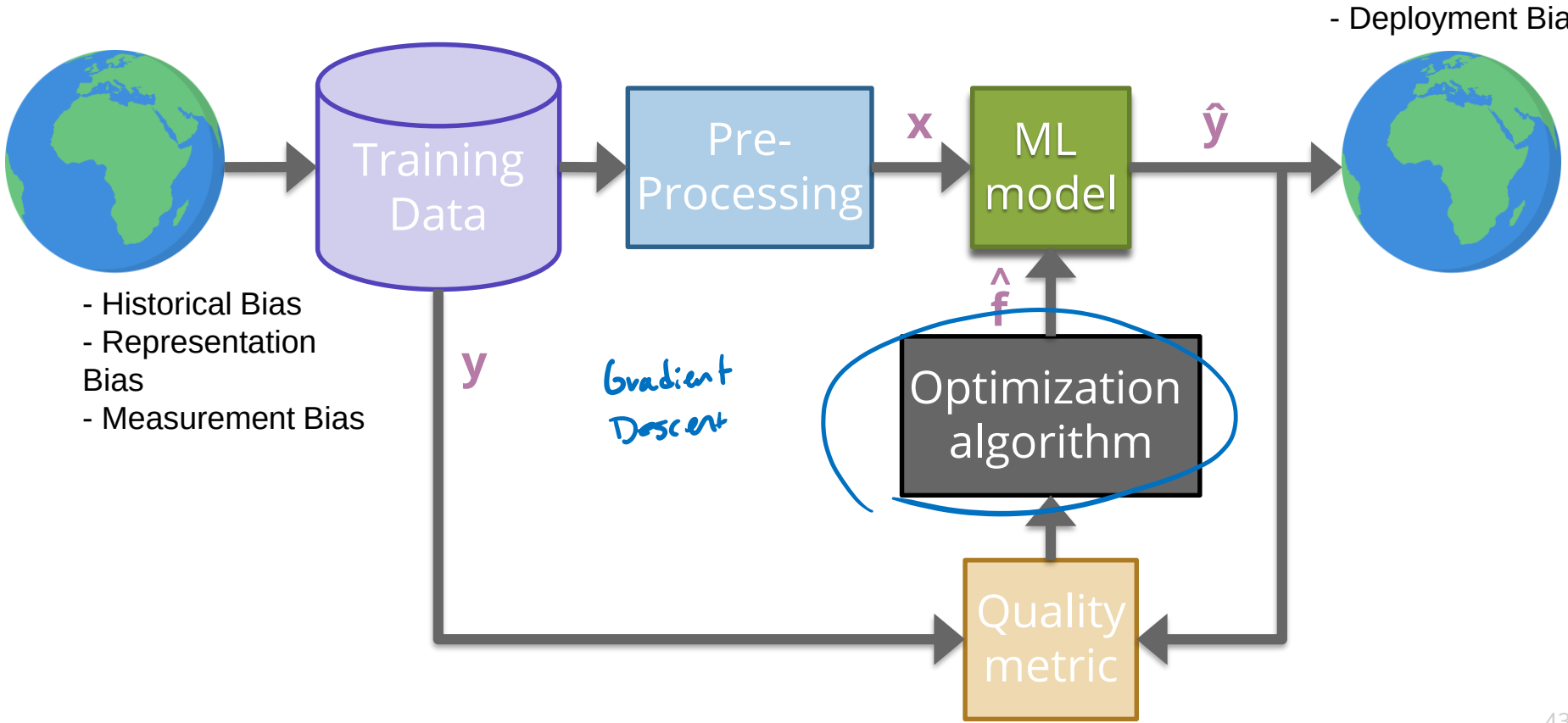
2 min

sli.do #cs416

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

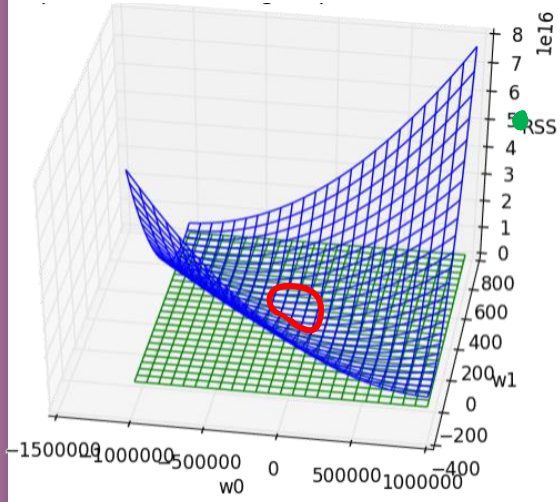


ML Pipeline


- Deployment Bias
Training Data
Pre-Processing
x
ML model
ŷ
- Historical Bias
- Representation Bias
- Measurement Bias
y
Gradient Descent
f̂
Optimization algorithm
Quality metric

Minimizing Cost

MSE is a function with inputs w_0, w_1, different settings have different MSE for a dataset



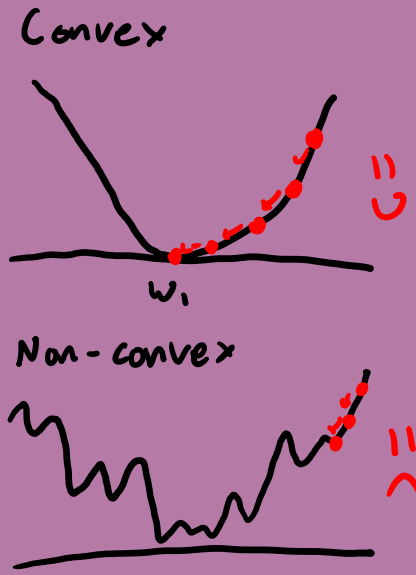
Find the settings of w_0, w_1 that minimize MSE

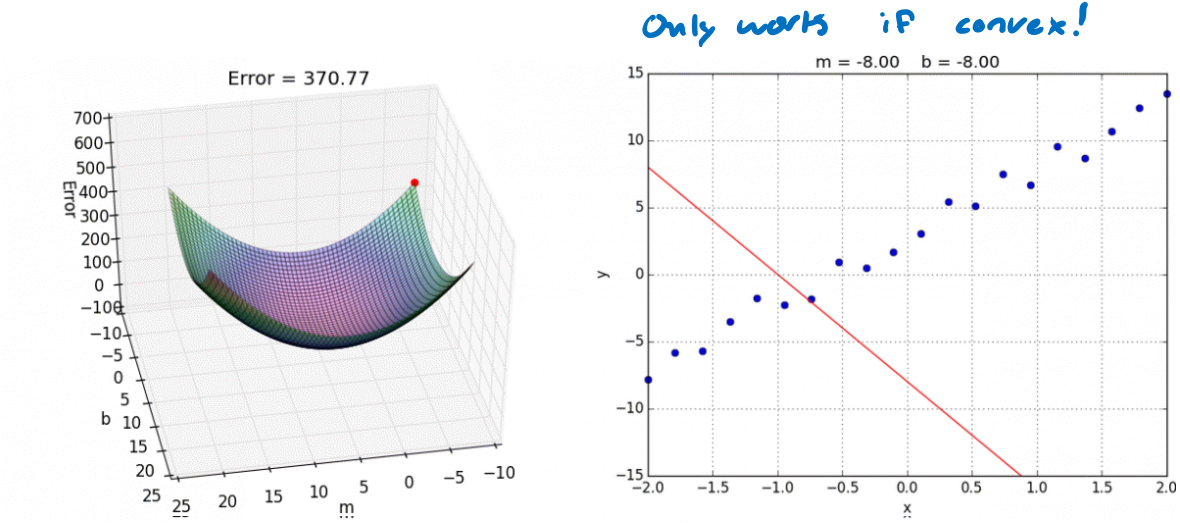
$$\hat{w}_0, \hat{w}_1 = \underset{w_0, w_1}{\operatorname{argmin}} \, MSE(w_0, w_1)$$

$$= \underset{w_0, w_1}{\operatorname{argmin}} \, \frac{1}{n} \sum_{i=1}^{n} \left(y_i - (w_0 + w_1 x_i)\right)^2$$

Unfortunately, we can't try it out on all possible settings ☹

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:

$$w^{(t+1)} = w^{(t)} - \alpha \nabla L(w)$$

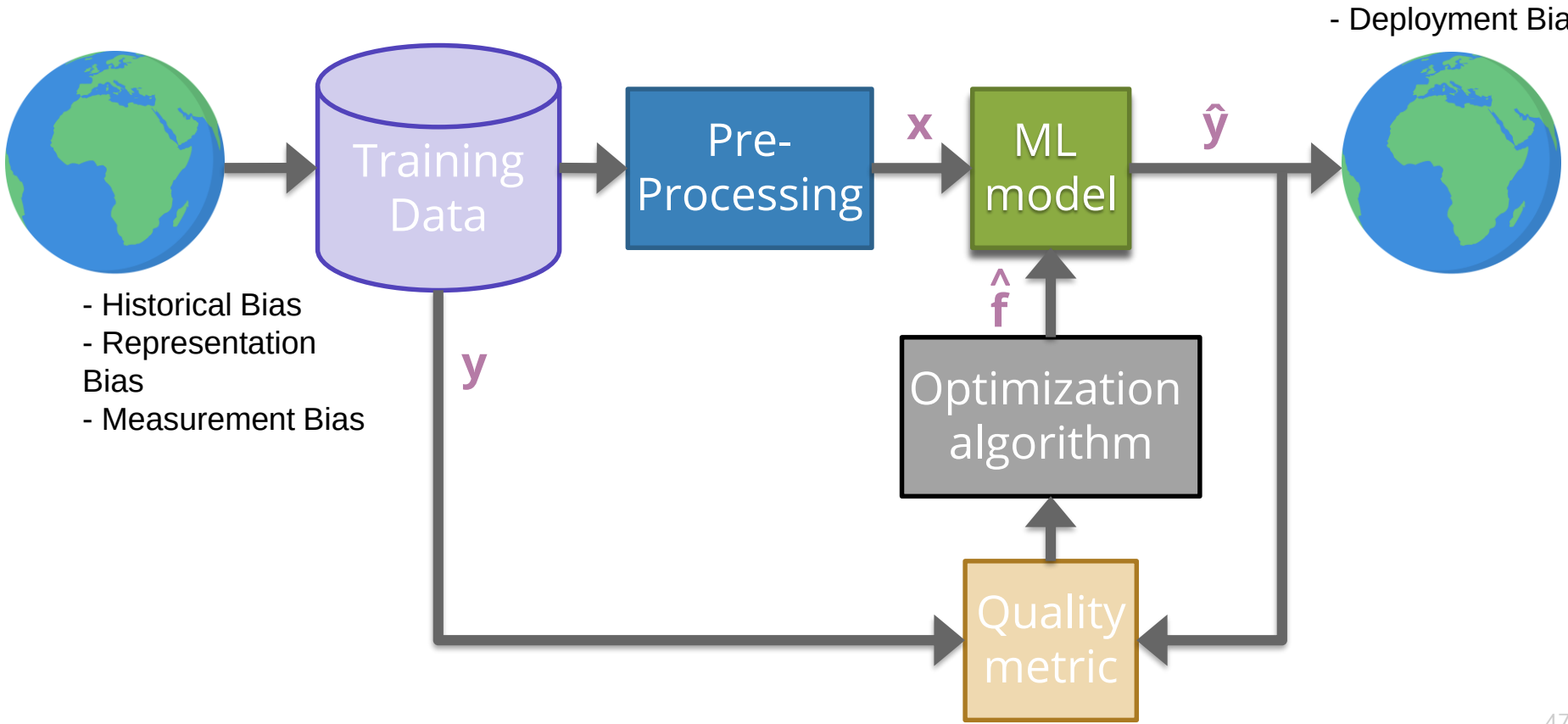
- α: learning rate
- $\nabla L(w)$: the gradients of loss function L on a set of weights w

gradient = fancy word for "slope"

Brain Break

Stopped here!

ML Pipeline

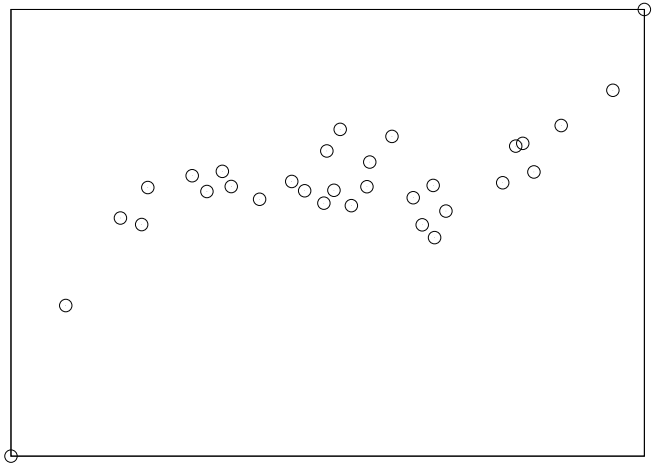

- Deployment Bias
Training Data
Pre-Processing
x
ML model
ŷ
- Historical Bias
- Representation Bias
- Measurement Bias
y
f̂
Optimization algorithm
Quality metric

Higher Order Features

This data doesn't look exactly linear, why are we fitting a line instead of some higher-degree polynomial?

We can! We just have to use a slightly different model!

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3 + \epsilon_i$$



Polynomial Regression

Model

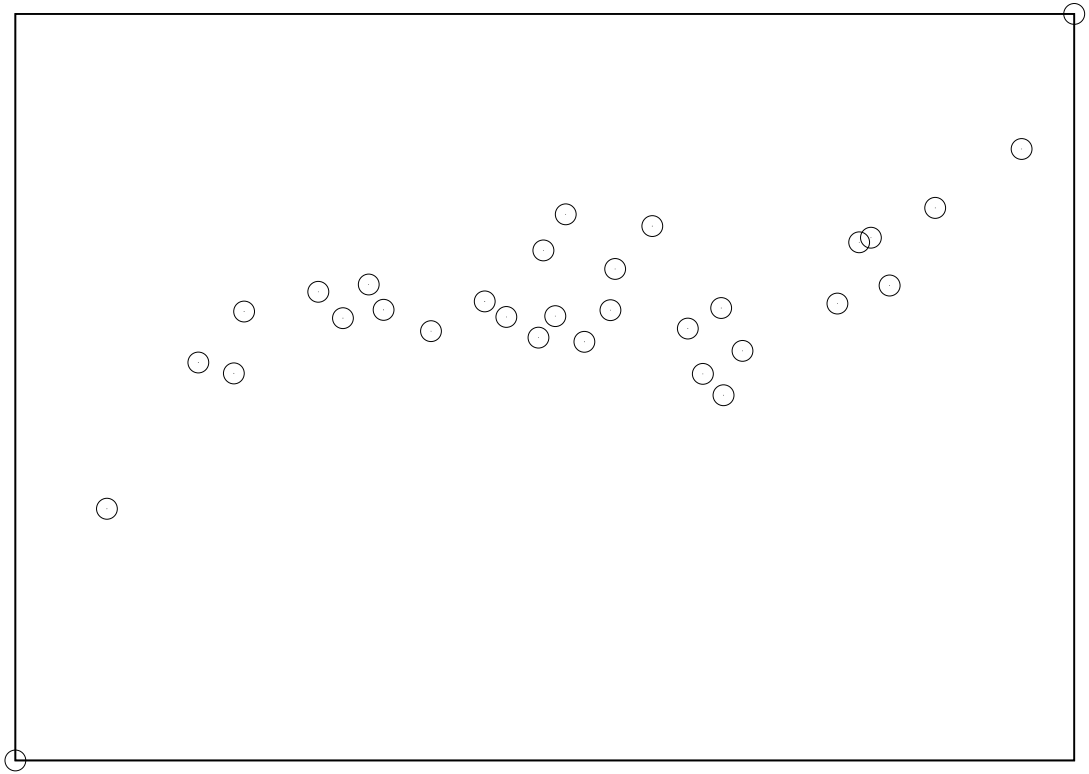
$$y_i = w_0 + w_1 x_i + w_2 x_i + \ldots + w_p x_i^p + \epsilon_i$$

Just like linear regression, but uses more features!

Feature	Value	Parameter
0	1 (constant)	w_0
1	x	w_1
2	x^2	w_2
…	…	…
p	x^p	w_p

How do you train it? Gradient descent (with more parameters)

Polynomial Regression



How to decide what the right degree? Come back Wednesday!

Features

Features are the values we select or compute from the data inputs to put into our model. **Feature extraction** is the process of turning the data into features.

Model

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \epsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(x_i) + \epsilon_i$$

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

Adding Other Inputs

Generally we are given a data table of values we might look at that include more than one value per house.

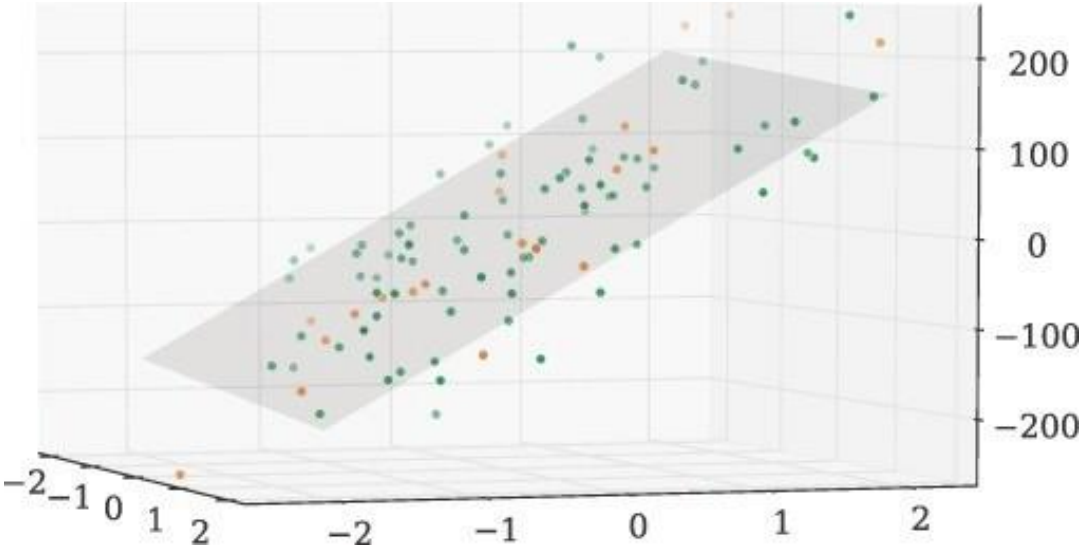
- Each row is a single house.
- Each column (except Value) is a data input.

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

More Inputs - Visually

Adding more features to the model allows for more complex relationships to be learned

$$y_i = w_0 + w_1(sq.ft.) + w_2(\#\ bathrooms) + \epsilon_i$$



Coefficients tell us the rate of change **if all other features are constant**

Notation

Important: Distinction is the difference between a *data input* and a *feature.*

- Data inputs are columns of the raw data
- Features are the values (possibly transformed) for the model (done after our feature extraction $h(x)$)

Data Input: $x_i = (x_i[1], x_i[2], \ldots, x_i[d])$

Output: y_i

- x_i is the i^{th} row
- $x_i[j]$ is the i^{th} row's j^{th} data input
- $h_j(x_i)$ is the j^{th} feature of the i^{th} row

This makes explicit, an often-implicit modeling choice of which features to use and how to transform them.

Features

You can use anything you want as features and include as many of them as you want!

Generally, more features means a more complex model. This might not always be a good thing!

Choosing good features is a bit of an art.

Feature	Value	Parameter
0	1 (constant)	w_0
1	$h_1(x)$ … $x[1]$ = sq. ft.	w_1
2	$h_2(x)$ … $x[2]$ = # bath	w_2
…	…	…
D	$h_D(x)$ … like $\log(x[7]) * x[2]$	w_D

Term recap

- **Supervised learning**: The machine learning task of learning a function that maps an input to an output based on example input-output pairs.
- **Regression**: A supervised learning task where the outputs are continuous values.
- **Feature**:
  - An attribute that we're selecting for our model
  - Can come from the original dataset, or through some transformations (***feature extraction***)
- **Parameter**: The weight or bias associated with a feature. The goal of machine learning is to adjust the weights to optimize the loss functions on training data.
- **Loss function**: A function that computes the distance between the predicted output from a machine learning model and the actual output.
- **Machine learning model**: An algorithm that combs through an amount of data to find patterns, make predictions, or generate insights
- **Optimization algorithm**: An algorithm used to minimize the loss during training. The most common one is ***Gradient Descent***.

Linear Regression Recap

Dataset

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

Feature Extraction

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

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

Regression Model

$$y = f(x) + \epsilon$$
$$= \sum_{j=0}^{D} w_j h_j(x) + \epsilon$$
$$= w^T h(x) + \epsilon$$

Quality Metric

$$MSE(w) = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2$$

Predictor

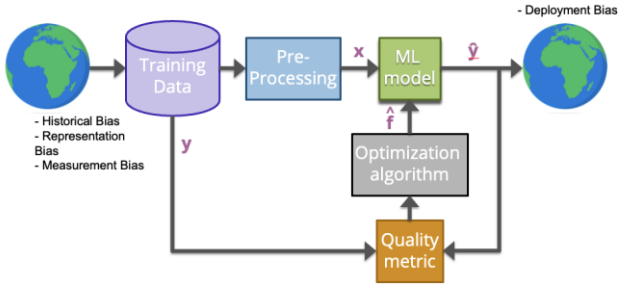
$$\hat{w} = \underset{w}{\operatorname{argmin}}\, MSE(w)$$

ML Algorithm

Optimized using Gradient Descent

Prediction

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



This Week

- **Complete Pre-Course Training** as soon as you can
- Before Wednesday
  - **Checkpoint 0** due before next class (EdStem)
  - **Pre-Class 1** should be watched before class
- Coming up soon
  - First Quiz Section on Thursday
  - Learning Reflection 0: **Due Friday 11:59 pm**
  - HW0 Released on Wednesday
- Please check EdStem regularly for the latest updates on course logistics.