

CS-GY 6923: Lecture 1

Introduction to Machine Learning

NYU Tandon School of Engineering, Prof. Christopher Musco

ARTIFICIAL INTELLIGENCE IS HAVING A MOMENT

Who has tried ChatGPT? DALLE? Imagen?



Edit the detailed description

Surprise me

Upload



Paint Washington Square Park in a snow storm in the style of Vincent Van Gogh.

Generate

ARTIFICIAL INTELLIGENCE IS HAVING A MOMENT

Who hasn't tried ChatGPT? Github Copilot? DALLE? Midjourney?



Can you write a bash script to resize all images in a directory to be less than 100kb?



Yes, here is an example bash script that uses the ``mogrify`` command from the ``imagemagick`` package to resize all images in a directory to be less than 100KB:

```
#!/bin/bash

# specify the directory containing the images
dir="path/to/images"

# resize all images in the directory
mogrify -path $dir -resize x800 -quality 50 -define jpeg:extent=100kb
-format jpg $dir/*
```

 Copy code

This script takes a directory specified by the variable ``dir`` and uses the ``mogrify`` command to resize all images in the directory with the following options:

ARTIFICIAL INTELLIGENCE IS HAVING A MOMENT

May look back on 2015-now as a the dawn of true artificial intelligence.

Times Topics: Artificial Intelligence

News about artificial intelligence, including commentary and archival articles published in The New York Times.

4h ago

BUSINESS

A.I. Isn't Magic, but Can It Be 'Agentic'?

So-called agentic artificial intelligence doesn't require a human to tell it what to do. At least, that's the idea.

By Erin Griffith

PRINT EDITION

September 8, 2024



Aug. 30

TECHNOLOGY

How Do You Change a Chatbot's Mind?

When I set out to improve my tainted reputation with chatbots, I discovered a new world of A.I. manipulation.

By Kevin Roose

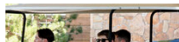
PRINT EDITION How to Change a Chatbot's Mind | September 6, 2024, Page B1



Sept. 3

TECHNOLOGY

OpenAI, Still Haunted by Its Chaotic Past, Is



Other developments in recent years:

- Human-level image classification and understanding.
- Near perfect machine translation.
- Human level game play in very complex games (Go, Starcraft).
- Machine learning as a central tool in science.

What other technologies have caught people's eye?

Give you a foundation to understand the main ideas in modern machine learning.

GOAL OF THIS CLASS

We will do so through a combination of:

- Hands on implementation.
 - Demos and take-home labs using **Python** and **Jupyter notebooks**. 20% of grade
 - We will use **Google Colab** as the primary programming environment.
- Theoretical exploration.
 - Written problem sets. 20%
 - Midterm and final exam. 25% of grade each.

Goals of theoretical component:

1. Build experience with the most important mathematical tools used in machine learning, including probability, statistics, and linear algebra. This experience will prepare you for more advanced coursework in ML, or research.
2. Be able to understand contemporary research in machine learning, including papers from NeurIPS, ICML, ICLR, and other major machine learning venues.
3. Learn how theoretical analysis can help explain the performance of machine learning algorithms and lead to the design of entirely new methods.

Goals of hands-on component:

1. Reinforce theory learned in class, and make sure you understand algorithms described by implementing them.
2. Learn how to view and formulate real world problems in the language of machine learning.
3. Gain experience applying the most popular and successful machine learning algorithms to these problems.

MORE ADVANCED/FOCUSED CLASSES AT TANDON

- CS-GY 6953: **Deep Learning** (Prof. Chinmay Hegde)
- CS-GY 6943: **Artificial Intelligence for Games** (Prof. Julian Togelius)
- ECE-GY 9163: **Machine Learning for Cybersecurity** (Prof. Siddharth Garg)
- ECE-GY 7143: **Advanced Machine Learning** (Prof. Anna Chromanska)
- CS-GY 6763: **Algorithmic Machine Learning and Data Science** (me)
- Keep your eyes out for special topics courses.

All class information can be found at:

`www.chrismusco.com/machinelearning2024\_grad`

Make sure you will be around for midterm (10/18) and final (12/20).

TWO MOST IMPORTANT THINGS FROM SYLLABUS

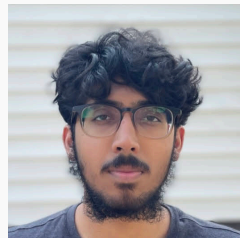
1. Make sure you are signed into and follow **Ed discussion**, which will be used for all classroom communication (no email). Now integrated into Brightspace.
2. Don't hesitate to ask me or the TAs for help.¹



Prajwal Bhattarai



Marc Chiu



Usaid Malik

¹Fill out office hours poll on Ed!

Collaboration: Students may discuss problem set problems and coding assignments. However, you must write their solutions independently.

We have a zero-tolerance policy for copied solutions. Do not let other students copy off your work or risk a zero on the assignment.

AI tools: You can use AI tools like ChatGPT as you wish, just make sure they do not impede your learning.

Class participation accounts for 10% of your grade. It's easy to get a perfect score:

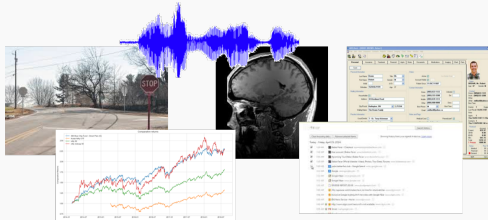
- Ask and answer questions in lecture.
- Post questions or responses to other students on Ed. Or other things you find interesting.
- Participate in professor or TA office hours.

THE PREDICTION PROBLEM

BASIC GOAL

Goal: Develop algorithms to make predictions based on data.

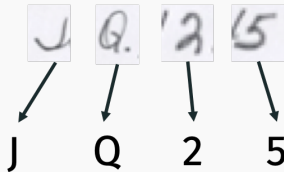
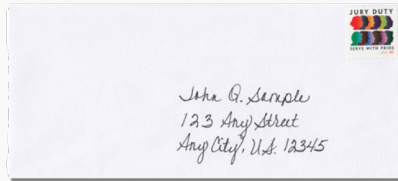
- **Input:** A single piece of data (an image, audio file, patient healthcare record, MRI scan, beginning of a sentence).



- **Output:** A prediction (this image is a stop sign, this stock will go up 10% next quarter, this song is in French, the next word in the sentence is “tomorrow”).

CLASSIC EXAMPLE

Optical character recognition (OCR): Decide if a handwritten character is an $a, b, \dots, z, 0, 1, \dots, 9, \dots$



Optical character recognition (OCR): Decide if a handwritten character is an $a, b, \dots, z, 0, 1, \dots, 9, \dots$

Applications:

- Automatic mail sorting.
- Text search in handwritten documents.
- Digitizing scanned books.
- License plate detection for tolls.
- Precursor to translation from images.

How would you write an **code** to distinguish these digits?



Suppose you just want to distinguish between a 1 and a 7.

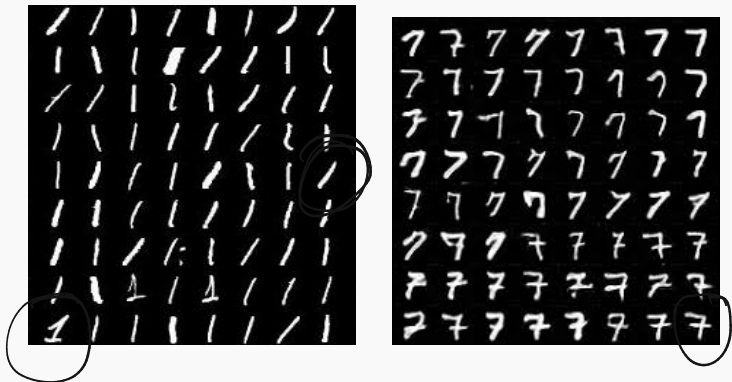
1S VS. 7S ALGORITHM

Reasonable approach: A number which contains one vertical line is a 1, if it contains one vertical and one horizontal line, it's a 7.

```
1  def count_vert_lines(image):
2  ...
3
4  def count_horiz_lines(image):
5  ...
6
7  def classify(image):
8  ...
9      nv = count_vert_lines(image)
10     nh = count_vert_lines(image)
11
12     if (nv == 1) and (nh == 1):
13         return '7'
14     elif (nv == 1) and (nh == 0):
15         return '1'
16     elif ...
```

1S VS. 7S ALGORITHM

This rule breaks down in practice:



Even fixes/modifications of the rule tend to be brittle... Maybe you could get 80% accuracy, but not nearly good enough.

CHALLENGE OF EXPERT SYSTEMS

Rule based systems, also called Expert Systems were the dominant approach to artificial intelligence in the 1970s and 1980s. Still widely used (e.g., in Amazon Alexa and similar systems).

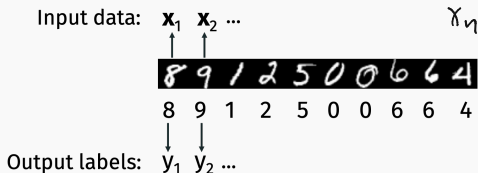
Major limitation: While human's are very good at many tasks,

- It's often hard to encode why humans make decisions in simple programmable logic.
- We think in abstract concepts with no mathematical definitions (how exactly do you define a line? how do you define a curve? straight line?)

A DIFFERENT APPROACH: SUPERVISED MACHINE LEARNING

Focus on what humans do well: solving the task at hand!

Step 1: Collect and label many input/output pairs $(\mathbf{x}_i, y_i)$. For our digit images, we have each $\mathbf{x}_i \in \mathbb{R}^{28 \times 28}$ and $y_i \in \{0, 1, \dots, 9\}$.



This is called the **training dataset**.

Step 2: Learn from the examples we have.

- Have the computer automatically find some function $f(\mathbf{x})$ such that $f(\mathbf{x}_i) = y_i$ for most $(\mathbf{x}_i, y_i)$ in our training data set (by searching over many possible functions).

Think of f as any crazy equation, or an arbitrary program:

$$f(\mathbf{x}) = 10 \cdot x[1, 1] - 6 \cdot x[3, 45] \cdot x[9, 99] + 5 \cdot \text{mean}(\mathbf{x}) + \dots$$

This approach of learning a function from labeled data is called **supervised learning**.

SUPERVISED LEARNING FOR OCR

National Institute for Standards and Technology collected a huge amount of handwritten digit data from census workers and high school students in the early 90s:

NAME		DATE	CITY	STATE	ZIP
		8-5-89	CHICAGO, ILL	IL	60652

The purpose of handwriting is being collected for use in various computer recognition and hand-printed analysis, and scoring. Please print the following characters in the boxes that appear below.

0123456789		0123456789		0123456789	
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z		a b c d e f g h i j k l m n o p q r s t u v w x y z		0 1 2 3 4 5 6 7 8 9	
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0	1	2
3	4	5	6	7	8
9	0	1	2	3	4
5	6	7	8	9	0
1	2	3	4	5	6
7	8	9	0		

This is called the NIST dataset, and was used to create the famous **MNIST handwritten digit dataset**.

Since the 1990s machine learning have overtaken expert systems as the dominant approach to artificial intelligence.

- Current methods achieve .17% error rate for OCR on benchmark datasets (MNIST).²
- Very successful on other problems as well. The big break through for supervised learning in the 2010s was image classification.

²Not because of overfitting! See: *Cold Case: The Lost MNIST Digits* by Chhavi Yadav + Léon Bottou.

Once we have the basic supervised machine learning setup, many very difficult questions remain:

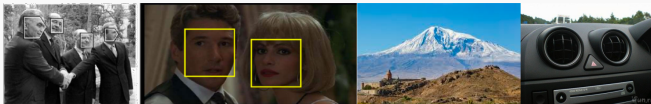
- How do we **parameterize** a class of functions f to search?
- How do we **efficiently find** a good function in the class?
- How do we ensure that an $f(\mathbf{x})$ which works well on our training data will **generalize** to perform well on future data?
- How do we deal with **imperfect data** (noise, outliers, incorrect training labels)?

Recall that in the **supervised learning** setup every input $\mathbf{x}_i$ in our training dataset comes with a desired output y_i (typically generated by a human, or some other process).

Types of supervised learning:

- **Classification** – predict a discrete class label.
- **Regression** – predict a continuous value.
 - Dependent variable, response variable, target variable, lots of different names for y_i .

Another example of supervised classification: **Face Detection**.



Each input data example x_i is an image. Each output y_i is 1 if the image contains a face, 0 otherwise.

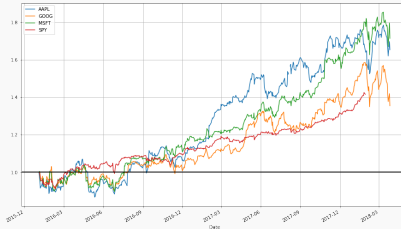
- Harder than digit recognition, but we now have essentially perfect methods (used in nearly all digital cameras, phones, etc.)

Other examples of supervised classification:

- Object detection (Input: image, Output: dog or cat)
- Spam detection (Input: email text, Output: spam or not)
- Medical diagnosis (Input: patient data, Output: disease condition or not)
- Credit decision making (Input: financial data, Output: offer loan or not)

SUPERVISED LEARNING

Example of supervised regression: **Stock Price Prediction.**



Each input $\mathbf{x}$ is a vector of metrics about a company (sales volume, PE ratio, earning reports, historical price data).

Each output y_i is the **price of the stock** 3 months in the future.

Other examples of supervised regression:

- Home price prediction (Inputs: square footage, zip code, number of bathrooms, Output: Price)
- Car price prediction (Inputs: make, model, year, miles driven, Output: Price)
- Weather prediction (Inputs: weather data at nearby stations, Output: tomorrows temperature)
- Robotics/Control (Inputs: information about environment and current position at time t , Output: estimate of position at time $t + 1$)

OTHER TYPES OF LEARNING

Later in the class we will talk about other frameworks:

- **Unsupervised learning** (no labels or response variable)
 - Important for representation learning and generative ML.
- **Self-supervised learning.**
 - Taking over the world. What Language Models like the GPT models are based on.

Focus less in this class on:

- **Reinforcement learning**
 - Game playing.
- **Active-learning.**
 - The learning algorithms can request labels.

Types of supervised learning:

- **Classification** – predict a discrete class label.
- **Regression** – predict a continuous value.
 - Dependent variable, response variable, target variable, lots of different names for y_i .

Motivating example: Predict the highway miles per gallon (MPG) of a car given quantitative information about its engine.
Demo in `demo_auto_mpg.ipynb`.

What factors might matter?

Data set available from the UCI Machine Learning Repository:
<https://archive.ics.uci.edu/>.



Machine Learning Repository
 Center for Machine Learning and Intelligent Systems

[About](#) [Citation Policy](#) [Donate a Data Set](#) [Contact](#)

[Repository](#) [Web](#)

[View ALL Data Sets](#)

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 485 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#).

Supported By:  In Collaboration With: 

Latest News:	Newest Data Sets:	Most Popular Data Sets (Hits since 2007):
09-24-2018: Welcome to the new Repository admin! Dheeru Dua and El Karra Tenenbaum! 04-04-2013: Welcome to the new Repository admin! Kevin Bache and Moshe Lohman! 03-01-2010: Data from donor regarding Netflix data 10-15-2009: Two new data sets have been added. 06-14-2009: Several data sets have been added. 03-24-2009: New data sets have been added! 06-25-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope	10-06-2019:  WISDM Smartphone and Smartwatch Activity and Biometrics Dataset 09-30-2019:  Hepatitis C Virus (HCV) for Egyptian patients 09-23-2019:  QSAR fish toxicity 09-23-2019:  QSAR aquatic toxicity 09-21-2019:  Online Retail II 09-20-2019:  Human Activity Recognition from Continuous Ambient Sensor Data 09-20-2019:  Beijing Multi-Site Air-Quality Data 09-20-2019:  MEIS 09-20-2019:  PPG-DaLiA 07-24-2019:  Ovarian Predictors data set 07-22-2019:  Alcohol QCM Sensor Dataset 07-14-2019:  Incident management process enriched event log	3099401:  Iris 1711996:  Adult 1328924:  Wine 1126497:  Heart Disease 1126386:  Wine Quality 1116408:  Car Evaluation 1116558:  Breast Cancer Wisconsin (Diagnostic) 1101179:  Bank Marketing 935256:  Human Activity Recognition Using Smartphones 885144:  Abalone 839187:  Forest Fires 586581:  Poker Hand

Featured Data Set: Ozone Level Detection



Task: Classification
Data Type: Multivariate, Sequential, Time-Series
Attributes: 73
Instances: 2536

Two ground ozone level data sets are included in this collection. One is the eight hour peak set (eighthr data), the other is the one hour peak set (onehr data). Those data were collected from 1980 to 2004 at the Houston, Galveston and Brazoria area.

PREDICTING MPG

Datasets from UCI (and many other places) comes as tab, space, or comma delimited files.

housing.data

auto-mpg.data ×

Users > christophermusco > Desktop > auto-mpg.data

1	18.0	8	307.0	130.0	3504.	12.0	70	1	"chevrolet chevelle malibu"
2	15.0	8	350.0	165.0	3693.	11.5	70	1	"buick skylark 320"
3	18.0	8	318.0	150.0	3436.	11.0	70	1	"plymouth satellite"
4	16.0	8	304.0	150.0	3433.	12.0	70	1	"amc rebel sst"
5	17.0	8	302.0	140.0	3449.	10.5	70	1	"ford torino"
6	15.0	8	429.0	198.0	4341.	10.0	70	1	"ford galaxie 500"
7	14.0	8	454.0	220.0	4354.	9.0	70	1	"chevrolet impala"
8	14.0	8	440.0	215.0	4312.	8.5	70	1	"plymouth fury ii"
9	14.0	8	455.0	225.0	4425.	10.0	70	1	"pontiac catalina"
10	15.0	8	390.0	190.0	3850.	8.5	70	1	"amc ambassador dpl"
11	15.0	8	383.0	170.0	3563.	10.0	70	1	"dodge challenger se"
12	14.0	8	340.0	160.0	3609.	8.0	70	1	"plymouth 'cuda 340"
13	15.0	8	400.0	150.0	3761.	9.5	70	1	"chevrolet monte carlo"
14	14.0	8	455.0	225.0	3086.	10.0	70	1	"buick estate wagon (sw)"
15	24.0	4	113.0	95.00	2372.	15.0	70	3	"toyota corona mark ii"
16	22.0	6	198.0	95.00	2833.	15.5	70	1	"plymouth duster"
17	18.0	6	199.0	97.00	2774.	15.5	70	1	"amc hornet"
18	21.0	6	200.0	85.00	2587.	16.0	70	1	"ford maverick"
19	27.0	4	97.00	88.00	2130.	14.5	70	3	"datsun pl510"
20	26.0	4	97.00	46.00	1835.	20.5	70	2	"volkswagen 1131 deluxe sedan"
21	25.0	4	110.0	87.00	2672.	17.5	70	2	"peugeot 504"
22	24.0	4	107.0	90.00	2430.	14.5	70	2	"audi 100 ls"
23	25.0	4	104.0	95.00	2375.	17.5	70	2	"saab 99e"
24	26.0	4	121.0	113.0	2234.	12.5	70	2	"bmw 2002"
25	21.0	6	199.0	90.00	2648.	15.0	70	1	"amc gremlin"
26	10.0	8	360.0	215.0	4615.	14.0	70	1	"ford f250"

PREDICTING MPG

Check dataset description to know what each column means.

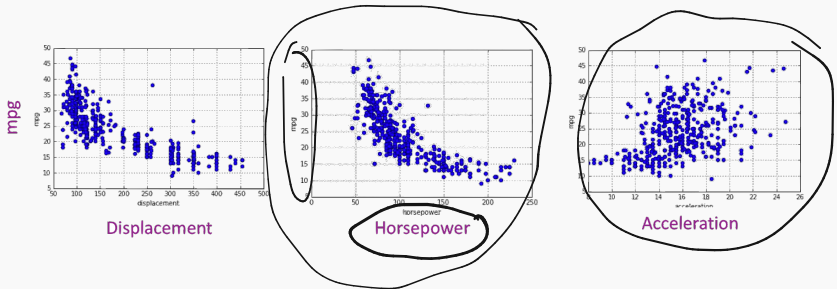
Users > christophermusco > Desktop > auto-mpg.data

y_1	1	18.0	8	307.0	130.0	3504.	12.0	70	1	"chevrolet chevelle malibu"	x_1
	2	15.0	8	350.0	165.0	3693.	11.5	70	1	"buick skylark 320"	x_2
y_2	3	18.0	8	318.0	150.0	3436.	11.0	70	1	"plymouth satellite"	x_3
	4	16.0	8	304.0	150.0	3433.	12.0	70	1	"amc rebel sst"	
y_3	5	17.0	8	302.0	140.0	3449.	10.5	70	1	"ford torino"	
	6	15.0	8	429.0	198.0	4341.	10.0	70	1	"ford galaxie 500"	
	7	14.0	8	454.0	220.0	4354.	9.0	70	1	"chevrolet impala"	
	8	14.0	8	440.0	215.0	4312.	8.5	70	1	"plymouth fury iii"	
	9	14.0	8	455.0	225.0	4425.	10.0	70	1	"pontiac catalina"	
	10	15.0	8	390.0	190.0	3850.	8.5	70	1	"amc ambassador dpl"	
	11	15.0	8	383.0	170.0	3563.	10.0	70	1	"dodge challenger se"	
	12	14.0	8	340.0	160.0	3609.	8.0	70	1	"plymouth cuda 340"	
	13	15.0	8	400.0	150.0	3761.	9.5	70	1	"chevrolet monte carlo"	
	14	14.0	8	455.0	225.0	3086.	10.0	70	1	"buick estate wagon (sw)"	
	15	24.0	4	113.0	95.00	2372.	15.0	70	3	"toyota corona mark ii"	
	16	22.0	6	198.0	95.00	2833.	15.5	70	1	"plymouth duster"	
	17	18.0	6	199.0	97.00	2774.	15.5	70	1	"amc hornet"	
	18	21.0	6	200.0	85.00	2587.	16.0	70	1	"ford maverick"	
	19	27.0	4	97.00	88.00	2130.	14.5	70	3	"datsun pl510"	
	20	26.0	4	97.00	46.00	1835.	20.5	70	2	"volkswagen 1131 deluxe sedan"	
	21	25.0	4	110.0	87.00	2672.	17.5	70	2	"peugeot 504"	
	22	24.0	4	107.0	90.00	2430.	14.5	70	2	"audi 100 ls"	
	23	25.0	4	104.0	95.00	2375.	17.5	70	2	"saab 99e"	
	24	26.0	4	121.0	113.0	2234.	12.5	70	2	"bmw 2002"	
	25	21.0	6	199.0	90.00	2648.	15.0	70	1	"amc gremlin"	
	26	10.0	8	360.0	215.0	4615.	14.0	70	1	"ford f250"	

('mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'model year', 'origin', 'car name')

LIBRARIES FOR INITIAL DATA READING

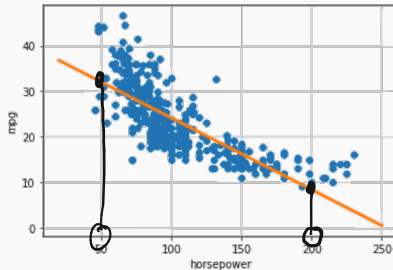
- Use **pandas** for reading data from delimited files. Stores data in a type of table called a “data frame” but this is just a wrapper around a **numpy** array.
- Use **matplotlib** for initial exploration.



SIMPLE LINEAR REGRESSION

SIMPLE LINEAR REGRESSION

Linear regression from a Machine Learning (not a Statistics) perspective. Our first supervised machine learning model.

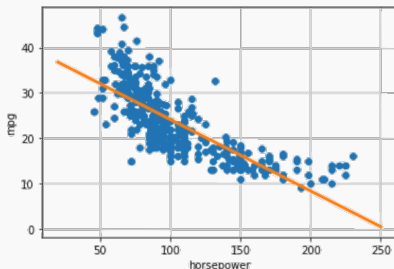


Only focus on one predictive variable at a time (e.g. horsepower). This is why it's called simple linear regression.

SIMPLE LINEAR REGRESSION

Dataset:

- $\underline{x_1}, \dots, \underline{x_n} \in \mathbb{R}$ (horsepowers of n cars – this is the predictor/independent variable)
- $\underline{y_1}, \dots, \underline{y_n} \in \mathbb{R}$ (MPG – this is the response/dependent variable)



SUPERVISED LEARNING DEFINITIONS

- **Model** $\widetilde{f_{\theta}}(x)$: Class of equations or programs which map input x to predicted output. We want $f_{\theta}(x_i) \approx y_i$ for training inputs.
- **Model Parameters** θ : Vector of numbers. These are numerical knobs which parameterize our class of models.
- **Loss Function** $L(\theta)$: Measure of how well a model fits our data. Often some function of $f_{\theta}(x_1) - y_1, \dots, f_{\theta}(x_n) - y_n$

Common Goal: Choose parameters θ^* which minimize the Loss

Function:

f_{θ^*}

$$\theta^* = \arg \min_{\theta} L(\theta)$$

$$\sum_{i=1}^n (f_{\theta}(x_i) - y_i)^2$$

Choosing θ^* based on minimizing the empirical error on our training data is called Empirical Risk Minimization. It is by far the most common approach to solving supervised learning problems.

LINEAR REGRESSION

General Supervised Learning

• Model: $f_{\theta}(x)$

• Model Parameters: $\theta = (\beta_0, \beta_1)$

$\beta_0 \rightarrow \text{intercept}$

$\beta_1 \rightarrow \text{slope}$

• Loss Function: $L(\theta)$

Linear Regression

• Model:

$$f_{\beta_0, \beta_1}(x) = \beta_0 + \beta_1 x$$

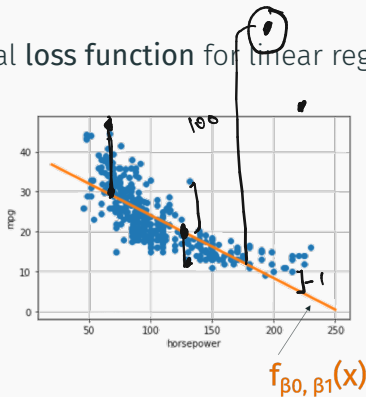
• Model Parameters:

• Loss Function:

$$\sum_{i=1}^n (f_{\beta_0, \beta_1}(x_i) - y_i)^2$$

HOW TO MEASURE GOODNESS OF FIT

What is a natural **loss function** for linear regression?

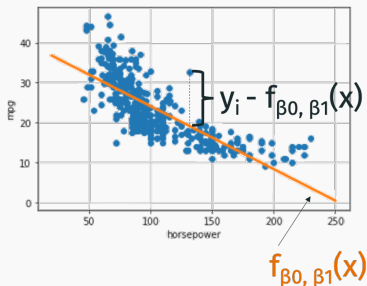


$$100^2 = 10000$$

$$1^2 = 1$$

HOW TO MEASURE GOODNESS OF FIT

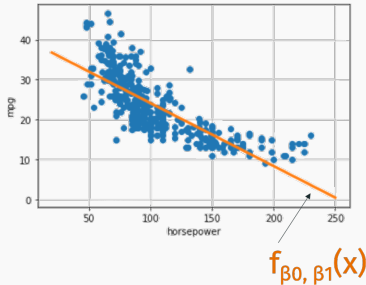
Typical choices are a function of $y_1 - f_{\beta_0, \beta_1}(x_1), \dots, y_n - f_{\beta_0, \beta_1}(x_n)$



- ℓ_2 /Squared Loss: $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - f_{\beta_0, \beta_1}(x_i))^2$.
- ℓ_1 /Least absolute deviation: $L(\beta_0, \beta_1) = \sum_{i=1}^n |y_i - f_{\beta_0, \beta_1}(x_i)|$.
- ℓ_∞ Loss $L(\beta_0, \beta_1) = \max_{i \in 1, \dots, n} |y_i - f_{\beta_0, \beta_1}(x_i)|$.

HOW TO MEASURE GOODNESS OF FIT

We're going to start with the Squared Loss/Sum-of-Squares Loss. Also called "Residual Sum-of-Squares (RSS)"



- Relatively robust to outliers.
- Simple to define, leads to simple algorithms for finding β_0, β_1
- Theoretically justified from classical statistics related to assumptions about Gaussian noise. Will discuss later in the course.

General Supervised Learning

- Model: $f_{\theta}(x)$
- Model Parameters: θ
- Loss Function: $L(\theta)$

Linear Regression

- Model:
 $f_{\beta_0, \beta_1}(x) = \underline{\beta_0 + \beta_1 \cdot x}$
- Model Parameters: β_0, β_1
- Loss Function: $L(\beta_0, \beta_1) = \sum_{i=1}^n \underline{(y_i - f_{\beta_0, \beta_1}(x_i))^2}$

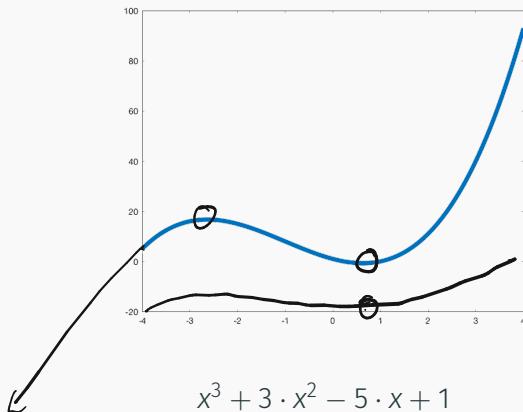
Goal: Choose β_0, β_1 to minimize

$$L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2.$$

This is the entire job of any **Supervised Learning Algorithm**.

FUNCTION MINIMIZATION

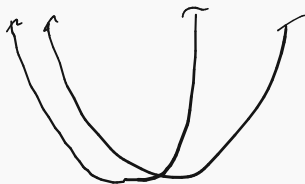
Univariate function:



- Find all places where derivative $f'(x) = 0$ and check which has the smallest value.

Multivariate function: $L(\beta_0, \beta_1)$

- Find values of β_0, β_1 where all partial derivatives equal 0.
- $\frac{\partial L}{\partial \beta_0} = 0$ and $\frac{\partial L}{\partial \beta_1} = 0$.



MINIMIZING SQUARED LOSS FOR REGRESSION

Multivariate function: $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$

- Find values of β_0, β_1 where all partial derivatives equal 0.
- $\frac{\partial L}{\partial \beta_0} = 0$ and $\frac{\partial L}{\partial \beta_1} = 0$. $x_1, \dots, x_n \quad y_1, \dots, y_n$

Some definitions:

- Let $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$. $\bar{y}$ is the mean of y .
- Let $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$. $\bar{x}$ is the mean of x .
- Let $\sigma_y^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$. σ_y^2 is the variance of y .
- Let $\sigma_x^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$. σ_x^2 is the variance of x .
- Let $\sigma_{xy} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$. σ_{xy} is the covariance.

Claim: $L(\beta_0, \beta_1)$ is minimized at:

- $\beta_1^* = \sigma_{xy} / \sigma_x^2$
- $\beta_0^* = \bar{y} - \beta_1^* \bar{x}$

PROOF

$\frac{\partial}{\partial \beta_0} L(\beta_0, \beta_1) = \sum_{i=1}^n \frac{\partial}{\partial \beta_0} (y_i - \beta_0 - \beta_1 x_i)^2$

$= \sum_{i=1}^n 2(y_i - \beta_0 - \beta_1 x_i) \cdot (-1)$

$= -2 \sum_{i=1}^n y_i + 2 \sum_{i=1}^n \beta_0 + 2 \beta_1 \sum_{i=1}^n x_i = 0$

$\Rightarrow -\frac{1}{n} \sum_{i=1}^n y_i + \beta_0 + \beta_1 \cdot \frac{1}{n} \sum_{i=1}^n x_i = 0$

$-\bar{y} + \beta_0 + \beta_1 \bar{x} = 0$

$\beta_0 = \bar{y} - \beta_1 \bar{x}$

1/24

$$\text{Loss function: } L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\frac{\partial}{\partial \beta_1} L(\beta_0, \beta_1) = \sum_{i=1}^n \frac{\partial}{\partial \beta_1} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$= \sum_{i=1}^n \frac{\partial}{\partial \beta_1} (y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i)^2$$

$$= \sum_{i=1}^n 2(y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i) \cdot (\bar{x} - x_i)$$

$$2 \cdot \left(\underbrace{\sum_{i=1}^n (y_i - \bar{y})(\bar{x} - x_i)}_{-n \cdot \bar{x} \cdot \bar{y}} + \beta_1 \underbrace{\sum_{i=1}^n (\bar{x} - x_i)(\bar{x} - x_i)}_{n \cdot \bar{x}^2} \right) = 0$$

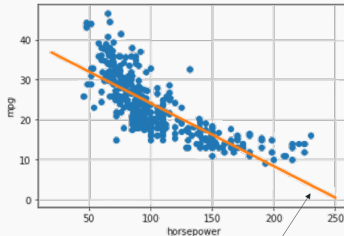
$$\textcircled{\beta_1} = \frac{n \cdot \bar{x} \cdot \bar{y}}{n \cdot \bar{x}^2} = \frac{\bar{x} \cdot \bar{y}}{\bar{x}^2}$$

Loss function: $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$

MINIMIZING SQUARED LOSS FOR REGRESSION

Takeaways:

- Minimizing functions exactly is sometimes easy with calculus, but not always! We will learn much more general tools (like gradient descent).
- Simple closed form formula for optimal parameters β_0^* and β_1^* for squared-loss!



$$B_0^* + B_1^*x$$

A FEW COMMENTS

Let $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$.

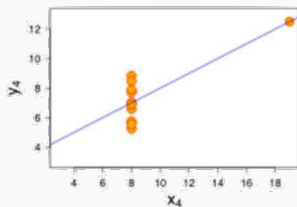
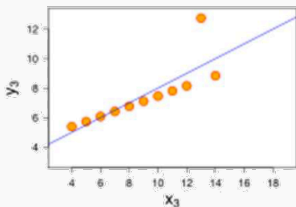
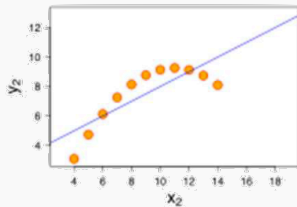
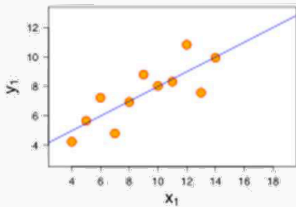
$$\underline{R^2} = 1 - \frac{L(\beta_0, \beta_1)}{n\sigma_y^2}$$

is exactly the R^2 value (“coefficient of determination”) you may remember from statistics.

The smaller the loss, the closer R^2 is to 1, which means we have a better regression fit.

A FEW COMMENTS

Many reasons you might get a poor regression fit:



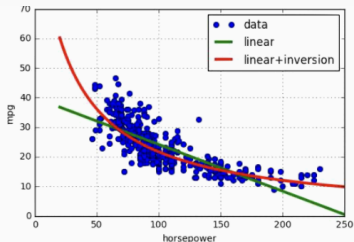
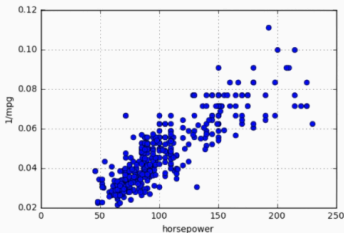
A FEW COMMENTS

Some of these are fixable!

$$1 / f_{\beta_0, \beta_1}(x_i)$$

- Remove outliers, use more robust loss function.
- **Non-linear model transformation.**

Fit the model $\frac{1}{\text{mpg}} \approx \beta_0 + \beta_1 \cdot \text{horsepower}$.



Much better fit, same exact learning algorithm!

MULTIPLE LINEAR REGRESSION

Predict target y using multiple features, simultaneously.

Motivating example: Predict diabetes progression in patients after 1 year based on health metrics. (Measured via numerical score.)

Features: Age, sex, average blood pressure, six blood serum measurements (e.g. cholesterol, lipid levels, iron, etc.)

Demo in `demo_diabetes.ipynb`.

Introducing Scikit Learn.


[Install](#)
[User Guide](#)
[API](#)
[Examples](#)
[More »](#)

scikit-learn

Machine Learning in Python

[Getting Started](#)
[What's New in 0.22.1](#)
[GitHub](#)

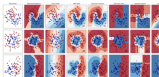
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



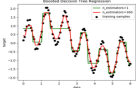
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



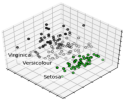
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation, metrics, and more...

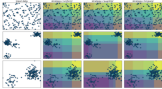


Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...





Pros:

- One of the most popular “traditional” ML libraries.
- Many built in models for regression, classification, dimensionality reduction, etc.
- Easy to use, works with ‘numpy’, ‘scipy’, other libraries we use.
- Great for rapid prototyping, testing models.

Cons:

- Everything is very “black-box”: difficult to debug, understand why models aren’t working, speed up code, etc.

Modules used:

- `datasets` module contains a number of pre-loaded datasets. Saves time over downloading and importing with `pandas`.
- `linear_model` can be used to solve Multiple Linear Regression. A bit overkill for this simple model, but gives you an idea of `sklearn`'s general structure.

THE DATA MATRIX

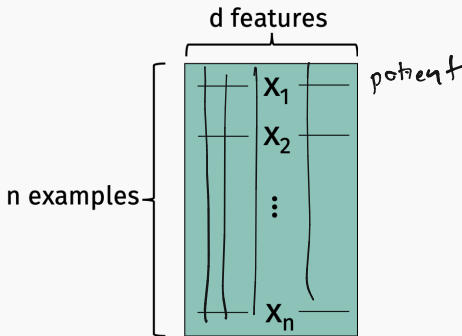
Target variable:

$\mathbb{R}$

- Scalars $y_1, \dots, y_n$ for n data examples (a.k.a. samples).

Predictor variables:

- d dimensional vectors $\underline{x}_1, \dots, \underline{x}_n$ for n data examples and d features



Now is the time to review your linear algebra!

Notation:

$$X^T_{ij} = X_{ji}$$

- Let X be an $n \times d$ matrix. Written $X \in \mathbb{R}^{n \times d}$.
- x_i is the i^{th} row of the matrix.
- $x^{(j)}$ is the j^{th} column.
- x_{ij} is the i, j entry.
- For a vector y , y_i is the i^{th} entry.
- X^T is the matrix transpose.
- y^T is a vector transpose.

Things to remember:

- Matrix multiplication. If I multiply $\underline{X} \in \mathbb{R}^{n \times d}$ by $\underline{Y} \in \mathbb{R}^{d \times k}$ I get $\underline{XY} = \underline{Z} \in \mathbb{R}^{n \times k}$.
- Inner product/dot product. $\langle \underline{y}, \underline{z} \rangle = \sum_{i=1}^n y_i z_i$.
- $\langle \underline{y}, \underline{z} \rangle = \underline{y}^T \underline{z} = \underline{z}^T \underline{y}$.
- Euclidean norm: $\|\underline{y}\|_2 = \sqrt{\underline{y}^T \underline{y}}$.
- $(\underline{XY})^T = \underline{Y}^T \underline{X}^T$.

$$\sqrt{\sum_{i=1}^n y_i^2} = \sqrt{\sum_{i=1}^n y_i \cdot y_i} = \sqrt{\underline{y}^T \underline{y}}$$

Things to remember:

- Identity matrix is denoted as $\mathbf{I}$.
- “Most” square matrices have an inverse: i.e. if $\mathbf{Z} \in \mathbb{R}^{n \times n}$, there is a matrix $\mathbf{Z}^{-1}$ such that $\mathbf{Z}^{-1}\mathbf{Z} = \mathbf{Z}\mathbf{Z}^{-1} = \mathbf{I}$.
- Let $\mathbf{D} = \text{diag}(\mathbf{d})$ be a diagonal matrix containing the entries in $\mathbf{d}$.
- $\mathbf{XD}$ scales the columns of $\mathbf{X}$. $\mathbf{DX}$ scales the rows.

You also need to be comfortable working with matrices in `numpy`. Go through the `demo_numpy_matrices.ipynb` slowly.



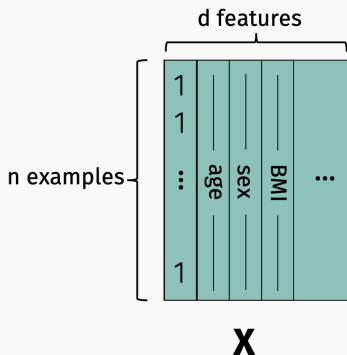
THE DATA MATRIX

Target variable:

- Scalars $y_1, \dots, y_n$ for n data examples (a.k.a. samples).

Predictor variables:

- d dimensional vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ for n data examples and d features



MULTIPLE LINEAR REGRESSION

Data matrix indexing:

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ X_{31} & X_{32} & \dots & X_{3d} \\ \vdots & \vdots & & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{bmatrix}$$

$$y_i \approx \beta_1 x_{i1} + \beta_2 x_{i2} + \dots$$

$$\beta_1, \dots, \beta_d$$

Multiple Linear Regression Model:

Predict

$$\underline{y_i} \approx \underline{\beta_1} x_{i1} + \overset{\nearrow 1}{\underline{\beta_2}} x_{i2} + \dots + \underline{\beta_d} x_{id}$$

The rate at which diabetes progresses depends on many factors, with each factor having a different magnitude effect.

MULTIPLE LINEAR REGRESSION

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ X_{31} & X_{32} & \dots & X_{3d} \\ \vdots & \vdots & & \vdots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{bmatrix} = \begin{bmatrix} 1 & X_{12} & \dots & X_{1d} \\ 1 & X_{22} & \dots & X_{2d} \\ 1 & X_{32} & \dots & X_{3d} \\ \vdots & \vdots & & \vdots \\ 1 & X_{n2} & \dots & X_{nd} \end{bmatrix}$$

$\underbrace{\hspace{10em}}_{d-1}$

Multiple Linear Regression Model:

Predict $y_i \approx \beta_1 + \beta_2 x_{i2} + \dots + \beta_d x_{id}$

In this case, β_1 serves as the “intercept” parameter.

MULTIPLE LINEAR REGRESSION

Use as much linear algebra notation as possible!

- Model Parameters:

$$\beta_1, \dots, \beta_d = \underline{\underline{\beta}}$$

- Model:

$$f_{\beta_1, \dots, \beta_d}(\vec{x}) = \beta_1 x[1] + \beta_2 x[2] \dots \beta_d x[d]$$

$\nearrow$ vector

$$= \langle \vec{\beta}, \vec{x} \rangle$$

- Loss Function:

$$\sum_{i=1}^n (y_i - \langle \vec{\beta}, \vec{x}_i \rangle)^2$$

Linear Least-Squares Regression.

- Model Parameters:

$$\beta = [\beta_1, \beta_2, \dots, \beta_d]$$

- Model:

$$f_{\beta}(x) = \langle x, \beta \rangle$$

- Loss Function:

$$\begin{matrix} d \\ \downarrow \\ \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix} \end{matrix} \quad \begin{matrix} d \\ \downarrow \\ \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \end{matrix} = \begin{matrix} n \\ \downarrow \\ \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \end{matrix}$$

$$L(\beta) = \sum_{i=1}^n |y_i - \langle x_i, \beta \rangle|^2$$

$$= \|y - X\beta\|_2^2$$

$(\beta_1, \dots, \beta_d)$

$\|y - X\beta + 1 \cdot \beta_0\|_2$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

LINEAR ALGEBRAIC FORM OF LOSS FUNCTION

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \begin{matrix} \\ \\ \\ B \end{matrix} = \begin{bmatrix} \langle x_1, \beta \rangle \\ \langle x_2, \beta \rangle \\ \vdots \\ \langle x_n, \beta \rangle \end{bmatrix}$$

$\beta_1, \dots, \beta_n$
 $\langle x, \beta \rangle + \beta_0$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \langle x_1, \beta \rangle \\ \vdots \\ \langle x_n, \beta \rangle \end{bmatrix} = \begin{bmatrix} y_1 - \langle x_1, \beta \rangle \\ \vdots \\ y_n - \langle x_n, \beta \rangle \end{bmatrix}$$

$\underline{y - X\beta}$

Machine learning goal: minimize the loss function

$$L(\boldsymbol{\beta}) : \mathbb{R}^d \rightarrow \mathbb{R}.$$

Find possible optima by determining for which $\boldsymbol{\beta} = [\beta_1, \dots, \beta_d]$ all the partial derivatives equals $\mathbf{0}$. I.e. when do we have:

$$\nabla L(\boldsymbol{\beta}) = \begin{bmatrix} \frac{\partial L}{\partial \beta_1} \\ \frac{\partial L}{\partial \beta_2} \\ \vdots \\ \frac{\partial L}{\partial \beta_d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

The vector of partial derivatives of $L(\boldsymbol{\beta})$ is called the gradient of $L(\boldsymbol{\beta})$, denoted by $\nabla L(\boldsymbol{\beta})$.

Loss function:

$$L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Gradient:

$$-2 \cdot \mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta)$$

Can check that this is equal to 0 if $\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. There are no other options, so this must be the minimum.

SINGLE VARIABLE WARMUP

What is the derivative of: $f(x) = x^2$?

Loss function:

$$L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Loss function:

$$L(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2$$

Take away: Simple form for the gradient means that multiple linear regression models are easy and efficient to optimize.

$$\beta^* = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- Often the “go to” first regression method. Throw your data in a matrix and see what happens.
- Serve as the basis for much richer classes of models.

It is not always immediately clear how to do this! One of the first issue we run into is categorical data:

$$\mathbf{x}_1 = [42, 4, 104, \text{hybrid}, \text{ford}]$$
$$\mathbf{x}_2 = [18, 8, 307, \text{gas}, \text{bmw}]$$
$$\mathbf{x}_2 = [31, 4, 150, \text{gas}, \text{honda}]$$
$$\vdots$$

ENCODING DATA AS A NUMERICAL MATRIX

Binary data is easy to deal with – pick one category to be 0, one to be 1. The choice doesn't matter – it will not impact the overall loss of the model

$$\mathbf{x}_1 = [42, 4, 104, \text{hybrid}, \text{ford}]$$
$$\mathbf{x}_2 = [18, 8, 307, \text{gas}, \text{bmw}]$$
$$\mathbf{x}_2 = [31, 4, 150, \text{gas}, \text{honda}]$$
$$\vdots$$
$$\mathbf{x}_1 = [42, 4, 104, 1, \text{ford}]$$
$$\mathbf{x}_2 = [18, 8, 307, 0, \text{bmw}]$$
$$\mathbf{x}_2 = [31, 4, 150, 0, \text{honda}]$$
$$\vdots$$

What about a categorical predictor variable for car make with more than 2 options: e.g. Ford, BMW, Honda. **How would you encode as a numerical column?**

$$\begin{bmatrix} \text{ford} \\ \text{ford} \\ \text{honda} \\ \text{bmw} \\ \text{honda} \\ \text{ford} \end{bmatrix} \rightarrow \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

Better approach: One Hot Encoding.

$$\begin{bmatrix} \text{ford} \\ \text{ford} \\ \text{honda} \\ \text{bmw} \\ \text{honda} \\ \text{ford} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

- Create a separate feature for every category, which is 1 when the variable is in that category, zero otherwise.
- Not too hard to do by hand, but you can also use library functions like `sklearn.preprocessing.OneHotEncoder`.

Avoids adding inadvertent linear relationships.