# CS-GY 6923: Lecture 1
# Introduction to Machine Learning

NYU Tandon School of Engineering, Akbar Rafiey

# Artificial intelligence is having a moment

Who has tried ChatGPT? DALLE? Imagen?

Paint the Iron Throne from Game of Therons with inspiration from a pineapple

# Artificial intelligence is having a moment

## Reasoning ?

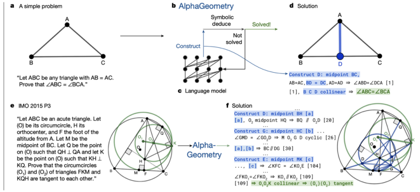
**Article**

# Solving olympiad geometry without human demonstrations

https://doi.org/10.1038/s41586-023-06747-5
Received: 30 April 2023
Accepted: 13 October 2023
Published online: 17 January 2024
Open access
Check for updates

Trieu H. Trinh[1,2✉], Yuhuai Wu[1], Quoc V. Le[1], He He[2] & Thang Luong[1✉]

Proving mathematical theorems at the olympiad level represents a notable milestone in human-level automated reasoning[1–4], owing to their reputed difficulty among the world's best talents in pre-university mathematics. Current machine-learning approaches, however, are not applicable to most mathematical domains owing to the high cost of translating human proofs into machine-verifiable format. The problem is even worse for geometry because of its unique translation challenges[1,5], resulting in



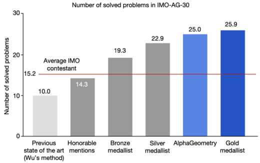
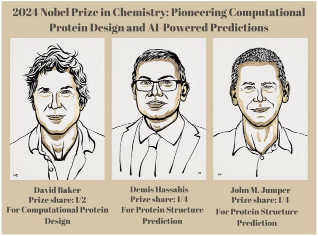
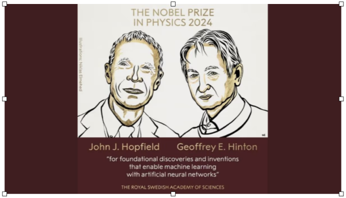




**Fig. 2 | AlphaGeometry advances the current state of geometry theorem prover from below human level to near gold-medallist level.** The test

# Artificial intelligence is having a moment

In other sciences and governmental level:





Trump tech agenda begins with $500B private AI plan and cuts to regulation

Executives from Softbank, OpenAI and Oracle joined Trump at the White House on Tuesday to announce "Stargate," a $500 billion effort to build new AI data centers.

January 21, 2025

5 min 223

# Artificial intelligence is having a moment

**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 technologies have caught people's eye?**

# Goal of this class

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.

# Course objectives

**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, research, and job.
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.

# Course objectives

**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 thse problems.

# more advanced classes at tandon

- CS-GY 6953: **Deep Learning** (Prof. Chinmay Hegde)
- ECE-GY 7143: **Advanced Machine Learning** (Prof. Anna Chromanska)
- CS-GY 6763: **Algorithmic Machine Learning and Data Science** (Prof. Christopher Musco)
- Keep your eyes out for special topics courses.

# Basic information

All class information can be found at:

`https://akbarrafiey.github.io/sp25-ml6923/`

# Two most important things from syllabus

1. Make sure you are signed into and follow **EdStem**, which will be used for all classroom communication (no email). Now integrated into Brightspace.
2. We will be using **Gradescope** for Lab and Homework assignments.

# Course team

- Don't hesitate to ask me or the TAs for help. (Fill out office hours poll on Ed!)
- Course Assistant

Adith Santosh

Sreeharsh Namani

# Class participation

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

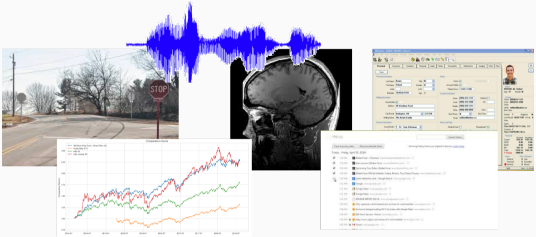
- Ask and answer questions in lecture.
- Post questions or responses to other students on Ed. Or other things you find interesting.
- Participate in professor or TA office hours.

# The prediction problem

# Basic goal

**Goal:** Develop algorithms (functions) to make predictions based on data.

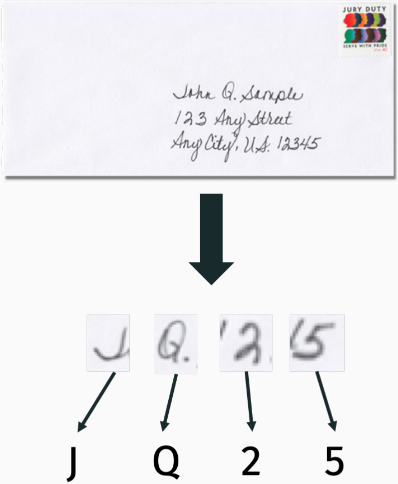
- **Input:** A single piece of data (an image, audio file, patient healthcare record, MRI scan).




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

# Classic example

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



# Classic example

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

**Applications:**

- Automatic mail sorting.
- Text search in handwritten documents.
- Digitizing scanned books.
- License plate detection for tolls.
- Etc.

# Exercise: expert systems

How would you write a **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.

```
def count_vert_lines(image):
...

def count_horiz_lines(image):
...

def classify(image):
...
    nv = count_vert_lines(image)
    nh = count_vert_lines(image)

    if (nv == 1) and (nh == 1):
        return '7'
    elif (nv == 1) and (nh == 0):
        return '1'
    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.

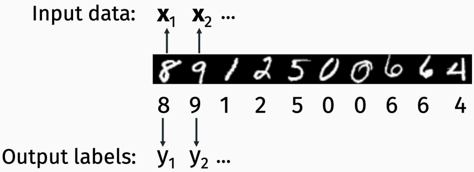
**Major limitation:** While human's are very good at many tasks,

- 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 $y_i \in \{0, 1, \ldots, 9\}$.




This is called the **training dataset.**

## A different approach: supervised machine learning

**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).

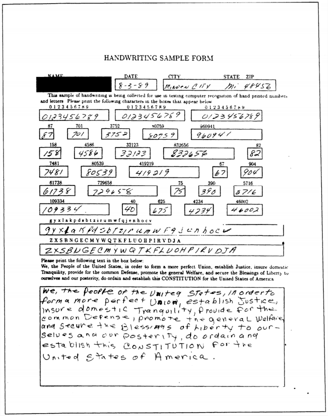
Think of $f$ as any crazy equation, or an arbitrary program:

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

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:



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY Minden City STATE Mi ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below

0 1 2 3 4 5 6 7 8 9 — 0123456789
0 1 2 3 4 5 6 7 8 9 — 0123456789
0 1 2 3 4 5 6 7 8 9 — 0123456789

87 701 3752 80759 960941
158 4586 32123 832656 82
7481 80539 419219 67 904
61738 729658 75 390 5716
109334 40 625 4234 46002

g y x l a k p d s b t z i r u m w f q j e n h o c v

Z X S B N G E C M Y W Q T K F L U O H P I R V D J A

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

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

# Machine learning

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).[1]
- Very successful on other problems as well. The big break through for supervised learning in the 2010s was image classification.

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

# Central questions in supervised machine learning

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)?

# Machine learning

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 $y_i$.

# Supervised learning

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

Each input data example $\mathbf{x}_i$ 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.)

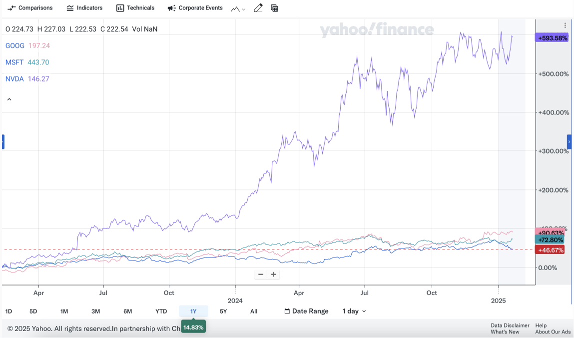
# Supervised learning

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, Price/Earning ratio, earning reports, historical price data).

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

# Supervised learning

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.
- **Semi-supervised learning, self-supervised learning**.

Focus less in this class on:

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

# Supervised learning

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 $y_i$.

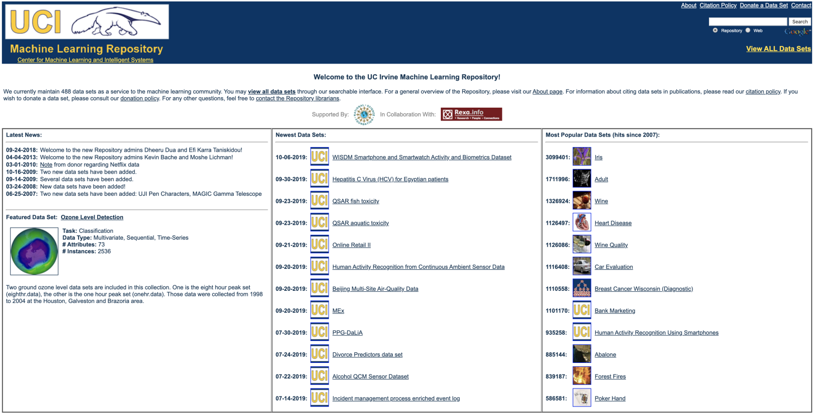
# Predicting mpg

**Motivating example:** Predict the highway miles per gallon (MPG) of a car given quantitative information about its engine. Demo in `demo_auto_mpg.ipynb (Demo 2)`.

**What factors might matter?**

# Predicting mpg

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



# Predicting mpg

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

auto-mpg.data

```
18.0   8   307.0      130.0      3504.      12.0   70  1	"chevrolet chevelle malibu"
15.0   8   350.0      165.0      3693.      11.5   70  1	"buick skylark 320"
18.0   8   318.0      150.0      3436.      11.0   70  1	"plymouth satellite"
16.0   8   304.0      150.0      3433.      12.0   70  1	"amc rebel sst"
17.0   8   302.0      140.0      3449.      10.5   70  1	"ford torino"
15.0   8   429.0      198.0      4341.      10.0   70  1	"ford galaxie 500"
14.0   8   454.0      220.0      4354.       9.0   70  1	"chevrolet impala"
14.0   8   440.0      215.0      4312.       8.5   70  1	"plymouth fury iii"
14.0   8   455.0      225.0      4425.      10.0   70  1	"pontiac catalina"
15.0   8   390.0      190.0      3850.       8.5   70  1	"amc ambassador dpl"
15.0   8   383.0      170.0      3563.      10.0   70  1	"dodge challenger se"
14.0   8   340.0      160.0      3609.       8.0   70  1	"plymouth 'cuda 340"
15.0   8   400.0      150.0      3761.       9.5   70  1	"chevrolet monte carlo"
14.0   8   455.0      225.0      3086.      10.0   70  1	"buick estate wagon (sw)"
24.0   4   113.0      95.00      2372.      15.0   70  3	"toyota corona mark ii"
22.0   6   198.0      95.00      2833.      15.5   70  1	"plymouth duster"
18.0   6   199.0      97.00      2774.      15.5   70  1	"amc hornet"
21.0   6   200.0      85.00      2587.      16.0   70  1	"ford maverick"
27.0   4   97.00      88.00      2130.      14.5   70  3	"datsun pl510"
26.0   4   97.00      46.00      1835.      20.5   70  2	"volkswagen 1131 deluxe sedan"
25.0   4   110.0      87.00      2672.      17.5   70  2	"peugeot 504"
24.0   4   107.0      90.00      2430.      14.5   70  2	"audi 100 ls"
25.0   4   104.0      95.00      2375.      17.5   70  2	"saab 99e"
26.0   4   121.0      113.0      2234.      12.5   70  2	"bmw 2002"
21.0   6   199.0      90.00      2648.      15.0   70  1	"amc gremlin"
10.0   8   360.0      215.0      4615.      14.0   70  1	"ford f250"
10.0   8   307.0      200.0      4376.      15.0   70  1	"chevy c20"
11.0   8   318.0      210.0      4382.      13.5   70  1	"dodge d200"
9.0    8   304.0      193.0      4732.      18.5   70  1	"hi 1200d"
27.0   4   97.00      88.00      2130.      14.5   71  3	"datsun pl510"
28.0   4   140.0      90.00      2264.      15.5   71  1	"chevrolet vega 2300"
25.0   4   113.0      95.00      2228.      14.0   71  3	"toyota corona"
```

# Predicting mpg

Check dataset description to know what each column means.

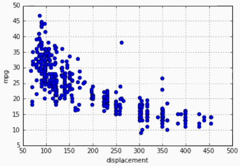
auto-mpg.data

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 18.0 | 8 | 307.0 | 130.0 | 3504. | 12.0 | 70 | 1 | "chevrolet chevelle malibu" |
| 15.0 | 8 | 350.0 | 165.0 | 3693. | 11.5 | 70 | 1 | "buick skylark 320" |
| 18.0 | 8 | 318.0 | 150.0 | 3436. | 11.0 | 70 | 1 | "plymouth satellite" |
| 16.0 | 8 | 304.0 | 150.0 | 3433. | 12.0 | 70 | 1 | "amc rebel sst" |
| 17.0 | 8 | 302.0 | 140.0 | 3449. | 10.5 | 70 | 1 | "ford torino" |
| 15.0 | 8 | 429.0 | 198.0 | 4341. | 10.0 | 70 | 1 | "ford galaxie 500" |
| 14.0 | 8 | 454.0 | 220.0 | 4354. | 9.0 | 70 | 1 | "chevrolet impala" |
| 14.0 | 8 | 440.0 | 215.0 | 4312. | 8.5 | 70 | 1 | "plymouth fury iii" |
| 14.0 | 8 | 455.0 | 225.0 | 4425. | 10.0 | 70 | 1 | "pontiac catalina" |
| 15.0 | 8 | 390.0 | 190.0 | 3850. | 8.5 | 70 | 1 | "amc ambassador dpl" |
| 15.0 | 8 | 383.0 | 170.0 | 3563. | 10.0 | 70 | 1 | "dodge challenger se" |
| 14.0 | 8 | 340.0 | 160.0 | 3609. | 8.0 | 70 | 1 | "plymouth 'cuda 340" |
| 15.0 | 8 | 400.0 | 150.0 | 3761. | 9.5 | 70 | 1 | "chevrolet monte carlo" |
| 14.0 | 8 | 455.0 | 225.0 | 3086. | 10.0 | 70 | 1 | "buick estate wagon (sw)" |
| 24.0 | 4 | 113.0 | 95.00 | 2372. | 15.0 | 70 | 3 | "toyota corona mark ii" |
| 22.0 | 6 | 198.0 | 95.00 | 2833. | 15.5 | 70 | 1 | "plymouth duster" |
| 18.0 | 6 | 199.0 | 97.00 | 2774. | 15.5 | 70 | 1 | "amc hornet" |
| 21.0 | 6 | 200.0 | 85.00 | 2587. | 16.0 | 70 | 1 | "ford maverick" |
| 27.0 | 4 | 97.00 | 88.00 | 2130. | 14.5 | 70 | 3 | "datsun pl510" |
| 26.0 | 4 | 97.00 | 46.00 | 1835. | 20.5 | 70 | 2 | "volkswagen 1131 deluxe sedan" |
| 25.0 | 4 | 110.0 | 87.00 | 2672. | 17.5 | 70 | 2 | "peugeot 504" |
| 24.0 | 4 | 107.0 | 90.00 | 2430. | 14.5 | 70 | 2 | "audi 100 ls" |
| 25.0 | 4 | 104.0 | 95.00 | 2375. | 17.5 | 70 | 2 | "saab 99e" |
| 26.0 | 4 | 121.0 | 113.0 | 2234. | 12.5 | 70 | 2 | "bmw 2002" |
| 21.0 | 6 | 199.0 | 90.00 | 2648. | 15.0 | 70 | 1 | "amc gremlin" |
| 10.0 | 8 | 360.0 | 215.0 | 4615. | 14.0 | 70 | 1 | "ford f250" |
| 10.0 | 8 | 307.0 | 200.0 | 4376. | 15.0 | 70 | 1 | "chevy c20" |
| 11.0 | 8 | 318.0 | 210.0 | 4382. | 13.5 | 70 | 1 | "dodge d200" |
| 9.0 | 8 | 304.0 | 193.0 | 4732. | 18.5 | 70 | 1 | "hi 1200d" |
| 27.0 | 4 | 97.00 | 88.00 | 2130. | 14.5 | 71 | 3 | "datsun pl510" |
| 28.0 | 4 | 140.0 | 90.00 | 2264. | 15.5 | 71 | 1 | "chevrolet vega 2300" |
| 25.0 | 4 | 113.0 | 95.00 | 2228. | 14.0 | 71 | 3 | "toyota corona" |

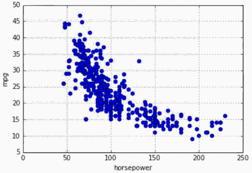
'mpg', 'cylinders','displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car name'

- 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.

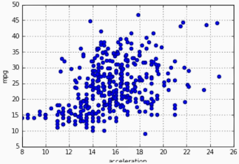
mpg




Displacement




Horsepower



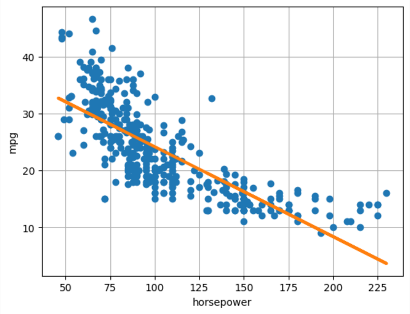

Acceleration

# Simple linear regression

# Simple linear regression

Our first supervised machine learning model:

**Linear regression from a Machine Learning (not a Statistics) perspective.**

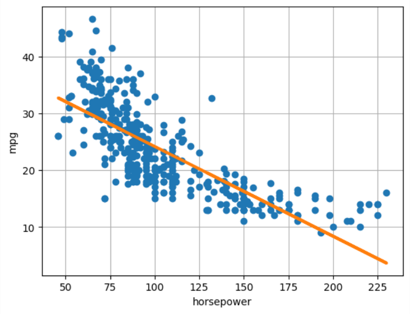



Only focus on one predictive variable at a time (e.g. horsepower).
This is why it's called simple linear regression.

# Simple linear regression

**Dataset:**

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



# Supervised learning definitions

- **Model $f_{\boldsymbol{\theta}}(x)$:** Class of functions, equations, or programs which map input $x$ to a predicted output.

  We want $f_{\boldsymbol{\theta}}(x_i) \approx y_i$ for training inputs.

- **Model Parameters $\boldsymbol{\theta}$:** Vector of numbers. These are numerical knobs which parameterize our class of models.

# Supervised learning definitions

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

# Supervised learning definitions

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

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

Choosing $\boldsymbol{\theta}^*$ 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**

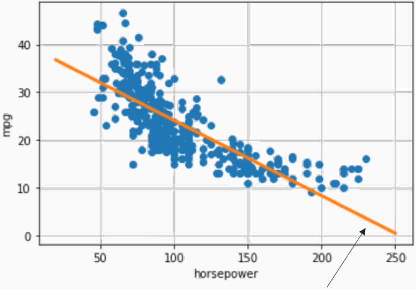
- Model: $f_{\boldsymbol{\theta}}(x)$
- Model Parameters: $\boldsymbol{\theta}$
- Loss Function: $L(\boldsymbol{\theta})$

**Linear Regression**

- Model:
- Model Parameters:
- Loss Function:

# How to measure goodness of fit

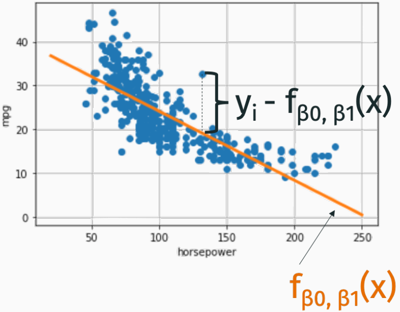
What is a natural **loss function** for linear regression?




$f_{\beta 0,\ \beta 1}(x)$

# How to measure goodness of fit

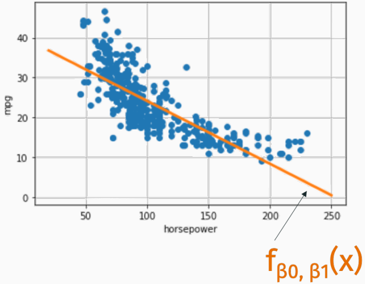
Typical choices are a function of $y_1 - f_{\beta_0,\beta_1}(x_1), \ldots, y_n - f_{\beta_0,\beta_1}(x_n)$




- $\ell_2$/**Squared Loss:** $L(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - f_{\beta_0,\beta_1}(x_i))^2$.
- $\ell_1$/**Least Absolute Deviations:** $L(\beta_0, \beta_1) = \sum_{i=1}^{n} |y_i - f_{\beta_0,\beta_1}(x_i)|$.
- $\ell_\infty$ **Loss** $L(\beta_0, \beta_1) = \max_{i \in 1,\ldots,n} |y_i - f_{\beta_0,\beta_1}(x_i)|$.

# 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 $\beta_0, \beta_1$
- Theoretically justified from classical statistics related to assumptions about Gaussian noise. Will discuss later in the course.

# Linear regression

**General Supervised Learning**

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

**Linear Regression**

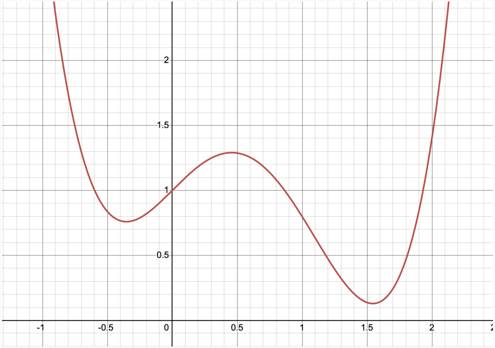
- Model: $f_{\beta_0,\beta_1}(x) = \beta_0 + \beta_1 \cdot x$
- Model Parameters: $\beta_0, \beta_1$
- Loss Function: $L(\beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - f_{\beta_0,\beta_1}(x_i))^2$

**Goal:** Choose $\beta_0, \beta_1$ 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:**




$$x^4 - 2.2 \cdot x^3 + x + 1$$

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

# Function minimization

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

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

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

# 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 $\beta_0, \beta_1$ where all partial derivatives equal 0.
- $\frac{\partial L}{\partial \beta_0} = 0$ and $\frac{\partial L}{\partial \beta_1} = 0$.

**Some definitions:**

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

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

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

# Proof

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

# Proof

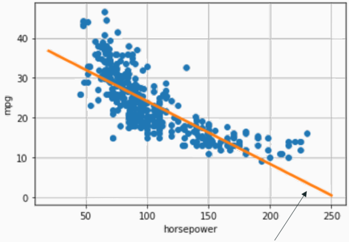
**Loss function after substitution:**

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

# 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 $\beta_0^*$ and $\beta_1^*$ for squared-loss!




$B_0^* + B_1^* x$

# A few comments

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

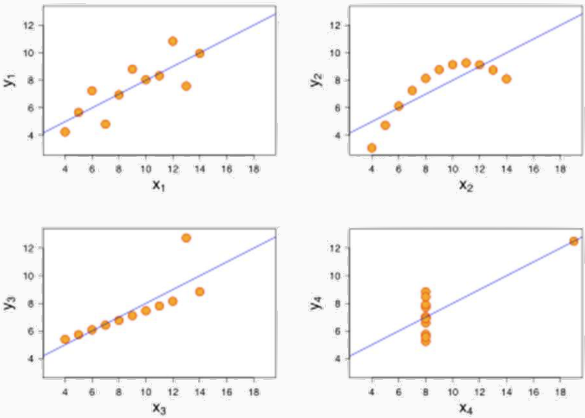
$$R^2 = 1 - \frac{L(\beta_0, \beta_1)}{n\sigma_y^2}$$

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:**

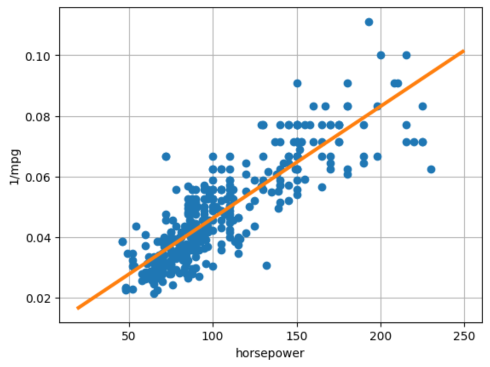


## a few comments

Some of these are fixable!

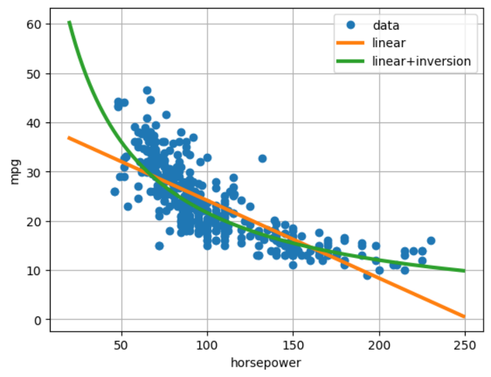
- Remove outliers, use more robust loss function.
- **Non-linear model transformation.**

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



# A few comments

- Fit the model $\frac{1}{\text{mpg}} \approx \beta_0 + \beta_1 \cdot \text{horsepower}$.
- Compute the estimate in the original domain




**Much better fit, same exact learning algorithm!**

**Next time: multiple linear regression**