CS-GY 6923: Lecture 1
Introduction to Machine Learning

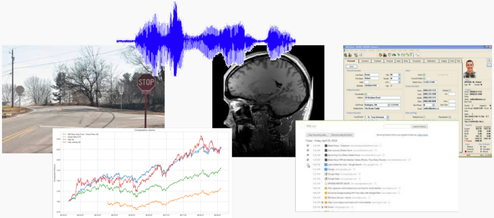
NYU Tandon School of Engineering, Prof. Christopher Musco

NO RIGHT ANSWERS

- What is Machine Learning?
- How is it different, the same as Artificial Intelligence?
- How is it different, the same as Statistics?

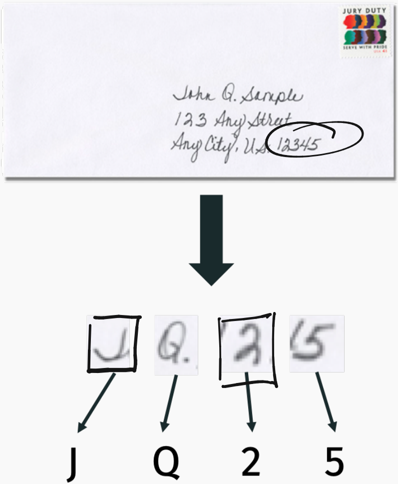
Goal: Develop algorithms to make decisions or predictions based on data.

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



- **Output:** A prediction or decision (this image is a stop sign, this stock will go up 10% next quarter, turn the car right).

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



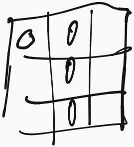
Optical character recognition (OCR): Decide if a handwritten character is an $a, b, \ldots, z, 0, 1, \ldots, 9, \ldots$.

Applications:

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

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

Suppose you just want to distinguish <u>between a 1 and a 7</u>.



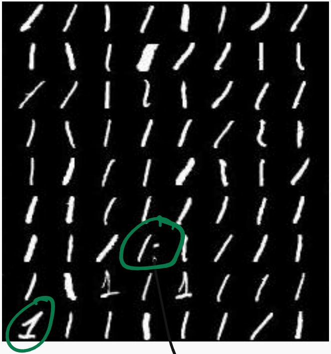
1s vs. 7s algorithm

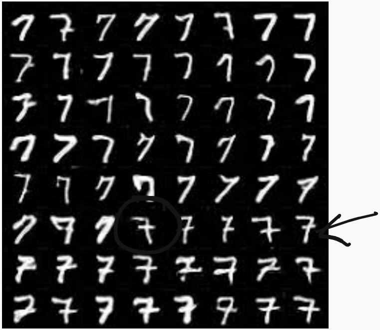
Reasonable approach: A number which contains one vertical line is a 1, if it contains one vertical and one horizontal line, it's a 7.

```python
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.

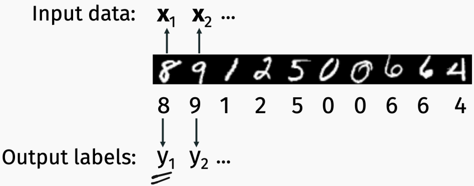
Rule based systems, also called Expert Systems were the dominant approach to artificial intelligence in the 70s and 80s.

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?)

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.

Step 2: Learn from the examples we have.

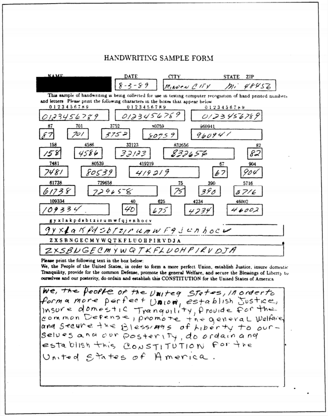
- Have the computer automatically find some function $f(\mathbf{x})$ such that $f(\mathbf{x}_i) = y_i$ for most $(\mathbf{x}_i, y_i)$ in our training data set (by searching over many possible functions).

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.**

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

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

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.

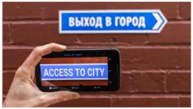
- Current methods achieve .17% error rate[1] for OCR on benchmark datasets (MNIST).[2]
- Very successful on other problems as well.

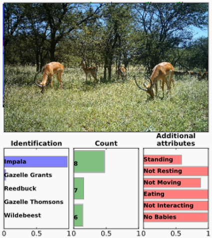
[1] Last time I taught this course it was .21%.

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

You could not be studying ML at a more exciting time!

- Autonomous vehicles.
- Human level play in very difficult games.
- Incredible machine translation.
- Pervasive impact in science and engineering.
- Many, many more.





Machine learning has benefited from an explosion in our ability to collect and store data:

- Cheap, fast storage. Large data centers accessible via the cloud.
- Pervasive monitoring (satellite imagery, cheap sensors, improved and reduced cost for technologies like LIDAR).
- Crowd-sourced data collection (images, text on the internet)
- Crowd-sourced data labeling via the internet (Amazon Mechanical Turk, reCAPTCHA, etc.)

Having lots of data isn't enough. We have to know how to use it effectively.

Once we have the basic 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)?

In this course you will learn to answer these central questions through a combination of:

- Hands on implementation.
    - In-class demos and take-home labs using `Python` and `Jupyter notebooks`.
    - We will use `Google Colab` as the primary programming environment.
    - Mini-final project (on any dataset/problem you like).
- Theoretical exploration.
    - Written problem sets.
    - Midterm and final exam.

Goals of hands-on component:

1. Learn how to view and formulate real world problems in the language of machine learning.
2. Gain experience applying the most popular and successful machine learning algorithms to example problems.

Goals of theoretical component:

1. Learn how theoretical analysis can help explain the performance of machine learning algorithms and lead to the design of entirely new methods.
2. 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.
3. Be able to understand contemporary research in machine learning, including papers from NeurIPS, ICML, ICLR, and other major machine learning venues.

- CS-GY 6763: **Algorithmic Machine Learning and Data Science** (Dr. Rajesh Jayaram)
- CS-GY 9223: **Statistical and Computational Foundations of Machine Learning** (Dr. David Pal)
- CS-GY 9223: **Foundations of Deep Learning** (Prof. Chinmay Hegde, Ph.D. students only)

All class information can be found at:

`www.chrismusco.com/machinelearning2022`

TWO MOST IMPORTANT THINGS FROM SYLLABUS

1. Make sure you are signed up for Ed Stem which will be used for all classroom communication (no email).
2. Don't hesitate to ask me or the TAs for help.[3]

Thomas Liu

Siddharth Sagar

Ozlem Yildiz

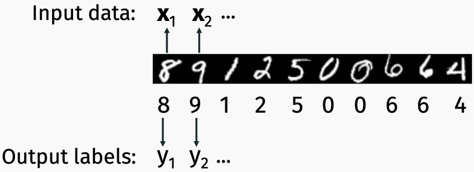
[3] Fill out office hours poll on Ed!

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.

SIMPLE LINEAR REGRESSION

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.**

Step 2: Learn from the examples we have.

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

In supervised learning 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 earning:

- **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**.

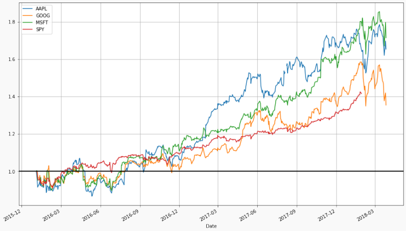
Each input data example $\mathbf{x}_i$ is an image. Each output y_i is 1 if the image contains a face, 0 otherwise.

- Harder than digit recognition, but we now have very reliable 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)

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



Each input **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 $t+1$)

Later in the class we will talk about other models:

- **Unsupervised learning** (no labels or response variable)
    - Clustering
    - Representation Learning
- **Reinforcement learning**
    - Game playing

You might also hear about semi-supervised learning or active learning – these categories aren't always cut and dry.

In **supervised learnings** 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).

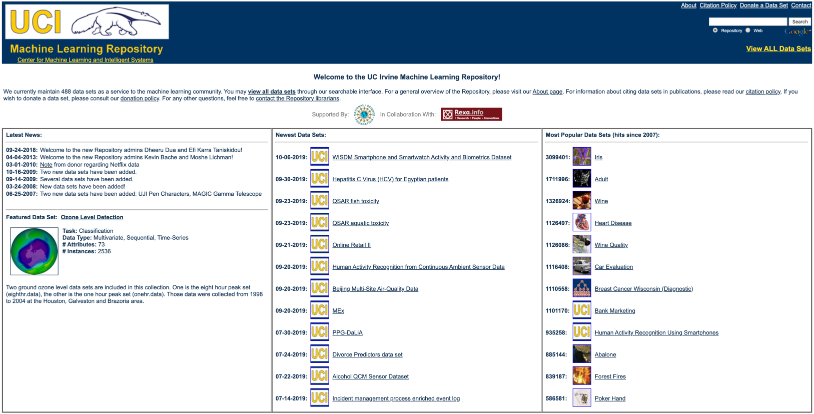
Types of supervised earning:

- **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/.



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

```
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"
```

Check dataset description to know what each column means.

housing.data   auto-mpg.data ×

Users > christophermusco > Desktop > auto-mpg.data

y_1 y_2 y_3 x_1 x_2 x_3

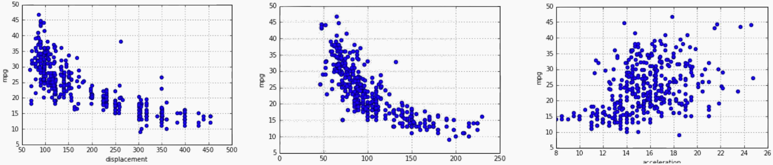
18.0	8	307.0	130.0	3504.	12.0	70	1	"chevrolet chevelle malibu"
15.0	8	350.0	165.0	3693.	11.5	70	1	"buick skylark 320"
18.0	8	318.0	150.0	3436.	11.0	70	1	"plymouth satellite"
16.0	8	304.0	150.0	3433.	12.0	70	1	"amc rebel sst"
17.0	8	302.0	140.0	3449.	10.5	70	1	"ford torino"
15.0	8	429.0	198.0	4341.	10.0	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"

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

LIBRARIES FOR INITIAL DATA READING

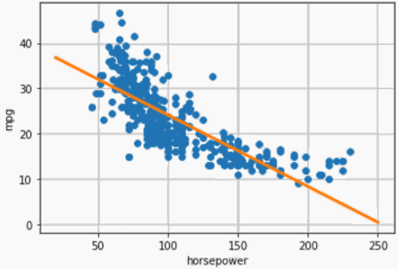
- Use `pandas` for reading data from delimited files. Stores data in a type of table called a “data frame” but this is just a wrapper around a `numpy` array.
- Use `matplotlib` for initial exploration.

mpg



Displacement Horsepower Acceleration

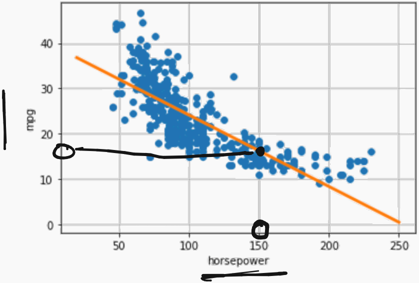
Linear regression from a Machine Learning (not a Statistics) perspective. Our first supervised machine learning model.



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

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)



- **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 nobs which parameterize our class of models.
- **Loss Function $L(\boldsymbol{\theta})$:** Measure of how well a model fits our data. Often some function of $f_{\boldsymbol{\theta}}(x_1) - y_1, \ldots, f_{\boldsymbol{\theta}}(x_n) - y_n$ $\sum_{i=1}^{n} |f_{\theta}(x_i) - y_i|$

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.

General Supervised Learning

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

$x_1, \ldots, x_n$

$y_1, \ldots, y_n$

Linear Regression

- Model:

$$f_\theta(x) = \overset{m}{B_1} x + \overset{b}{B_0}$$

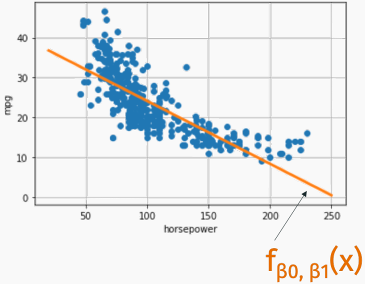
- Model Parameters:

$$\theta = [B_0, B_1]$$

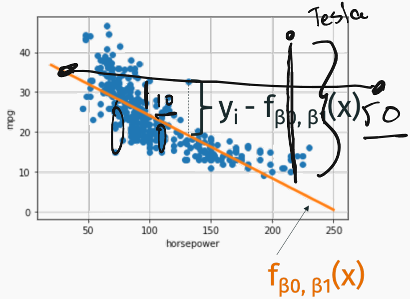
- Loss Function:

$$L(B_0, B_1) = \sum_{i=1}^{n} (B_1 x_i + B_0 - y_i)^2$$

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



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



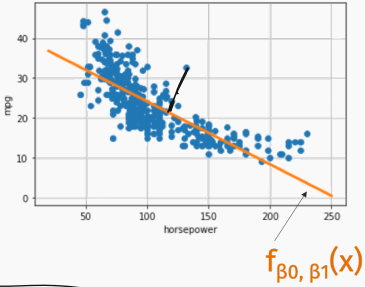
$\min_{\theta} L(\theta)$

$\min_{\theta} c \cdot L(\theta)$

- ℓ_2/**Squared Loss:** $L(\beta_0, \beta_1) = \frac{1}{n}\sum_{i=1}^{n} [y_i - f_{\beta_0,\beta_1}(x_i)]^2$.
- ℓ_1/**Lease absolute deviations:** $L(\beta_0, \beta_1) = \sum_{i=1}^{n} |y_i - f_{\beta_0,\beta_1}(x_i)|$.
- ℓ_∞ **Loss** $L(\beta_0, \beta_1) = \max_{i \in 1,\ldots,n} |y_i - f_{\beta_0,\beta_1}(x_i)|$.

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 β_0, β_1
- Theoretically justified from classical statistics related to assumptions about Gaussian noise. Will discuss later in the course.

General Supervised Learning

$\theta \to \beta_0, \beta_1$

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

Linear Regression

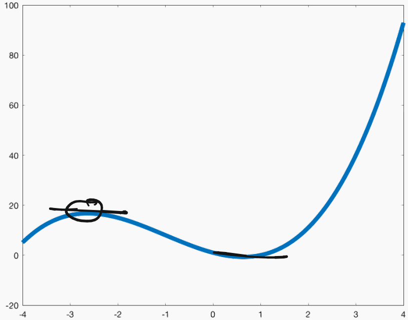
- Model: $\underline{f_{\beta_0,\beta_1}(x)} = \underline{\beta_0} + \underline{\beta_1 \cdot x}$
- Model Parameters: $\underline{\beta_0, \beta_1}$
- Loss Function: $L(\beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - f_{\beta_0,\beta_1}(x_i))^2$

Goal: Choose β_0, β_1 to minimize

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

This is the entire job of any Supervised Learning Algorithm.

Univariate function:



$$x^3 + 3 \cdot x^2 - 5 \cdot x + 1$$

- 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 $\frac{\partial L}{\partial \beta_1} = 0$.

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 $\sigma_y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$.	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 when:

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

$$0 = \frac{\partial L(\beta_0, \beta_1)}{\partial \beta_0} = \frac{\partial}{\partial \beta_0} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$0 = \sum_{i=1}^{n} \cancel{2}(y_i - \beta_0 - \beta_1 x_i) \cdot (-1)$$

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

$$0 = \underbrace{\frac{1}{n} \sum_{i=1}^{n} -y_i}_{-\bar{y}} + \underbrace{\frac{1}{n} \sum_{i=1}^{n} \beta_0}_{\beta_0} + \underbrace{\frac{\beta_1}{n} \sum_{i=1}^{n} x_i}_{\beta_1 \cdot \bar{x}}$$

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

PROOF

$$\tilde{L}(\beta_1) = L(\underline{\bar{y} - \beta_1 \bar{x}}, \beta_1)$$

$$\underline{\frac{\partial}{\partial \beta_0} L(\beta_0, \beta_1) = 0} \quad \text{and} \quad \frac{\partial}{\partial \beta_1} L(\beta_1, \beta_0) = 0$$

$$\text{if and only if} \quad \underline{\frac{\partial}{\partial \beta_1} \tilde{L}(\beta_1)} = 0.$$

$$0 = \frac{\partial}{\partial \beta_1} \sum_{i=1}^{n} \left(y_i - (\bar{y} - \beta_1 \bar{x}) - \beta_1 x_i\right)^2$$

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

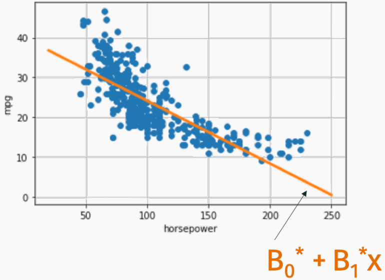
$$0 = \underbrace{\sum_{i=1}^{n} (y_i - \bar{y})(\bar{x} - x_i)}_{-\sigma_{xy} \cdot n} + \beta_1 \underbrace{\sum_{i=1}^{n} (\bar{x} - x_i)(\bar{x} - x_i)}_{n \sigma_x^2}$$

$$\to n \sigma_{xy} = \beta_1 n \sigma_x^2$$

$$\boxed{\beta_1 = \frac{\sigma_{xy}}{\sigma_x^2}}$$

Takeaways:

- Minimizing functions is sometimes easy with calculus, but not often! 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!



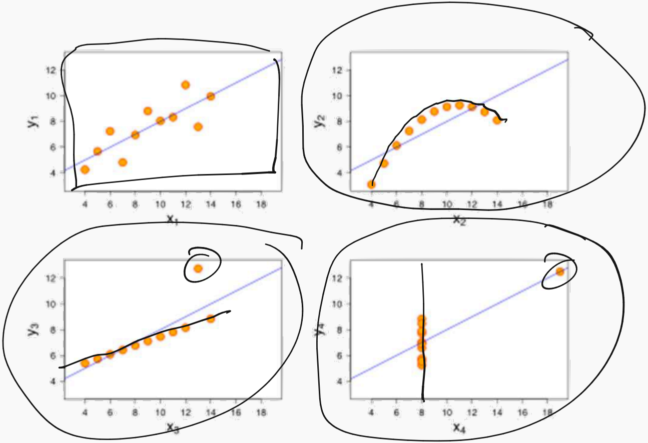
Let $L(\beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$.

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

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.

Many reasons you might get a poor regression fit:



Some of these are fixable!

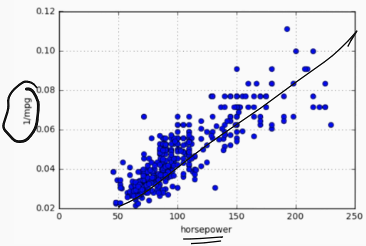
- Remove outliers, use more robust loss function.
- **Non-linear model transformation.**

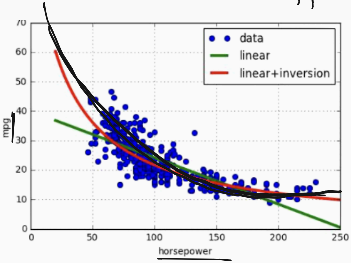
Fit the model $\frac{1}{\text{mpg}} \approx \beta_0 + \beta_1 \cdot \text{horsepower}$.

$1/\text{horsepower}$

horsepower^2

$\exp(\text{horsepower})$





Much better fit, same exact learning algorithm!