

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 Thrones 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^{1,2,3}, Yuhuai Wu¹, Quoc V. Le¹, He He² & Thang Luong^{1,2}

Proving mathematical theorems at the olympiad level represents a notable milestone in human-level automated reasoning¹⁻⁴, 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^{5,6}, resulting in

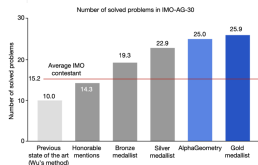
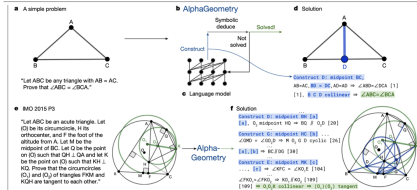
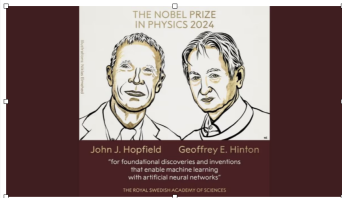
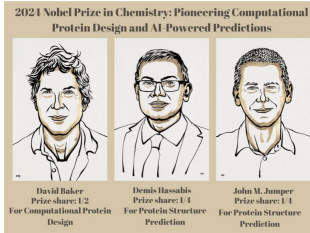


Fig. 2 | AlphaGeometry advances the current state of geometry theorem prover from below human level to near gold-medallist level. The test

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.

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.

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 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:

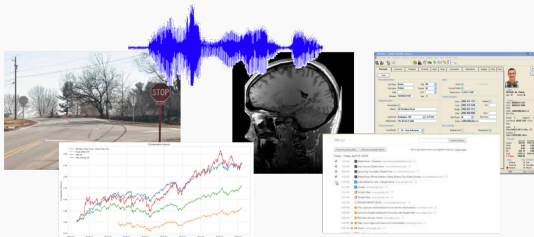
- Ask and answer questions in lecture.
- Post questions or responses to other students on Ed. Or other things you find interesting.
- Participate in professor or TA office hours.

The prediction problem

Basic goal

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

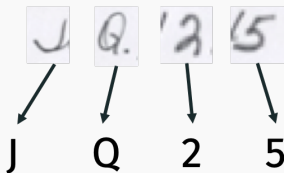
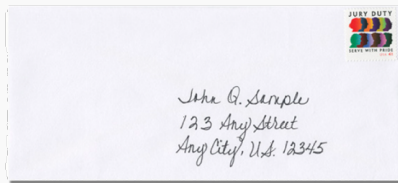
- **Input:** A single piece of data (an image, audio file, patient healthcare record, MRI scan).



- **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 $a, b, \dots, z, 0, 1, \dots, 9, \dots$



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.
- Etc.

Exercise: expert systems

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

0 1 2 3 4 5 6 7 8 9

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

0	1	0
0	1	0
0	1	0

1	1	0
0	1	0
0	1	0

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.

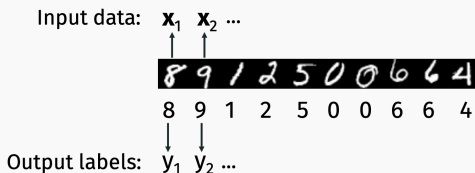
Major limitation: While human's are very good at many tasks,

- 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**.

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:

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

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

¹Not 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)?

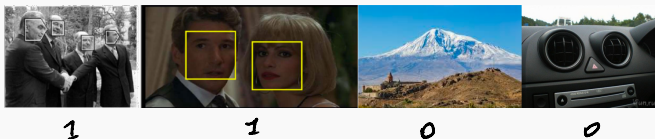
Recall that in the **supervised learning** setup every input x_i in our training dataset comes with a desired output y_i (typically generated by a human, or some other process).

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 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 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.

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.

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` (Demo 2).

What factors might matter?

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





UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems




















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

[View ALL Data Sets](#)

Welcome to the UC Irvine Machine Learning Repository!


We currently maintain 488 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):
<p>09-24-2018: Welcome to the new Repository admins Dheeru Dua and El Karra Tenenbaum!</p> <p>04-04-2013: Welcome to the new Repository admins Kevin Bache and Miroslav Lichner!</p> <p>03-01-2010: Note from donor regarding hepatic data</p> <p>10-16-2009: Two new data sets have been added.</p> <p>09-14-2009: Several data sets have been added.</p> <p>03-24-2008: New data sets have been added!</p> <p>06-29-2007: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope</p>	<p>10-06-2019:  WISDM Smartphones and Smartwatch Activity and Biometrics Dataset</p> <p>09-30-2019:  Hepatitis C Virus (HCV) for Egyptian patients</p> <p>09-23-2019:  QSAR fish toxicity</p> <p>09-23-2019:  QSAR aquatic toxicity</p> <p>09-21-2019:  Online Retail II</p> <p>09-20-2019:  Human Activity Recognition from Continuous Ambient Sensor Data</p> <p>09-20-2019:  Beijing Multi-Site Air Quality Data</p> <p>09-20-2019:  MEs</p> <p>07-30-2019:  PPQ-DaUA</p> <p>07-04-2019:  Chorizo Predictors data set</p> <p>07-22-2019:  Alcohol QCM Sensor Dataset</p> <p>07-14-2019:  Incident management process enriched event log</p>	<p>3093401:  Lia</p> <p>1711996:  Adult</p> <p>1329524:  Wine</p> <p>1126487:  Heart Disease</p> <p>1126086:  Wine Quality</p> <p>1116408:  Car Evaluation</p> <p>1110558:  Breast Cancer Wisconsin (Diagnostic)</p> <p>1101176:  Bank Marketing</p> <p>936286:  Human Activity Recognition Using Smartphones</p> <p>865144:  Alzabone</p> <p>839187:  Forest Fires</p> <p>568581:  Poker Hand</p>

Featured Data Set: Ozone Level Detection

Task: Classification
Data Types: 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 (eph8r-data), the other is the one hour peak set (onehr-data). These data were collected from 1998 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.

```
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 3886. 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. x_1

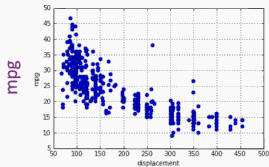
The screenshot shows a terminal window with the following data:

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
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 113i 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"

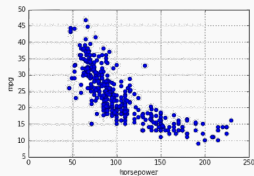
'mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'model year', 'origin', 'car name'

Libraries for initial data reading

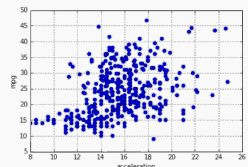
- Use `pandas` for reading data from delimited files. Stores data in a type of table called a “data frame” but this is just a wrapper around a `numpy` array.
- Use `matplotlib` for initial exploration.



Displacement



Horsepower



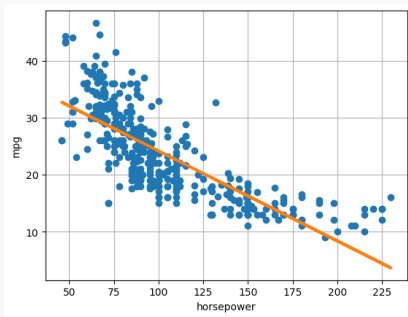
Acceleration

Simple linear regression

Simple linear regression

Our first supervised machine learning model:

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

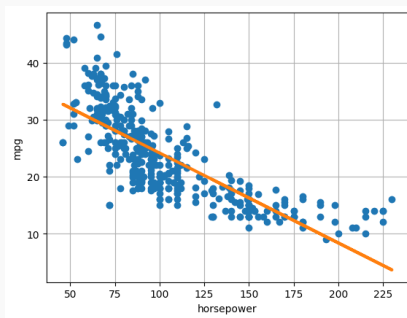


Only focus on one predictive variable at a time (e.g. horsepower).
This is why it's called simple linear regression.

Simple linear regression

Dataset:

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



Supervised learning definitions

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

Supervised learning definitions

- **Model** $f_{\theta}(x)$: 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 $f_{\theta}(x_1) - y_1, \dots, f_{\theta}(x_n) - y_n$

Supervised learning definitions

- **Model** $f_{\theta}(x)$: 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:

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

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: θ
- Loss Function: $L(\theta)$

Linear Regression

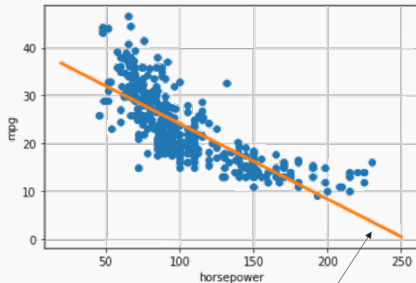
- Model:
 $f_{[\beta_0, \beta_1]}(x) = \beta_0 + \beta_1 \cdot x$
- Model Parameters:
 $\theta = [\beta_0, \beta_1]$
- Loss Function:

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

$$\sqrt{x^2} = |x|$$

How to measure goodness of fit

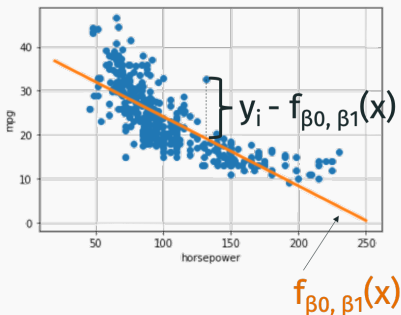
What is a natural **loss function** for linear regression?



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

How to measure goodness of fit

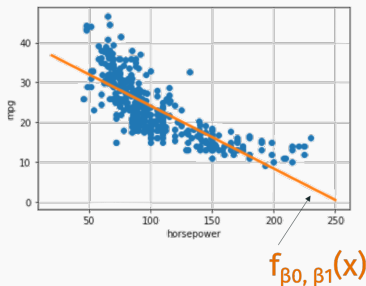
Typical choices are a function of $y_1 - f_{\beta_0, \beta_1}(x_1), \dots, y_n - f_{\beta_0, \beta_1}(x_n)$



- 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 Deviations: $L(\beta_0, \beta_1) = \sum_{i=1}^n |y_i - f_{\beta_0, \beta_1}(x_i)|$.
- l_∞ Loss $L(\beta_0, \beta_1) = \max_{i \in \{1, \dots, n\}} |y_i - f_{\beta_0, \beta_1}(x_i)|$.

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

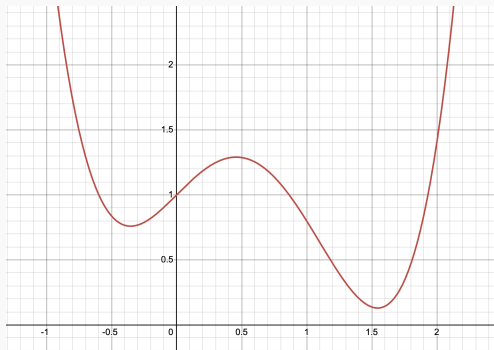
- Model:
 $f_{\beta_0, \beta_1}(x) = \beta_0 + \beta_1 \cdot x$
- Model Parameters: β_0, β_1
- Loss Function: $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - f_{\beta_0, \beta_1}(x_i))^2$

Goal: Choose β_0, β_1 to minimize
 $L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$.

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 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{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.

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{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 when:

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

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

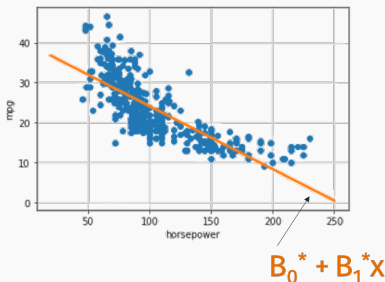
Loss function after substitution:

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

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 β_0^* and β_1^* for squared-loss!



A few comments

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

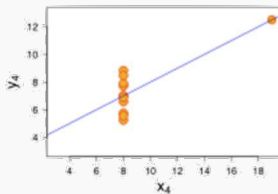
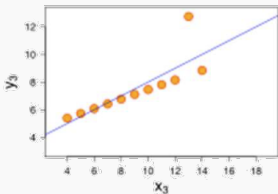
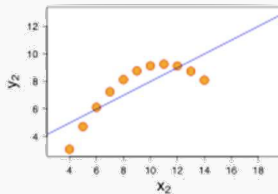
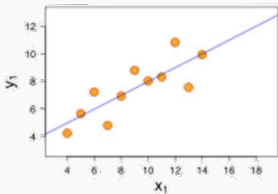
$$R^2 = 1 - \frac{L(\beta_0, \beta_1)}{n\sigma_y^2}$$

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:

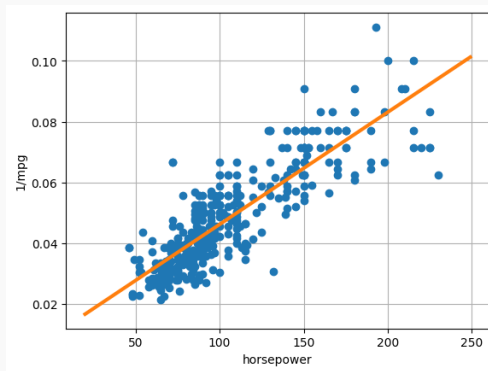


a few comments

Some of these are fixable!

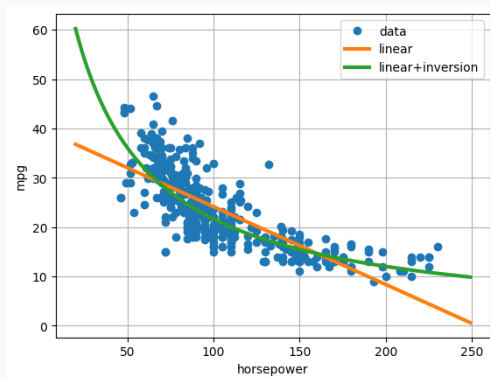
- Remove outliers, use more robust loss function.
- **Non-linear model transformation.**

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

More common goal


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` (Demo 3).

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](#)

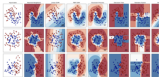
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

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



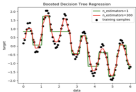
Examples

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...



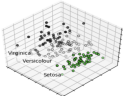
Examples

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...




Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

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



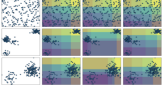
Examples

Preprocessing

Feature extraction and normalization.

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

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



Examples



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.

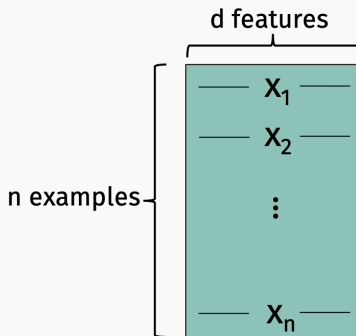
The data matrix

Target variable:

- 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



Now it the time to review your linear algebra!

Notation:

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

Things to remember:

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

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` (Demo 1) slowly.

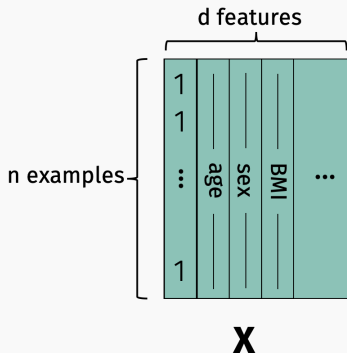
The data matrix

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:

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

Multiple Linear Regression Model:

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

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

Multiple linear regression

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

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

Multiple Linear Regression Model:

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

Data matrix:

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

Linear algebraic form:

$$y_i \sim \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle$$

$$\mathbf{y} \sim \mathbf{X}\boldsymbol{\beta}$$

Linear Least-Squares Regression.

- Model Parameters:

$$\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_d]$$

- Model:

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

- Loss Function:

$$\begin{aligned} L(\boldsymbol{\beta}) &= \sum_{i=1}^n |y_i - \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle|^2 \\ &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \end{aligned}$$

Linear algebraic form of loss function

Loss minimization

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 **gradient** equals $\mathbf{0}$. I.e. when do we have:

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

Gradient

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

Multiple linear regression solution

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.

Multiple linear regression solution

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

- Main cost is computing $(\mathbf{X}^T \mathbf{X})^{-1}$ which takes $O(nd^2)$ time.
- Can solve slightly faster using the method `numpy.linalg.lstsq`, which is running an algorithm based on QR decomposition.
- For larger problems, can solve much faster using an *iterative methods* like `scipy.sparse.linalg.lsqr`.

Will learn more about iterative methods when we study Gradient Descent.

Encoding data as a numerical matrix

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

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

Dealing with categorical variables

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?**

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

One hot encoding

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.