DEEP NETWORK DEVELOPMENT

Imre Molnár
PhD student, ELTE, AI Department
imremolnar@inf.elte.hu
curiouspercibal.github.io

Tamás Takács
PhD student, ELTE, AI Department
tamastheactual@inf.elte.hu
tamastheactual.github.io

Deep Network Development

Lecture 2.

Linear Regression & Artificial Neural Networks

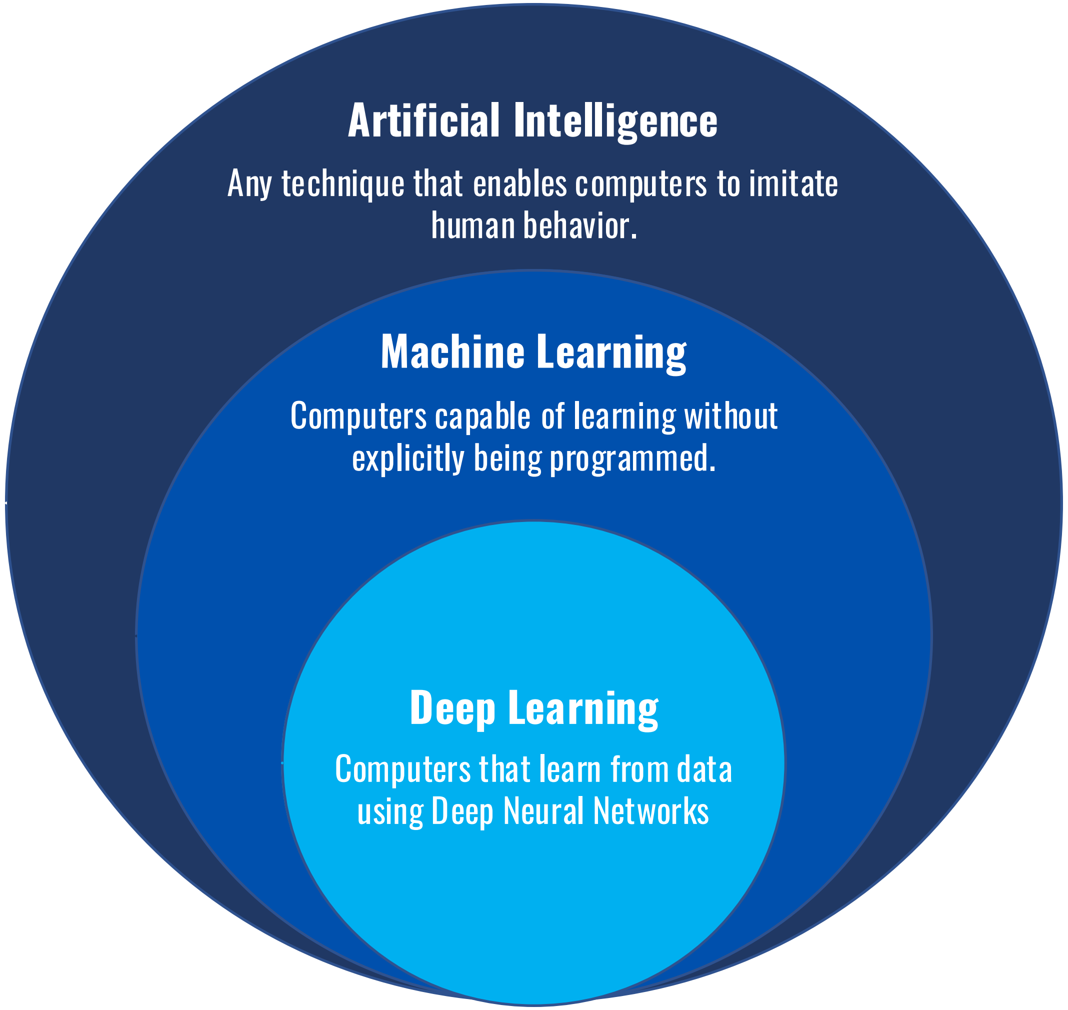
Budapest, 21st February 2025

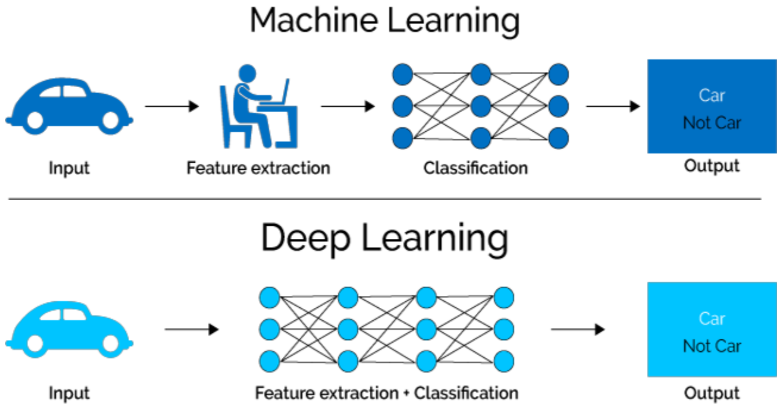
1 Linear Regression

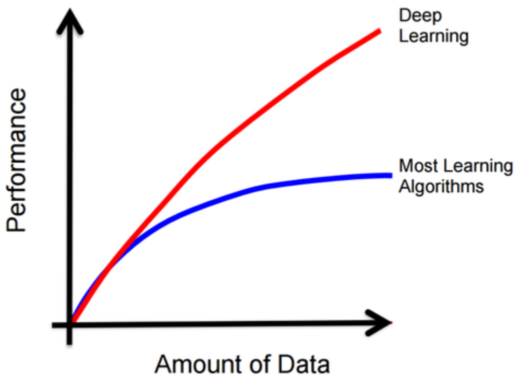
2 Artificial Neural Networks

3 Backpropagation

Previously on Lecture 1







Google DeepMind

originally published Nov. 2023; updated Jan. 2024

Levels of AGI: Operationalizing Progress on the Path to AGI

Meredith Ringel Morris[1], Jascha Sohl-dickstein[1], Noah Fiedel[1], Tris Warkentin[1], Allan Dafoe[1], Aleksandra Faust[1], Clement Farabet[1] and Shane Legg[1]
[1]Google DeepMind

We propose a framework for classifying the capabilities and behavior of Artificial General Intelligence (AGI) models and their precursors. This framework introduces levels of AGI performance, generality, and autonomy. It is our hope that this framework will be useful in an analogous way to the levels of autonomous driving, by providing a common language to compare models, assess risks, and measure progress along the path to AGI. To develop our framework, we analyze existing definitions of AGI, and distill six principles that a useful ontology for AGI should satisfy. These principles include focusing on capabilities rather than mechanisms; separately evaluating generality and performance; and defining stages along the path toward AGI, rather than focusing on the endpoint. With these principles in mind, we propose "Levels of AGI" based on depth (performance) and breadth (generality) of capabilities, and reflect on how current systems fit into this ontology. We discuss the challenging requirements for future benchmarks that quantify the behavior and capabilities of AGI models against these levels. Finally, we discuss how these levels of AGI interact with deployment considerations such as autonomy and risk, and emphasize the importance of carefully selecting Human-AI Interaction paradigms for responsible and safe deployment of highly capable AI systems.

Keywords: AI, AGI, Artificial General Intelligence, General AI, Human-Level AI, HLAI, ASI, frontier models, benchmarking, metrics, AI safety, AI risk, autonomous systems, Human-AI Interaction

2v2 [cs.AI] 5 Jan 2024

Futuristic | **Skeptical**

Political | **Enterprise**

Deep Network Development

Lecture 2.

Linear Regression & Artificial Neural Networks

Budapest, 21st February 2025

1 Linear Regression

2 Artificial Neural Networks

3 Backpropagation

Meet Tim and Tom

Once childhood friends, but now on different life paths


Tim Gradient (25)
Software Developer at XenuAI,

Stockholm, Sweden

Strong body 💪, sharp mind 🧠, and family at heart 🫶.


Tom Tensor (25)
CEO of ScaliburAI

Zurich, Switzerland

Started young 👦, chasing wealth 💸, and hustling nonstop. ☕

One day they met and…

Tim was mad at Tom for not spending time with him anymore.

Tim, emotionally said: "You're probably going to end up dead alone…"

"No way! I am healthier than you!"

Tom was mad at Tim for living in his comfort zone and not use his full potential.

Tom, emotionally replied: "But you will die first…"

"No way! I am wealthier than you!"

… and that is when they had an idea …

Meet Tim and Tom

TO USE ARTIFICIAL INTELLIGENCE TO PREDICT THEIR LIFE EXPECTANCY!

Supervised Learning: Linear Regression

- Have: (x, y)
  - **x** – data
  - **y** – label
- **Goal: Learn a function to map** $x \rightarrow y$
- Regression – Predict real valued / continuous output: $y \in \mathbb{R}$
  - What is the height of person B?
  - What is going to be the price of a share tomorrow?
  - How many cars are in a city?
  - **What is the life expectancy of person A?**

Life Expectancy function

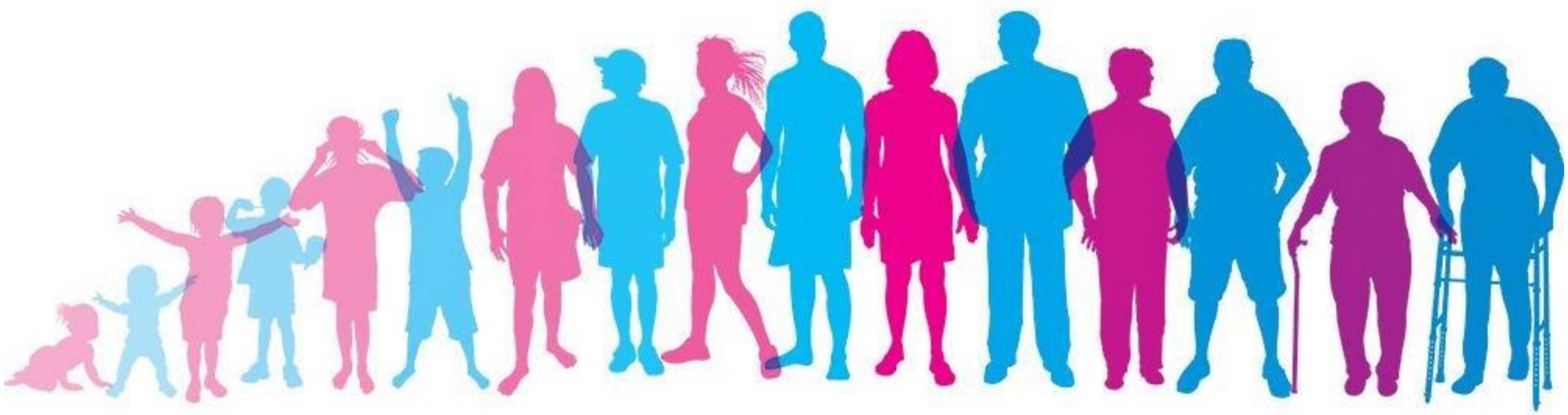
$$f\left(x\right) = y$$

f () =77.6 years

f () =78.1 years

Collect a dataset

- Life Expectancy (WHO) Dataset
- Statistical Analysis on factors influencing Life Expectancy across different countries
- https://www.kaggle.com/kumarajarshi/life-expectancy-who



Exploring the dataset

- Dataset has 2938 data points, each having 20 factors influencing Life Expectancy, the country name and the actual Life Expectancy value

- Adult Mortality - Adult Mortality Rates of both sexes, probability of dying between 15 and 60 years per 1000 population;
- BMI - Average Body Mass Index of entire population;
- GDP - Gross Domestic Product per capita (in USD);
- …

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years              2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Exploring the dataset

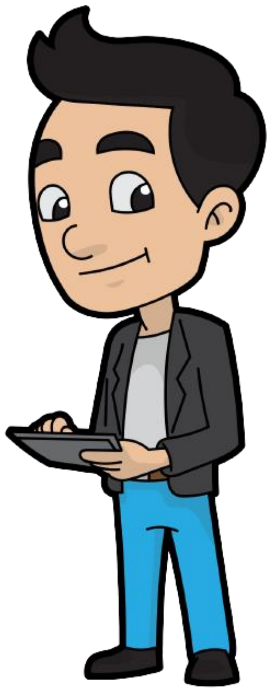
Country	Year	Status	Life expectancy	Adult Mortality	Infant Deaths	Alcohol	Percentage Expenditure	Hepatitis B	Measles	BMI
Hungary	2014	Developed	75.6	137	0	0.01	160.9449		0	64.2

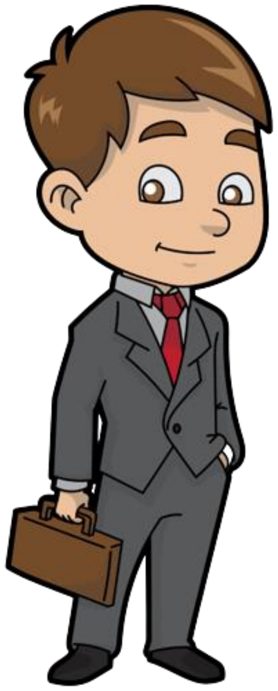
Under-five Deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	Thinness 1-19 years	Thinness 5-9 years	Income composition of resources	Schooling
0	99	7.4	99	0.1	14117.98	9866468	1.7	1.6	0.834	15.8

Feature Selection

Tim chose BMI (independent variable) for predicting Life expectancy (dependent variable)

X = BMI
Y = Life Expectancy





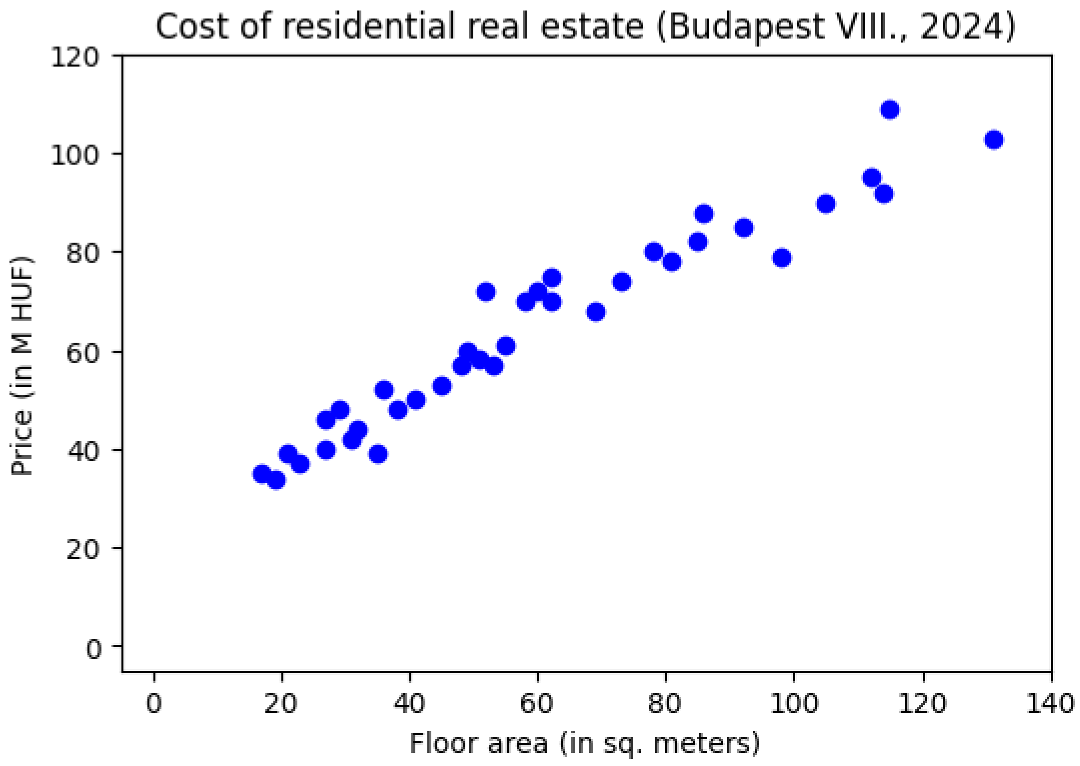
Tom chose GDP (independent variable) for predicting Life expectancy (dependent variable)

X = GDP
Y = Life Expectancy

Example

Cost of residential real estate (Budapest VIII.)

- **x** – (input) floor area (sq. meters)
- **y** – (output) price (in M HUF)

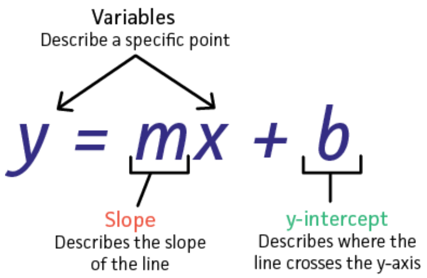


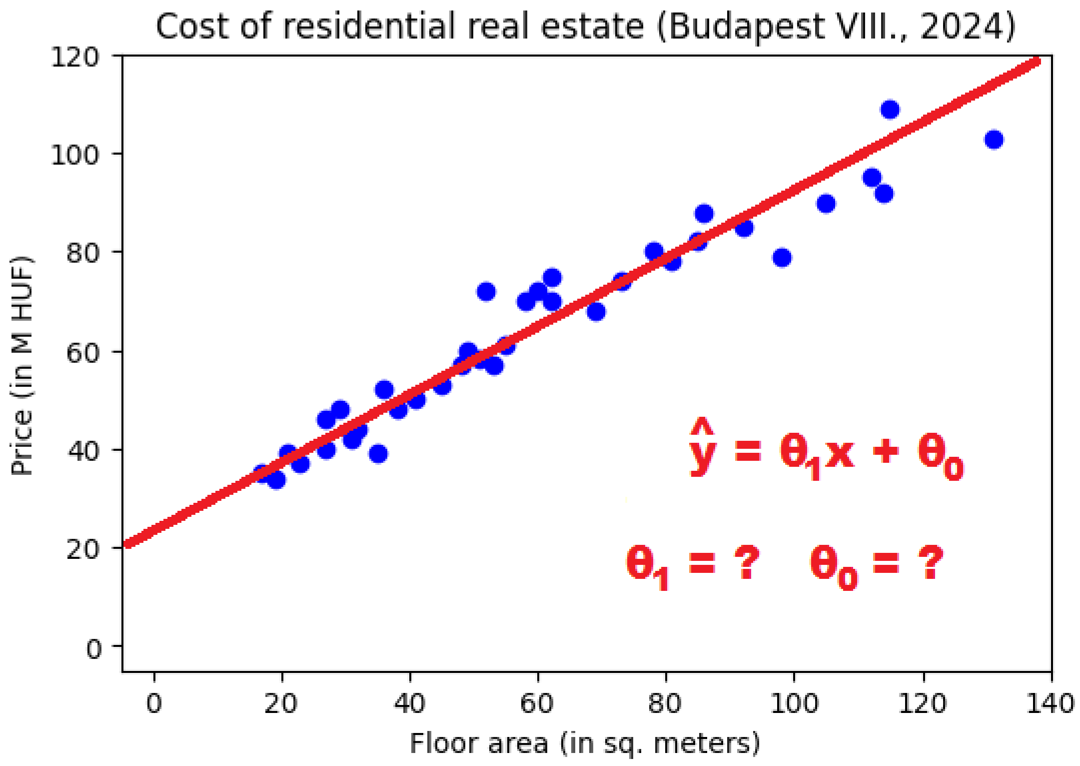
Example

x – (input) number of residents
y – (output) number of cars

Seems to have a linear relationship (we try to fit a line):

$$\hat{y} = \theta_0 + \theta_1 x = \theta x$$





Supervised Learning: Linear Regression

x – (input) floor area (sq. meters)
y – (output) price (in M HUF)

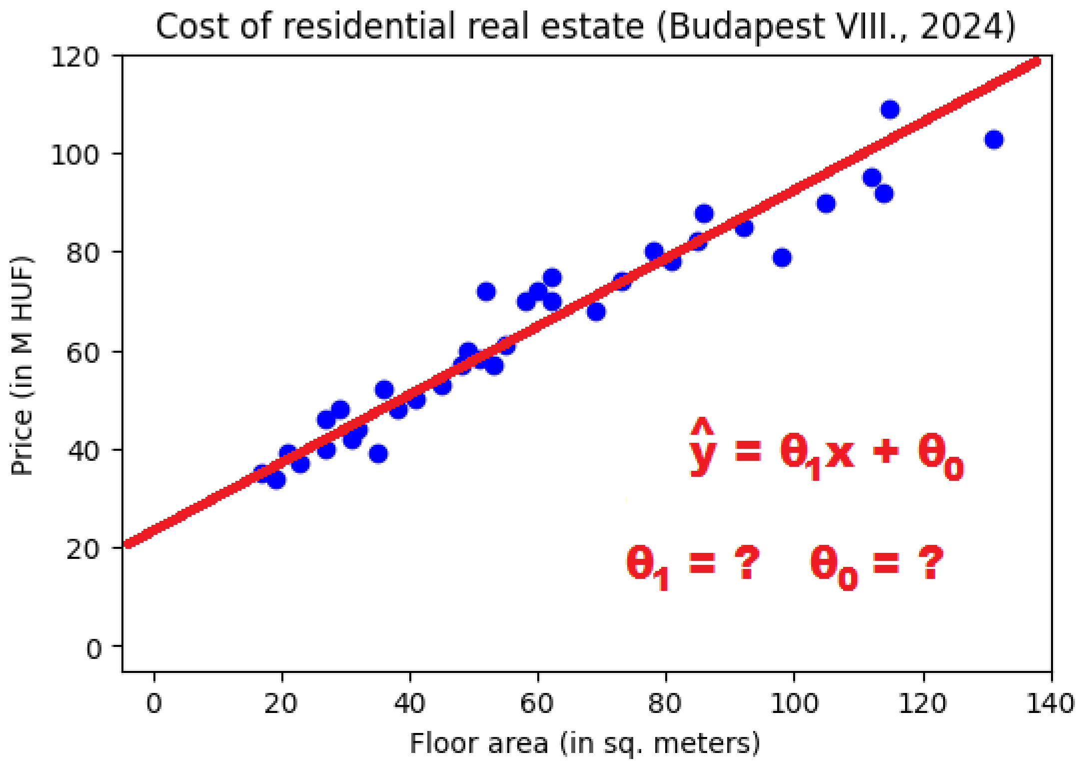
Seems to have a linear relationship. We want to find a function:

$$h(x) = \hat{y} = \theta_0 + \theta_1 x = \theta x$$

To select best, θ that minimize the loss:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta x^{(i)} - y^{(i)}\right)^2$$



Supervised Learning: Linear Regression

x – (input) floor area (sq. meters)
y – (output) price (in M HUF)

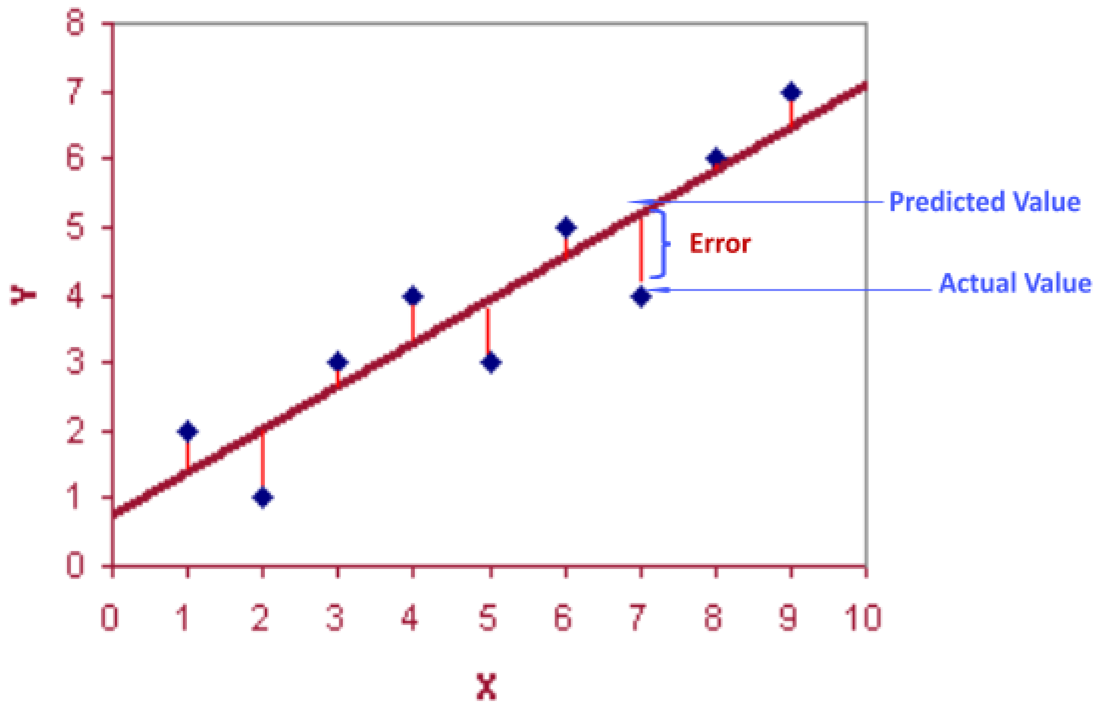
Seems to have a linear relationship. We want to find a function:

$$h\left(x\right) = \hat{y} = \theta_0 + \theta_1 x = \theta x$$

To select best, θ that minimize the loss:

$$J\left(\theta\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$J\left(\theta\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta x^{(i)} - y^{(i)}\right)^2$$

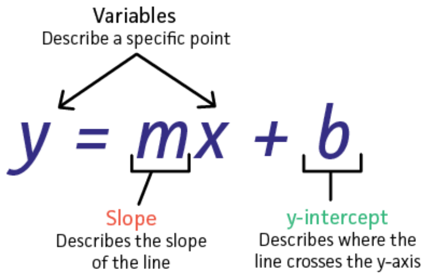


Vectorization

Say we have 200 data points:

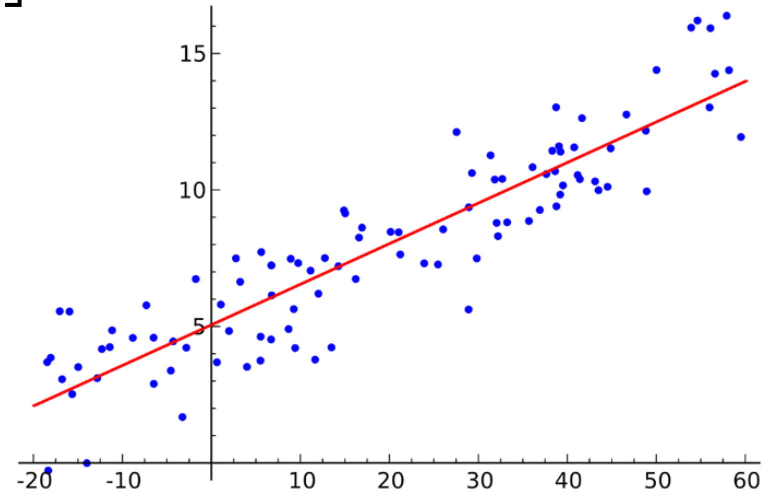
$$\underset{(200,2)}{x} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(200)} \end{bmatrix}, \underset{(200,1)}{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(200)} \end{bmatrix}$$

We want to find $\underset{(2,1)}{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ for $h(x) = \hat{y} = x\theta = \begin{bmatrix} 1 \cdot \theta_0 + x^{(1)}\theta_1 \\ 1 \cdot \theta_0 + x^{(2)}\theta_1 \\ \dots \\ 1 \cdot \theta_0 + x^{(200)}\theta_1 \end{bmatrix}$



That minimizes loss:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_0 + x^{(i)}\theta_1 - y^{(i)}\right)^2$$



(vectorized form): $J(\theta) = \left(\hat{Y} - Y\right)^T\left(\hat{Y} - Y\right) = (X\theta - Y)^T(X\theta - Y)$

Supervised Learning: Linear Regression

The problem can be summarized:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

Find the optimal θ^* that minimizes the loss (in this case the Mean Squared Error):

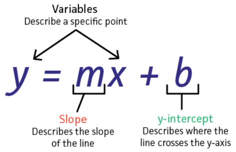
$$J(\theta^*) = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}\theta^* - y^{(i)}\right)^2$$

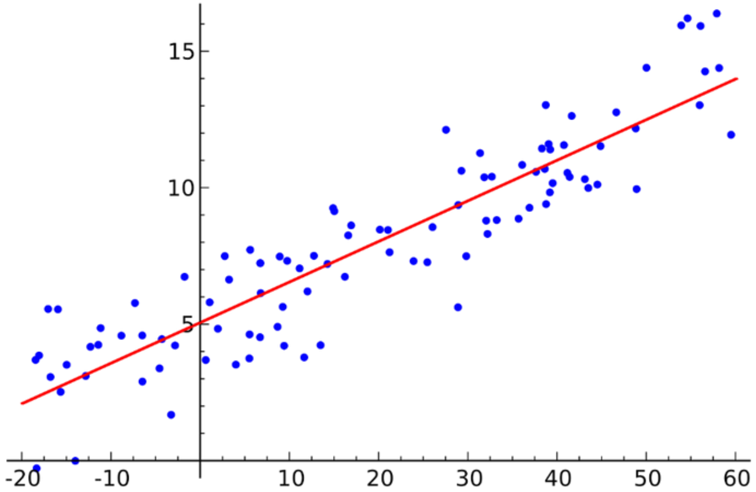
How to find optimal θ^* ?

- Analytical solution: Normal equation
- Numerical solution: Gradient Descent

Analytical solution: Normal Equation

Say we have 200 data points:





$$x = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \dots & \dots \\ 1 & x^{(200)} \end{bmatrix}$$

$$\theta = [\theta_0, \theta_1]^T$$

$$\mathbf{y} = [y^{(1)}, y^{(2)}, \dots, y^{(200)}]^T$$

$$\hat{\mathbf{y}} = X\theta$$

$$J(\theta) = (X\theta - Y)^T(X\theta - Y)$$

Normal Equation [1]: $\theta^* = (X^T X)^{-1}(X^T Y)$

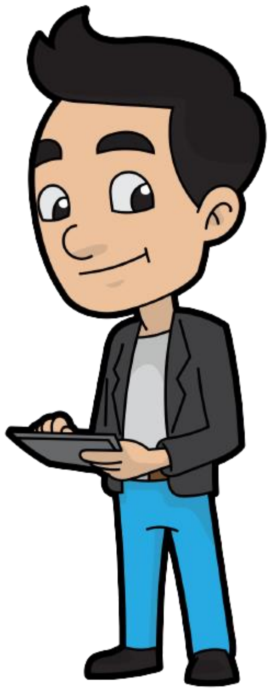
θ^* is the optimal parameter that minimizes the loss function $J(\theta)$

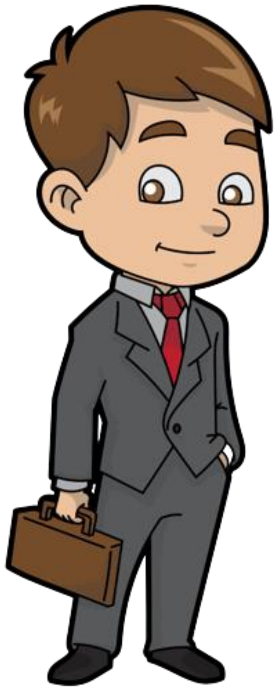
[1] Normal equation derivation (different): https://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression/

Feature selection (recap)

Tim chose BMI (independent variable) for predicting Life expectancy (dependent variable)

X = BMI
Y = Life Expectancy

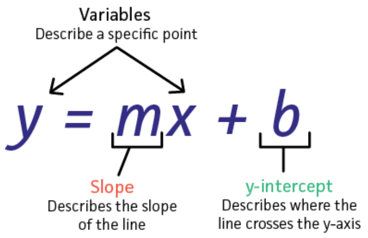




Tom chose GDP (independent variable) for predicting Life expectancy (dependent variable)

X = GDP
Y = Life Expectancy

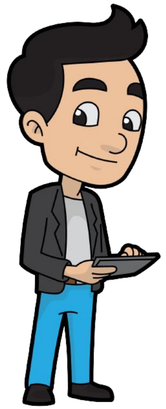
Analytical solution: Normal Equation (Tim)



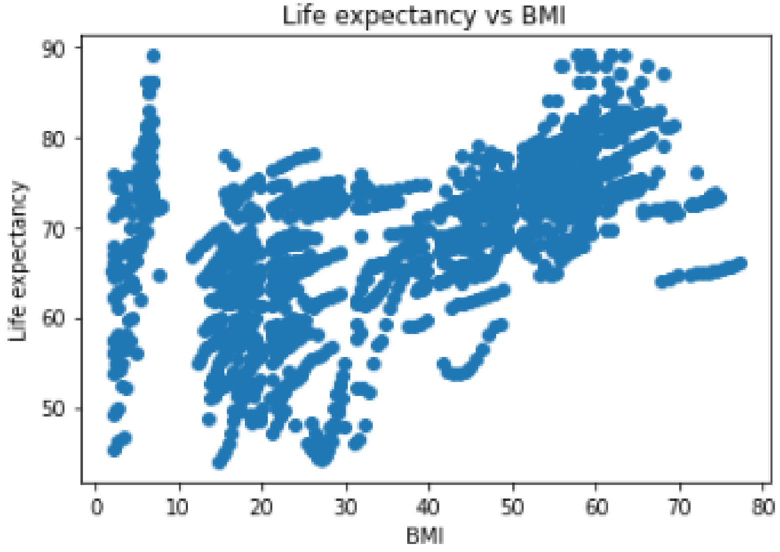
$$\theta^* = (X^T X)^{-1}(X^T Y)$$

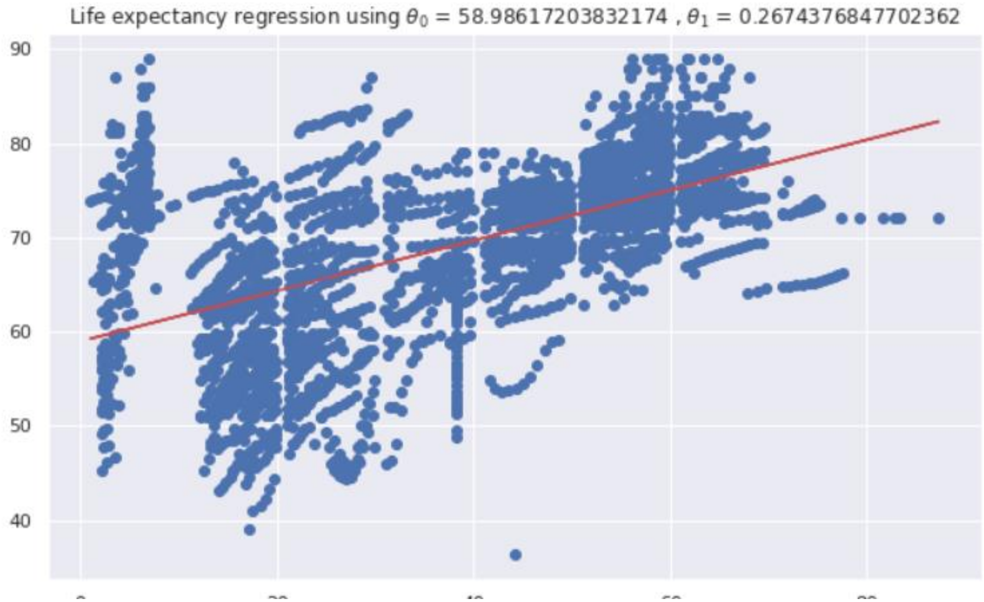
```
def normal_equation(X,Y):
    return np.linalg.inv(X.T @ X) @ (X.T @ Y)

tim_theta = normal_equation(tim_data_ready, Y)
print(f'y = mx + b')
print(f'y = {tim_theta[1]}x + {tim_theta[0]}')
```

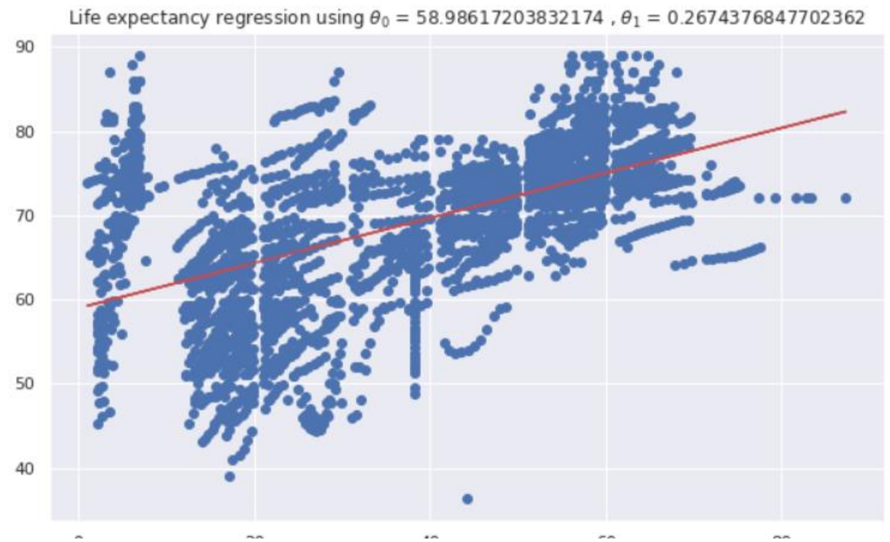


$$y = mx + b$$
$$y = 0.26743768477023727x + 58.98617203832164$$





Analytical solution: Normal Equation (Tim)




```python
def life_expectancy(X, theta):
    X = np.concatenate((np.ones(1),np.array(X)),axis=0)
    return round(np.dot(X, theta),1)

height = 1.82 #float(input("Please input your height (in meters): ")) #1.82
weight = 80 #float(input("Please input your weight (in kilograms): ")) #80
bmi = weight / height**2
print("BMI:",bmi)
life_exp_tim = life_expectancy([bmi], tim_theta)
print("Tim's life expectancy is", life_exp_tim , "years.")
```

BMI: 24.151672503320853

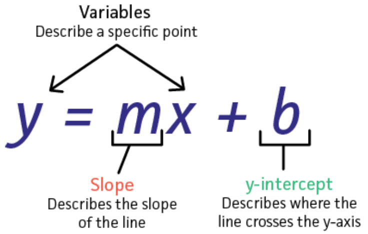
Tim's life expectancy is 65.4 years.

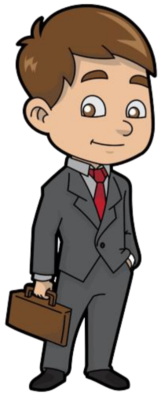
```python
height = 1.80 #float(input("Please input your height (in meters): ")) #1.82
weight = 73 #float(input("Please input your weight (in kilograms): ")) #80
bmi = weight / height**2
print("BMI:",bmi)
life_exp_tim_tom = life_expectancy([bmi], tim_theta)
print("Tom's life expectancy predicted by Tim's model is", life_exp_tim_tom , "years.")
```

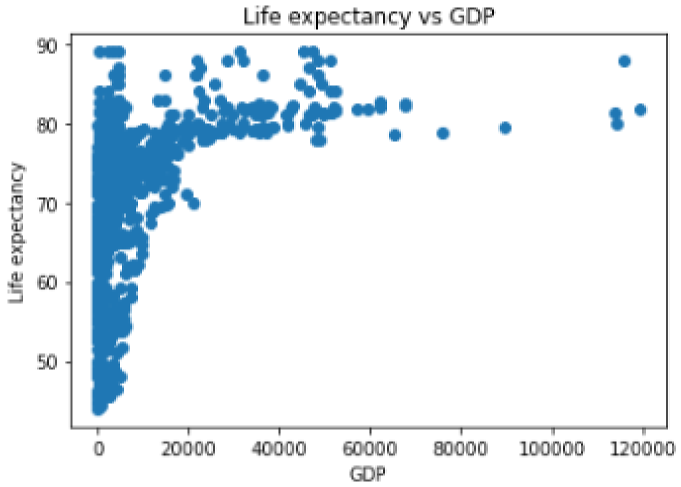
BMI: 22.530864197530864

Tom's life expectancy is 65.0 years.

Analytical solution: Normal Equation (Tom)

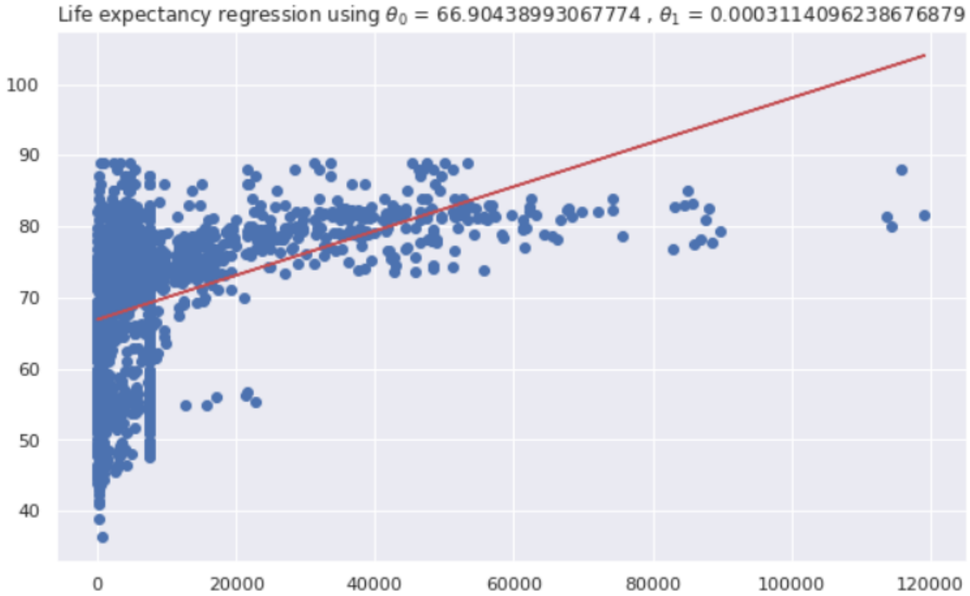




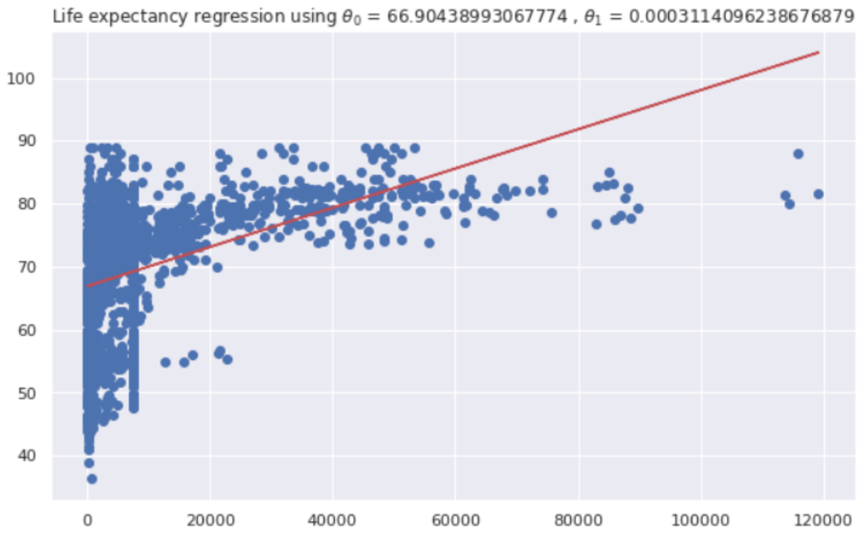



```
tom_theta = normal_equation(tom_data_ready, Y)
print(f'y = mx + b')
print(f'y = {tom_theta[1]}x + {tom_theta[0]}')
```

$$y = mx + b$$
$$y = 0.0003114096238676778x + 66.90438993067785$$



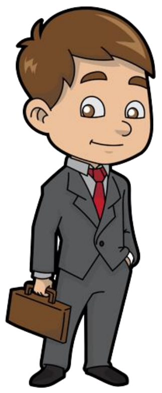
Analytical solution: Normal Equation (Tom)




```
def life_expectancy(X, theta):
    X = np.concatenate((np.ones(1),np.array(X)),axis=0)
    return round(np.dot(X, theta),1)

gdp = 8500 #float(input("Please input the GDP of your country: ")) #8500
life_exp_tom = life_expectancy([gdp], tom_theta)
```

Tom's life expectancy is 69.6 years.



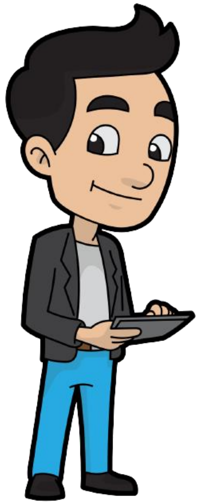
```
gdp = 5000 #float(input("Please input the GDP of your country: ")) #5000
life_exp_tom_tim = life_expectancy([gdp], tom_theta)
print("Tim's life expectancy predicted by Tom is", life_exp_tom_tim , "years.")
```

Tim's life expectancy predicted by Tom is 68.5 years.

Results

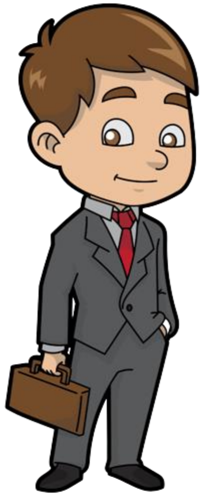
Tim's Life Expectancy predictions:
- Tim – 65.4 years
- Tom – 65 years

Tom's Life Expectancy predictions:
- Tim – 68.5 years
- Tom – 69.6 years



NO AGREEMENT!!

- Metrics?
- Further explore dataset?
- Better feature selection?



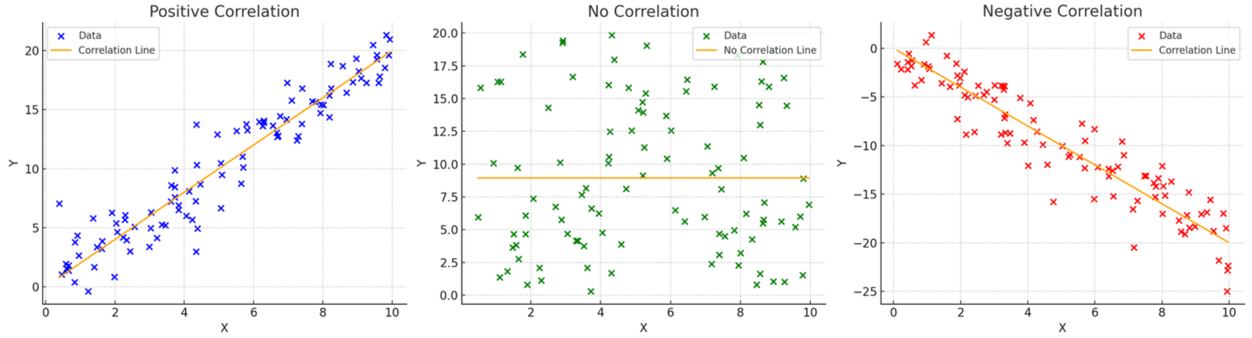
My model is better!
I'll live longer!

NO! Mine is better!
I'll live longer!

Exploring the dataset part 2

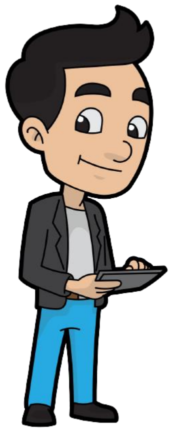
Correlation matrix

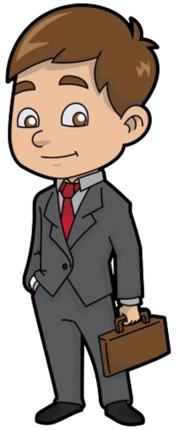
- What is the relationship between the different features and the Life expectancy?

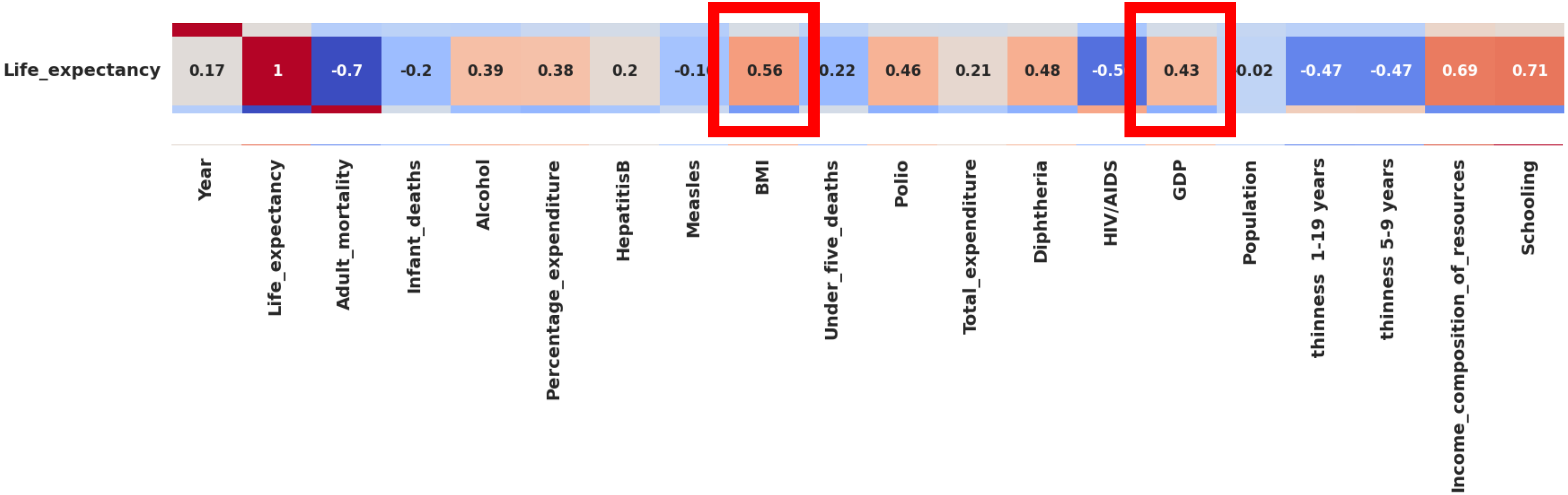


	Year	Life_expectancy	Adult_mortality	Infant_deaths	Alcohol	Percentage_expenditure	HepatitisB	Measles	BMI	Under_five_deaths	Polio	Total_expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income_composition_of_resources	Schooling
Year	1	0.17	-0.08	-0.04	-0.05	0.03	0.09	-0.08	0.11	-0.04	0.09	0.08	0.13	-0.14	0.09	0.01	-0.05	-0.05	0.24	0.2
Life_expectancy	0.17	1	-0.7	-0.2	0.39	0.38	0.2	-0.16	0.56	-0.22	0.46	0.21	0.48	-0.56	0.43	-0.02	-0.47	-0.47	0.69	0.71
Adult_mortality	-0.08	-0.7	1	0.08	-0.19	-0.24	-0.14	0.03	-0.38	0.09	-0.27	-0.11	-0.27	0.52	-0.28	-0.01	0.3	0.31	-0.44	-0.44
Infant_deaths	-0.04	-0.2	0.08	1	-0.11	-0.09	-0.18	0.5	-0.23	1	-0.17	-0.13	-0.18	0.03	-0.11	0.55	0.47	0.47	-0.14	-0.19
Alcohol	-0.05	0.39	-0.19	-0.11	1	0.34	0.08	-0.05	0.32	-0.11	0.21	0.29	0.22	-0.05	0.32	-0.03	-0.42	-0.41	0.42	0.5
Percentage_expenditure	0.03	0.38	-0.24	-0.09	0.34	1	0.01	-0.06	0.23	-0.09	0.15	0.17	0.14	-0.1	0.89	-0.02	-0.25	-0.25	0.38	0.39
HepatitisB	0.09	0.2	-0.14	-0.18	0.08	0.01	1	-0.09	0.13	-0.18	0.41	0.05	0.5	-0.1	0.06	-0.11	-0.11	-0.11	0.15	0.17
Measles	-0.08	-0.16	0.03	0.5	-0.05	-0.06	-0.09	1	-0.18	0.51	-0.14	-0.1	-0.14	0.03	-0.07	0.24	0.22	0.22	-0.12	-0.12
BMI	0.11	0.56	-0.38	-0.23	0.32	0.23	0.13	-0.18	1	-0.24	0.28	0.23	0.28	-0.24	0.28	-0.06	-0.53	-0.54	0.48	0.51
Under_five_deaths	-0.04	-0.22	0.09	1	-0.11	-0.09	-0.18	0.51	-0.24	1	-0.19	-0.13	-0.2	0.04	-0.11	0.54	0.47	0.47	-0.16	-0.21
Polio	0.09	0.46	-0.27	-0.17	0.21	0.15	0.41	-0.14	0.28	-0.19	1	0.13	0.67	-0.16	0.19	-0.03	-0.22	-0.22	0.36	0.39
Total_expenditure	0.08	0.21	-0.11	-0.13	0.29	0.17	0.05	-0.1	0.23	-0.13	0.13	1	0.15	-0	0.12	-0.07	-0.27	-0.28	0.15	0.22
Diphtheria	0.13	0.48	-0.27	-0.18	0.22	0.14	0.5	-0.14	0.28	-0.2	0.67	0.15	1	-0.16	0.18	-0.03	-0.23	-0.22	0.37	0.39
HIV/AIDS	-0.14	-0.56	0.52	0.03	-0.05	-0.1	-0.1	0.03	-0.24	0.04	-0.16	-0	-0.16	1	-0.13	-0.03	0.2	0.21	-0.25	-0.22
GDP	0.09	0.43	-0.28	-0.11	0.32	0.89	0.06	-0.07	0.28	-0.11	0.19	0.12	0.18	-0.13	1	-0.03	-0.27	-0.27	0.44	0.43
Population	0.01	-0.02	-0.01	0.55	-0.03	-0.02	-0.11	0.24	-0.06	0.54	-0.03	-0.07	-0.03	-0.03	-0.03	1	0.24	0.23	-0.01	-0.03
thinness 1-19 years	-0.05	-0.47	0.3	0.47	-0.42	-0.25	-0.11	0.22	-0.53	0.47	-0.22	-0.27	-0.23	0.2	-0.27	0.24	1	0.94	-0.41	-0.45
thinness 5-9 years	-0.05	-0.47	0.31	0.47	-0.41	-0.25	-0.11	0.22	-0.54	0.47	-0.22	-0.28	-0.22	0.21	-0.27	0.23	0.94	1	-0.4	-0.44
Income_composition_of_resources	0.24	0.69	-0.44	-0.14	0.42	0.38	0.15	-0.12	0.48	-0.16	0.36	0.15	0.37	-0.25	0.44	-0.01	-0.41	-0.4	1	0.8
Schooling	0.2	0.71	-0.44	-0.19	0.5	0.39	0.17	-0.12	0.51	-0.21	0.39	0.22	0.39	-0.22	0.43	-0.03	-0.45	-0.44	0.8	1

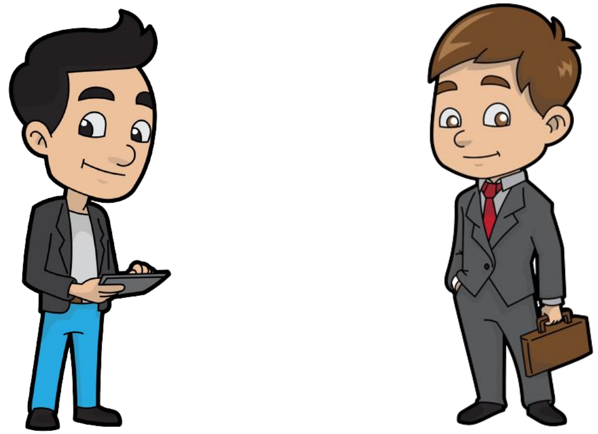
Exploring the dataset part 2

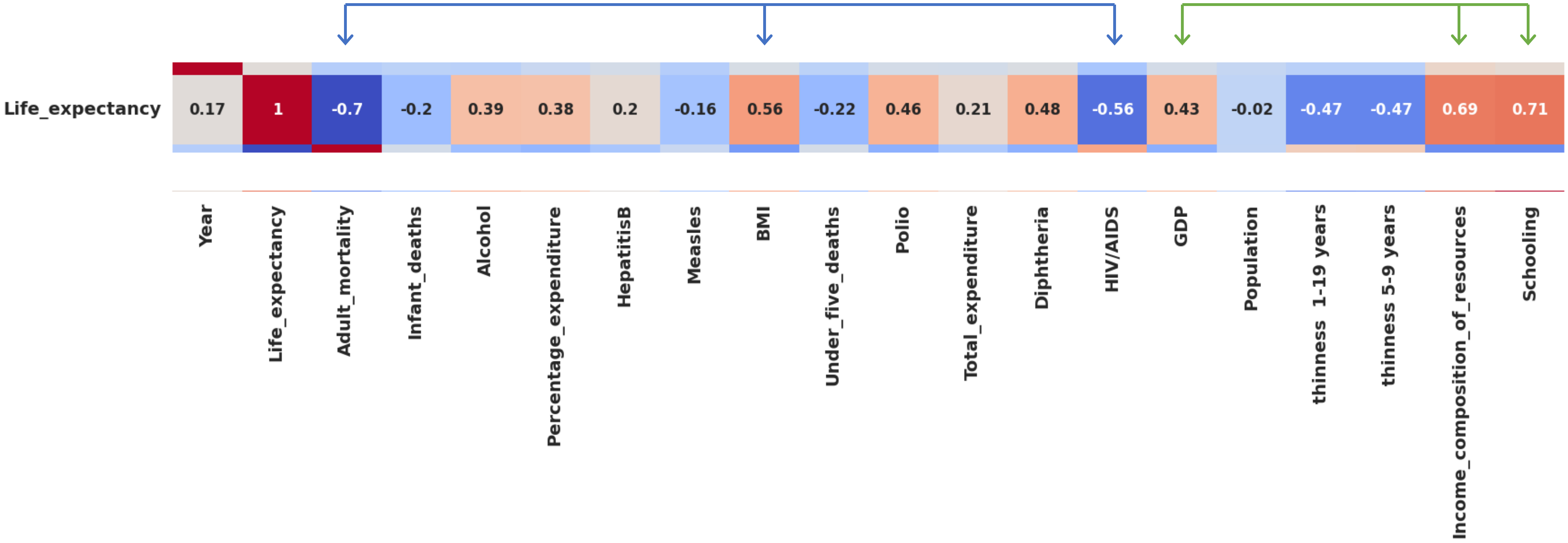






Exploring the dataset part 2

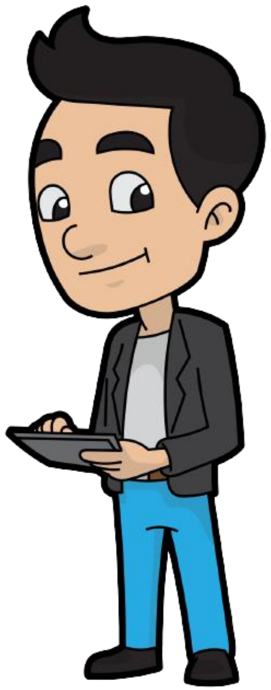


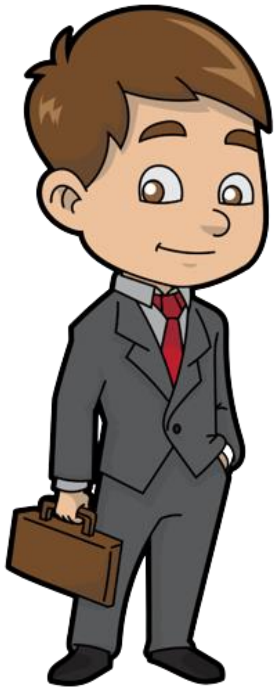


Exploring the dataset part 2

Tim chose BMI, Adult Mortality and HIV/AIDS (independent variables) for predicting Life expectancy (dependent variable)

X = BMI, Adult Mortality, HIV/AIDS
Y = Life Expectancy





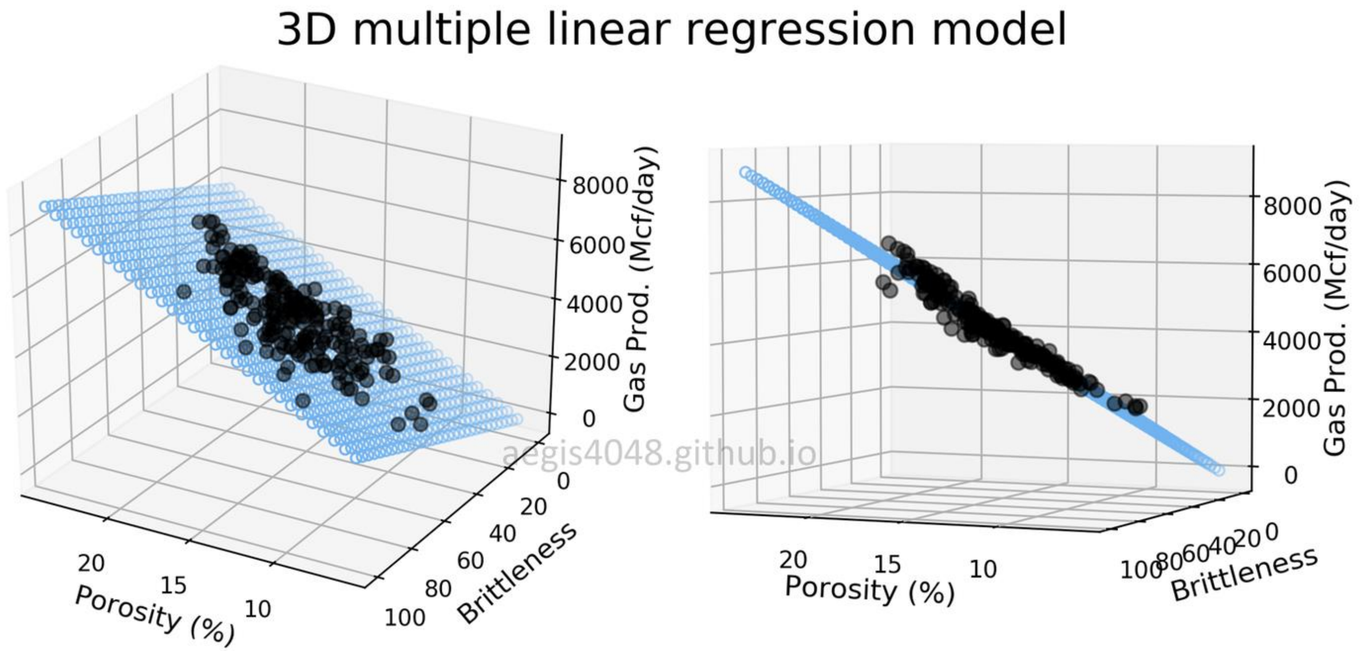
Tom chose GDP, Income composition of resources and Schooling (independent variables) for predicting Life expectancy (dependent variable)

X = GDP, Income comp. resources, Schooling
Y = Life Expectancy

Multiple Linear Regression

- How many cars are in a city?
- x_1 – (input) number of residents (in thousands)
- x_2 – (input) distance to capital (in km)
- y – (output) number of cars

- We can add more… $x_3, \ldots, x_n$



Multiple Linear Regression (Vectorization)

- Similar (only dimensions change)
- Generalizable
- **n** features, +1 column with ones

$$\underset{(m,n+1)}{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & \dots & x_n^{(2)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \underset{(n+1,1)}{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix} \quad \underset{(m,1)}{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

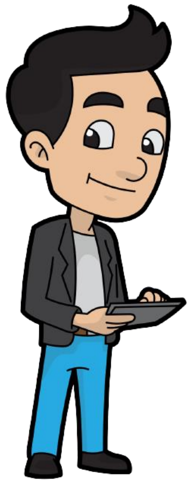
$$\underset{(m,1)}{y} = \underset{(m,n+1)}{X}\underset{(n+1,1)}{\theta}$$

Multiple Linear Regression: Normal equation (Tim)

```
tim_theta = normal_equation(tim_data_ready, Y)
print(f'theta = {tim_theta}')
```

```
theta = [ 6.95250308e+01  1.58686254e-01 -3.38199840e-02 -4.58041192e-01]
```

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2}$$



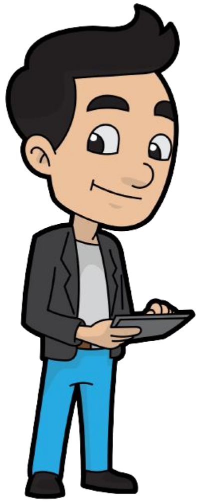
```
def RMSE(y_pred, y):
    return np.sqrt(metrics.mean_squared_error(y, y_pred))

tim_y_pred = np.dot(tim_data_ready, tim_theta)
tim_rmse = RMSE(tim_y_pred, Y)

print("Tim's RMSE: ", tim_rmse)
```

```
Tim's RMSE:  5.784750918898876
```

Multiple Linear Regression: Normal equation (Tim)



```
height = 1.82 #float(input("Please input your height (in meters): "))
weight = 80 #float(input("Please input your weight (in kilograms): "))
bmi = weight / height**2
print("BMI:",bmi)
adult_mortality = 53
hiv = 0.1
life_exp_tim = life_expectancy([bmi,adult_mortality,hiv], tim_theta)
print("Tim's life expectancy is", life_exp_tim , "years.")
```

```
BMI: 24.151672503320853
Tim's life expectancy is 71.5 years.
```

```
height = 1.80 #float(input("Please input your height (in meters): "))
weight = 73 #float(input("Please input your weight (in kilograms): "))
bmi = weight / height**2
print("BMI:",bmi)
adult_mortality = 70
hiv = 0.1
life_exp_tim_tom = life_expectancy([bmi,adult_mortality,hiv], tim_theta)
print("Tom's life expectancy predicted by Tim is", life_exp_tim_tom , "years.")
```

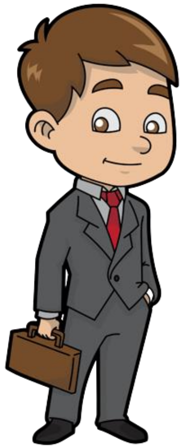
```
BMI: 22.530864197530864
Tom's life expectancy predicted by Tim is 70.7 years.
```

Multiple Linear Regression: Normal Equation (Tom)

```
tom_theta = normal_equation(tom_data_ready, Y)
print(f'theta = {tom_theta}')
```

```
theta = [4.49681347e+01 8.26882411e-05 1.42744076e+01 1.22489350e+00]
```

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2}$$



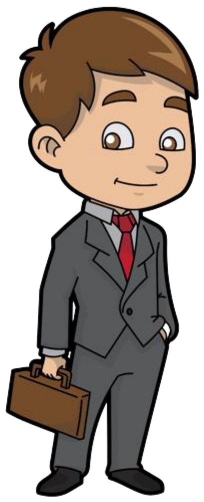
```
def RMSE(y_pred, y):
    return np.sqrt(metrics.mean_squared_error(y, y_pred))

tom_y_pred = np.dot(tom_data_ready, tom_theta)
tom_rmse = RMSE(tom_y_pred, Y)

print("Tom's RMSE: ", tom_rmse)
```

```
Tom's RMSE:  6.288890108092894
```

Multiple Linear Regression: Normal equation (Tom)



```
gdp = 8500
income = 0.8
school = 16.5
life_exp_tom = life_expectancy([gdp,income,school], tom_theta)
print("Tom's life expectancy is", life_exp_tom , "years.")
```

```
Tom's life expectancy is 77.3 years.
```

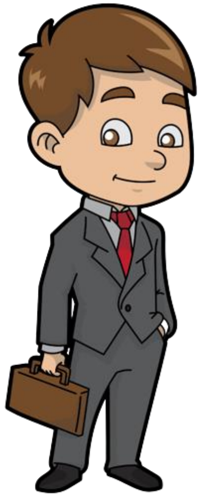
```
gdp = 5000
income = 0.77
school = 15.5
life_exp_tom_tim = life_expectancy([gdp,income,school], tom_theta)
print("Tim's life expectancy predicted by Tom is", life_exp_tom_tim , "years.")
```

```
Tim's life expectancy predicted by Tom is 75.4 years.
```

Results part 2

Tim’s Life Expectancy predictions:
- Tim – 71.5 years
- Tom – 70.7 years

Tim’s RMSE: 5.79

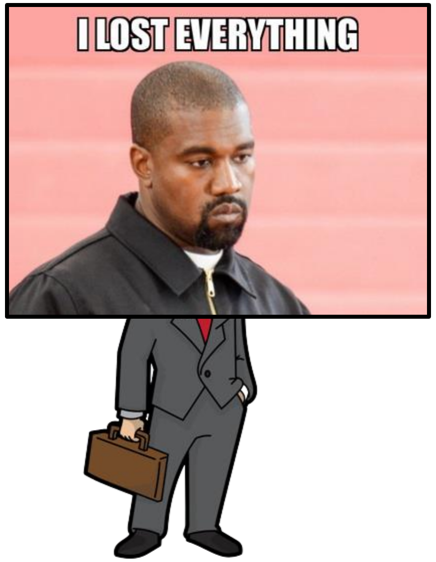


Tom’s Life Expectancy predictions:
- Tom – 77.3 years
- Tim – 75.4 years

Tim’s RMSE: 6.29

Next steps...

- Normalization? Standardization?
- Gradient Descent?
- Artificial Neural Networks?



Tim vs Tom to be continued...

Normalization vs Standardization

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

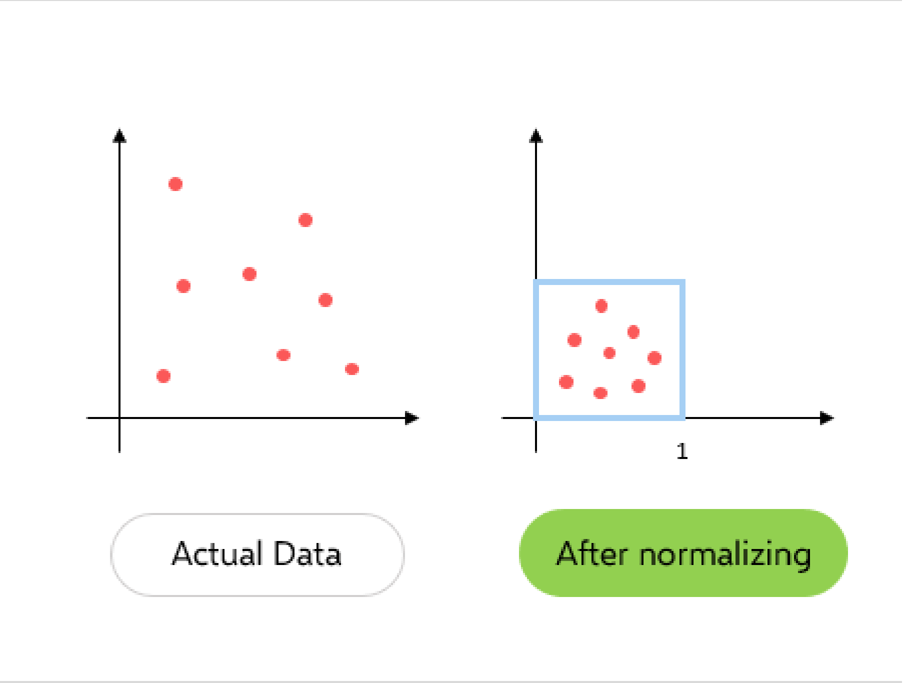
Normalization

Consider a dataset containing two features:

- Age
- Income

The range of age can go from 0-100 years old,
whereas the range of income can go from 0-1,000,000 HUF

If we would do linear regression, income would have the highest influence because of its large values.



https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

Normalization vs Standardization

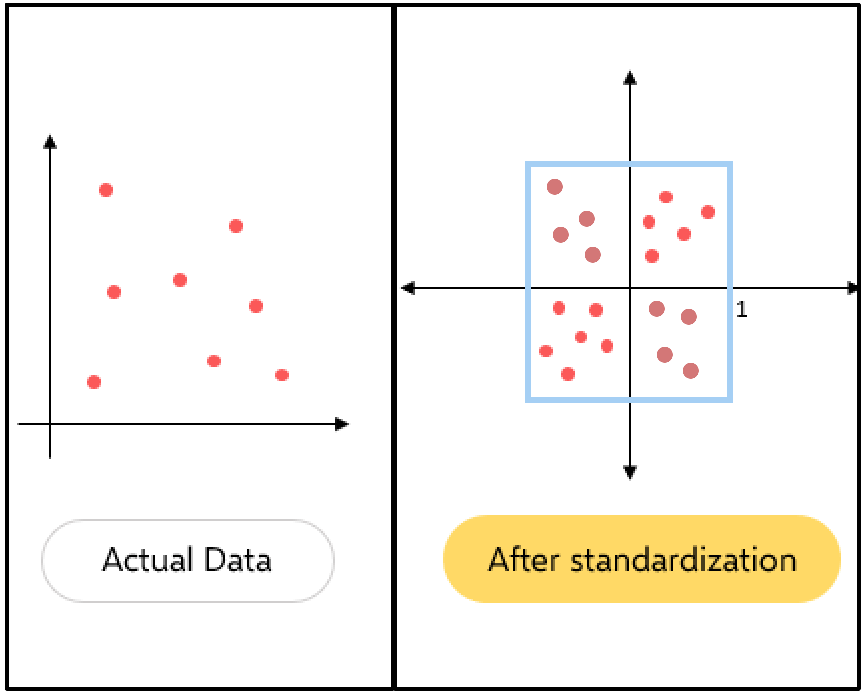
$$z = \frac{x - \bar{x}}{\sigma}$$

Standardization

Consider a dataset containing two features:

- Weight (Kg)
- Height (cm)

The range might be similar, but the meaning is different. For example, 100 Kg is different than 100 cm.
Most importantly, the deviation and the mean are different for both attributes.
For instance, it is more common to have people with a mean of 80 Kg and a few people with 100 or 200 Kg, than having people with a mean of 80 cm tall (the majority will have more 160 cm).

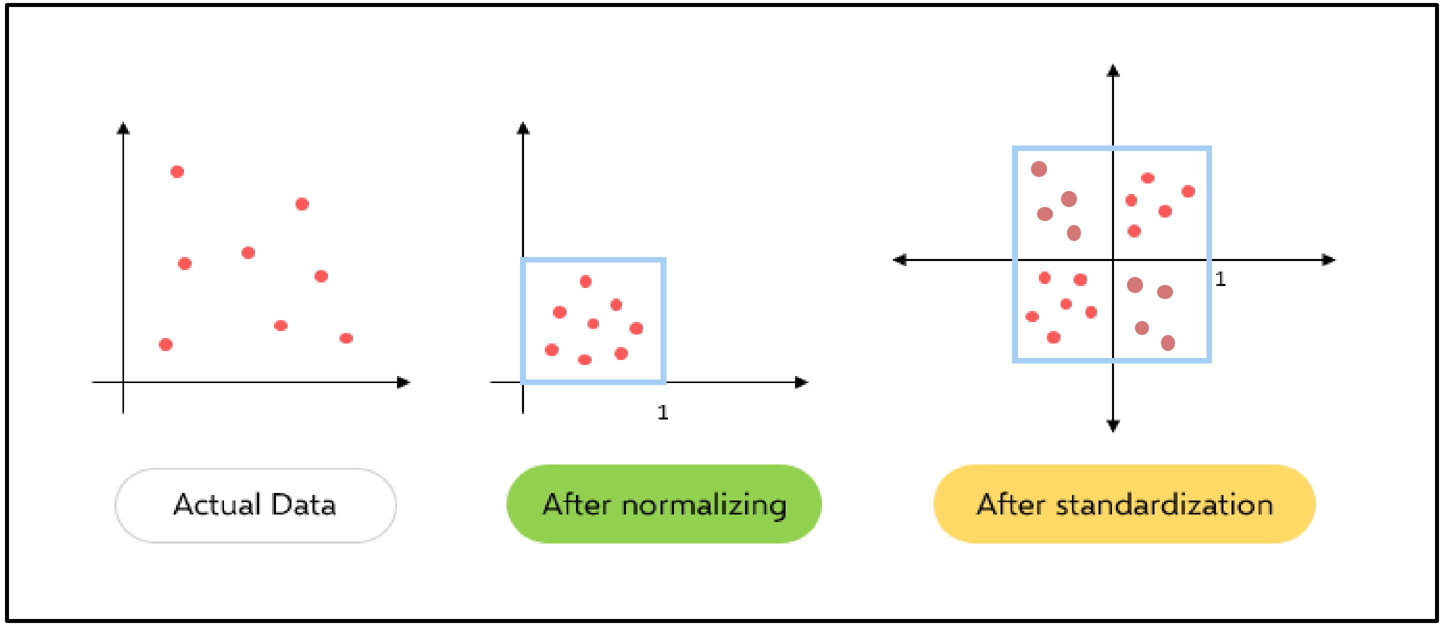


https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

Normalization vs Standardization

Normalization is recommended when your data has varying scales.
Standardization is recommended when your data has different units.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad z = \frac{x - \bar{x}}{\sigma}$$

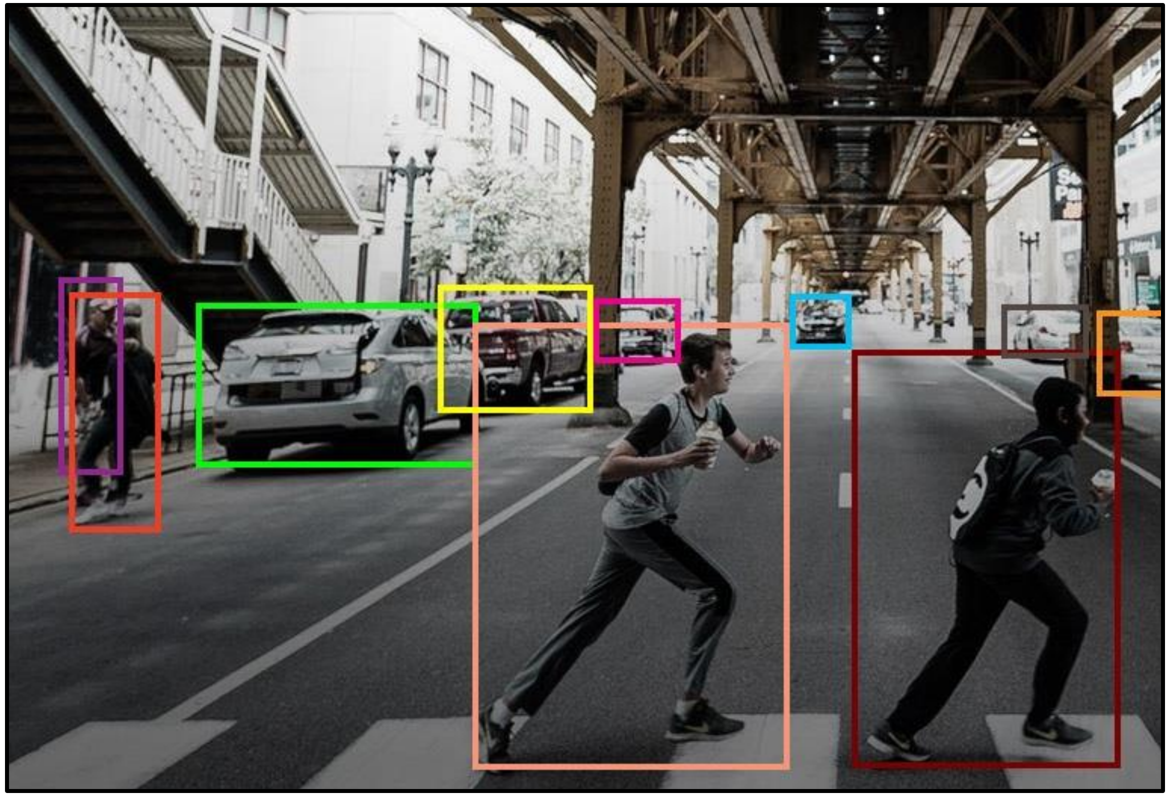


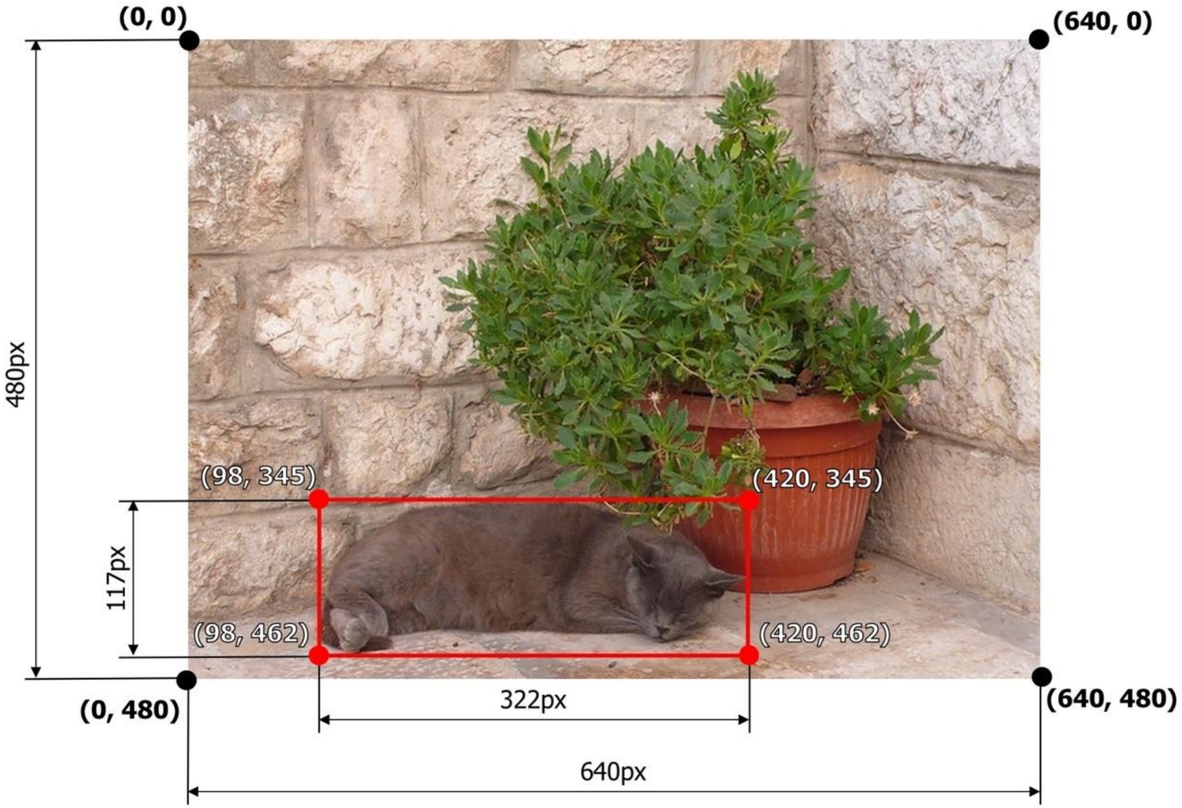
Normalisation	Standardisation
Scaling is done by the highest and the lowest values.	Scaling is done by mean and standard deviation.
It is applied when the features are of separate scales.	It is applied when we verify zero mean and unit standard deviation.
Scales range from 0 to 1	Not bounded
Affected by outliers	Less affected by outliers
It is applied when we are not sure about the data distribution	It is used when the data is Gaussian or normally distributed
It is also known as Scaling Normalization	It is also known as Z-Score

https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff

Applications

Object Detection (Bounding Box Regression)





Deep Network Development

Lecture 2.

Linear Regression & Artificial Neural Networks

Budapest, 21st February 2025

1 Linear Regression

2 Artificial Neural Networks

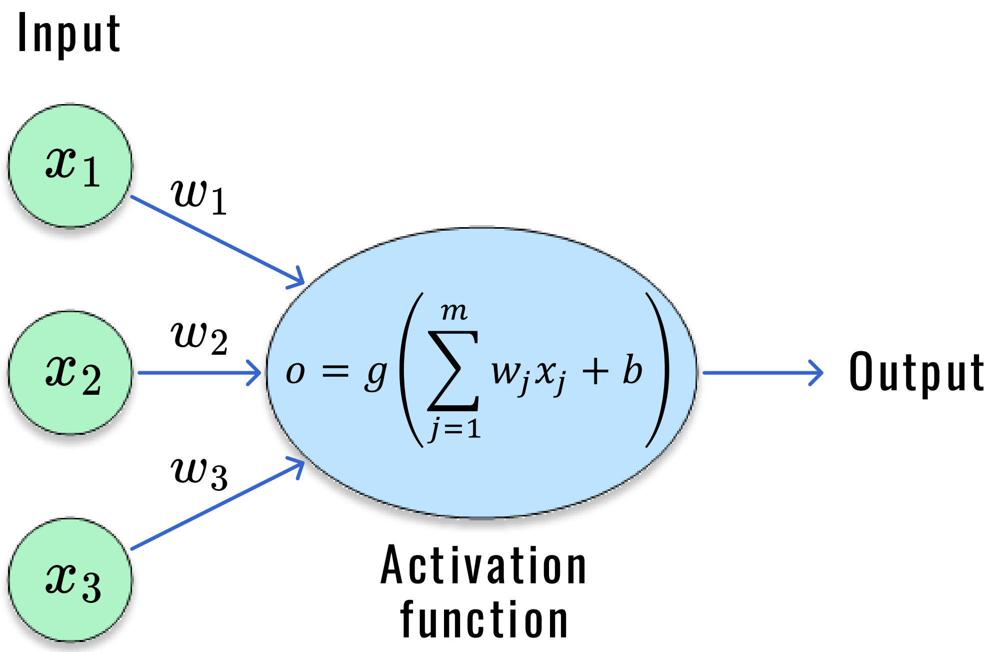
3 Backpropagation

How do ANNs work?

x – the inputs
w – weight parameters we will train
b – bias parameter we will train
g – nonlinear activation function
(ReLU, Softmax, ...)
o – the output

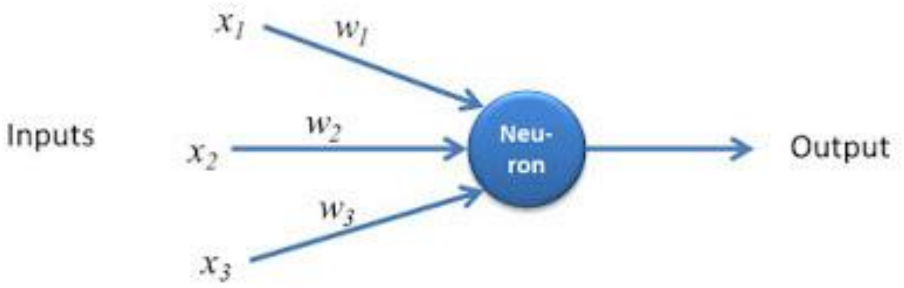
One neuron with *m* inputs does the following:

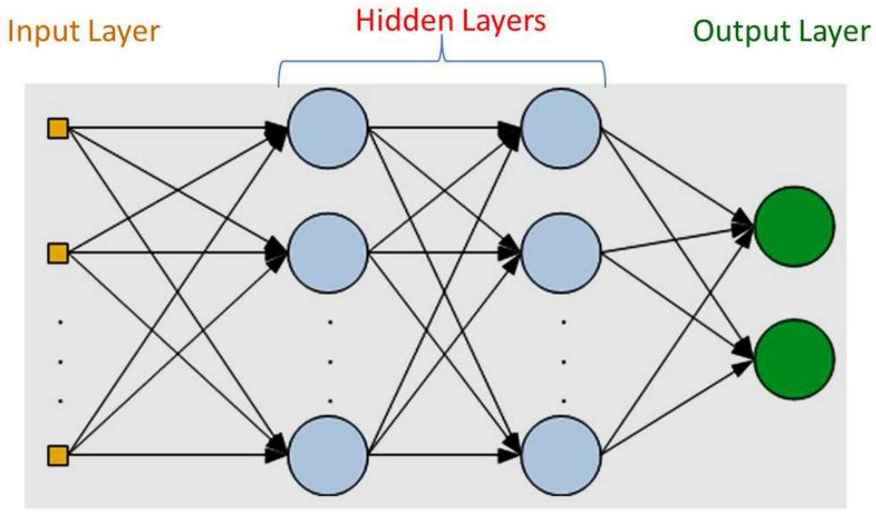
$$o = g\left(\sum_{j=1}^{m} w_j x_j + b\right)$$



Deep Neural Networks

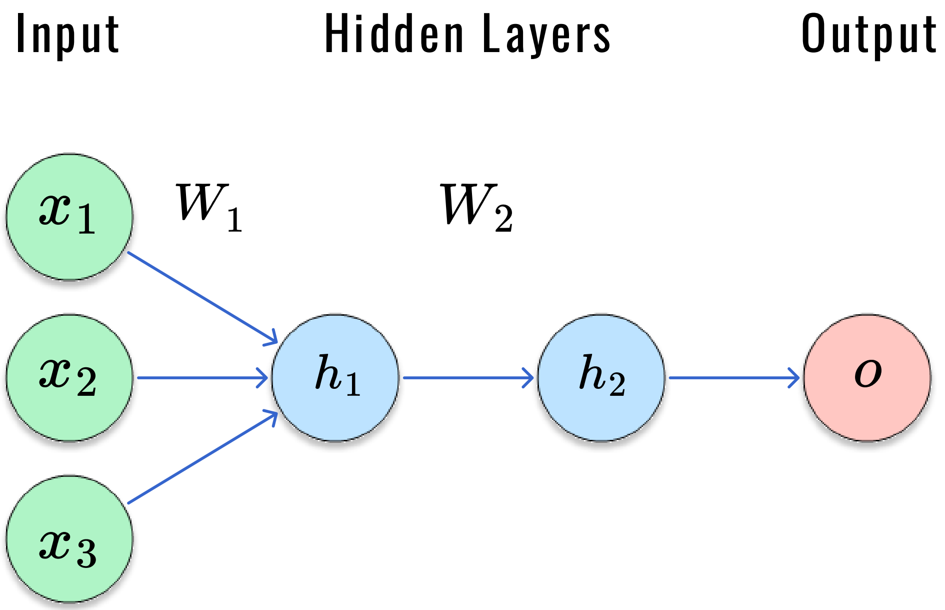
- Neural networks are built up from neurons, that have inputs and outputs
- Neurons are organised into layers
- Layers refine the output of the previous layers





Why do we need nonlinear activation functions?

What happens if we create a deep neural network with 2 hidden layers without an activation function?



(vectorized)

$$g(x) = x$$

$$h_1(x) = g\left(W_1 x + b_1\right) = W_1 x + b_1$$

$$h_2(x) = g\left(W_2 x + b_2\right) = W_2 x + b_2$$

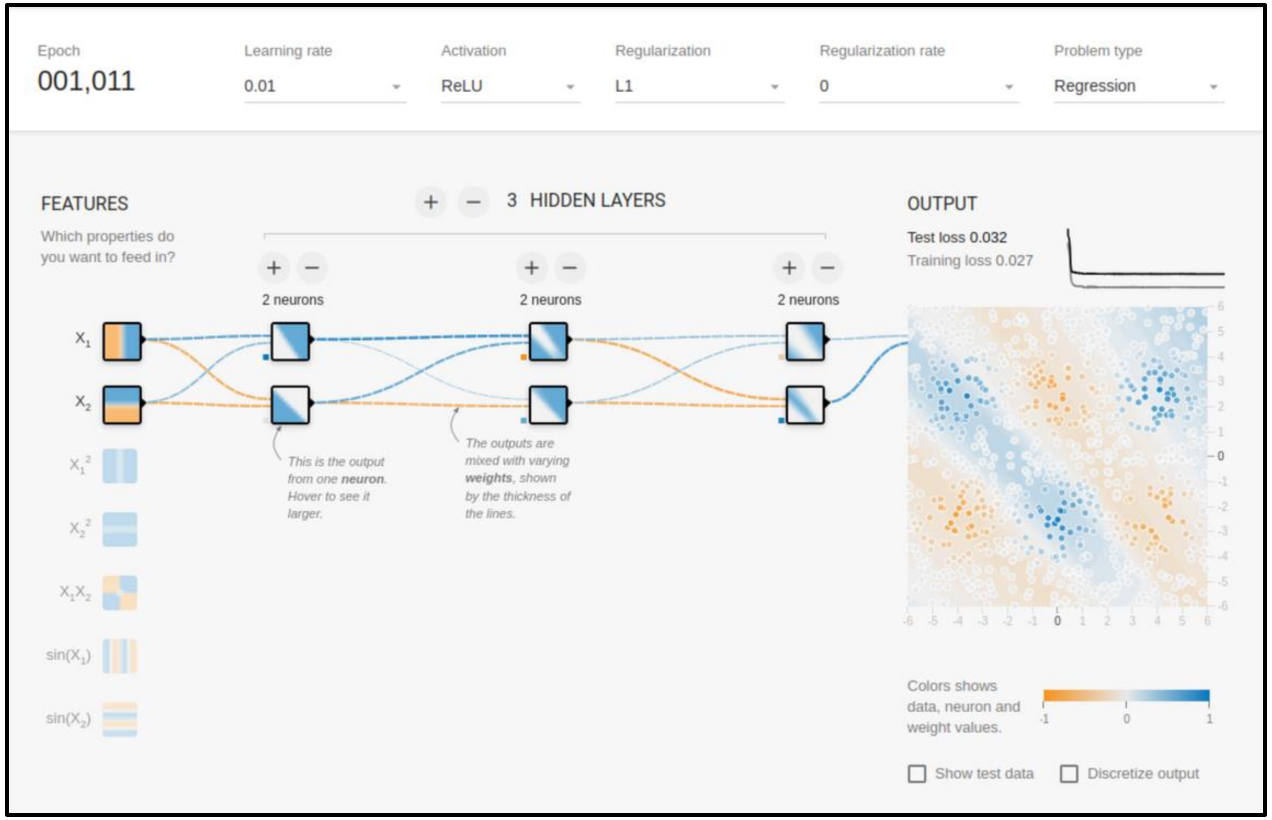
$$\begin{aligned} o = h_2(h_1(x)) &= h_2(W_1 x + b_1) = \\ &= W_2(W_1 x + b_1) + b_2 = \\ &= W_2 W_1 x + W_2 b_1 + b_2 \end{aligned}$$

we get a bigger linear regression model

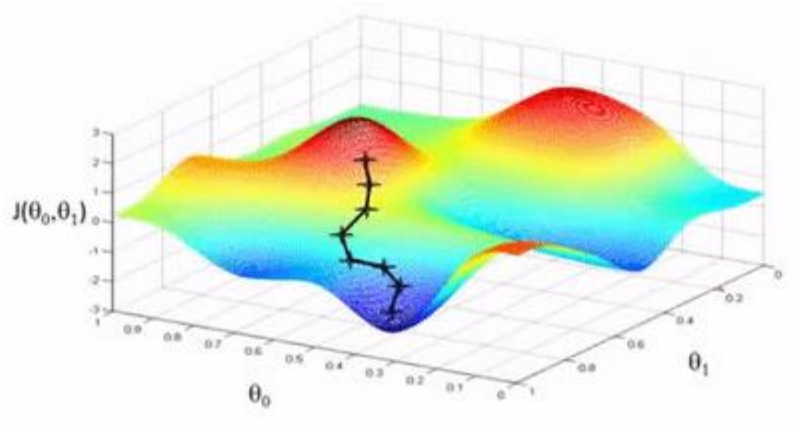
Why do we need nonlinear activation functions?

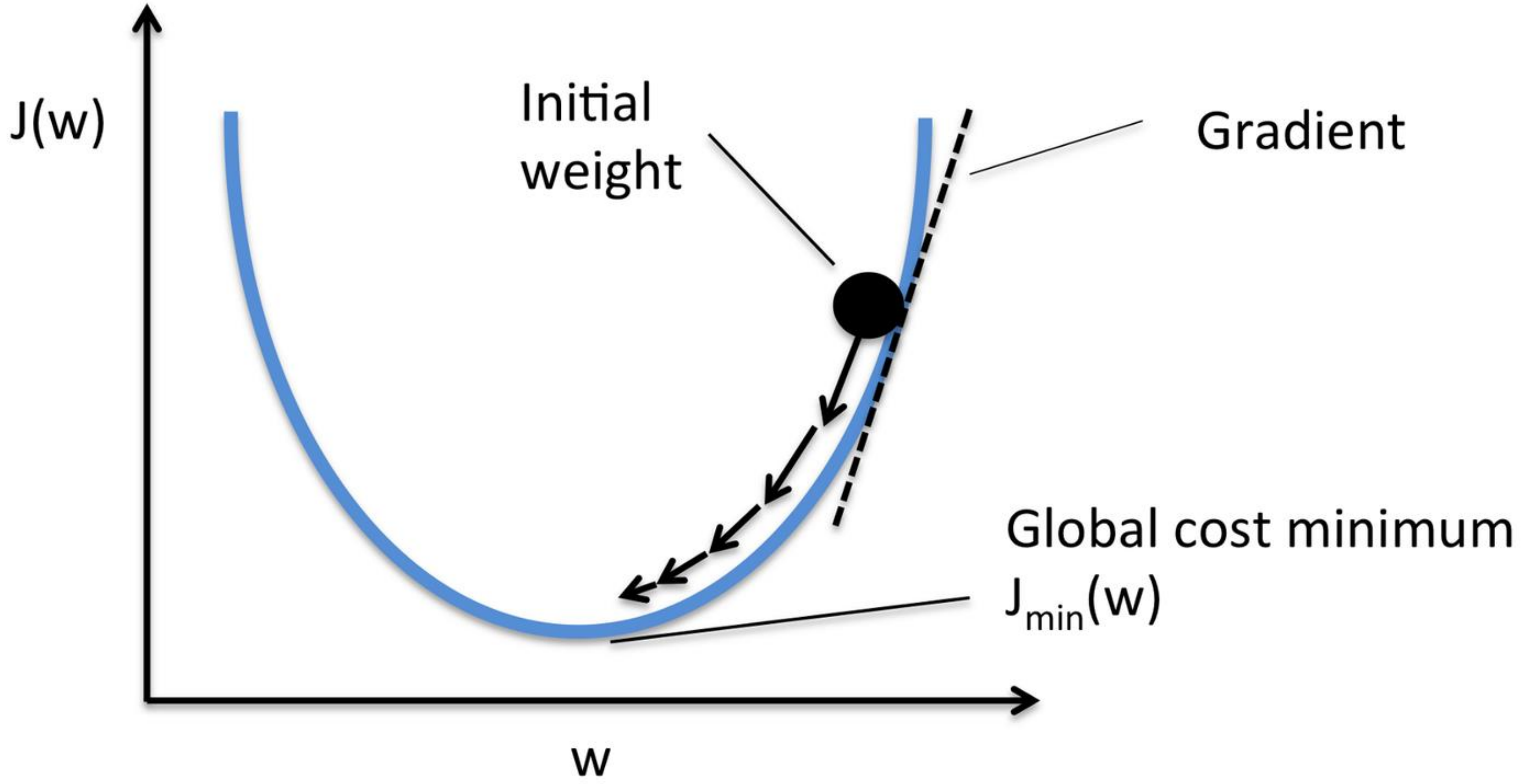
In short it allows the model to learn **complex patterns** and **relationships**

Demo: https://playground.tensorflow.org



Numerical Solution: Gradient Descent





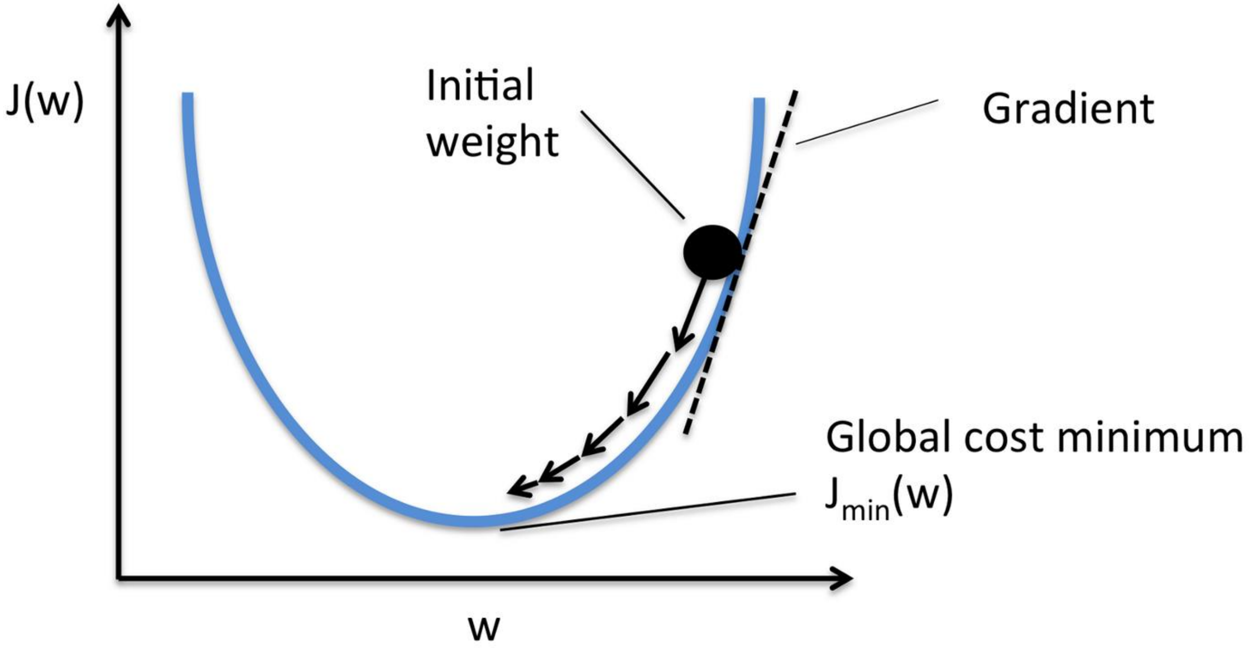
Numerical Solution: Gradient Descent

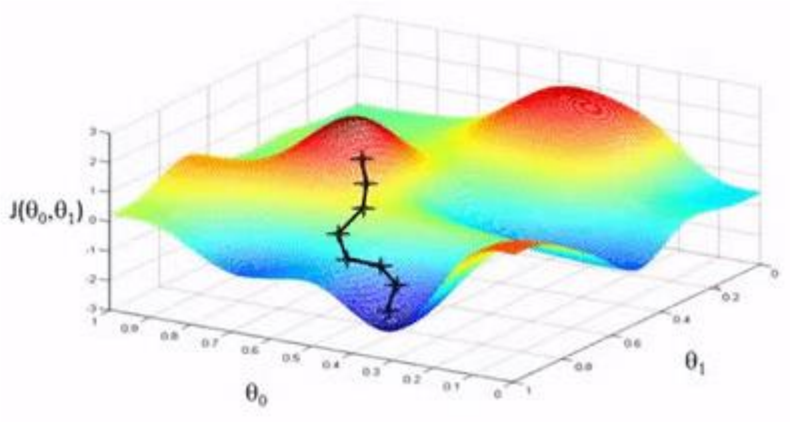
1. Select θ randomly
2. Update θ :

$$\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1})$$

   - where $J(\theta)$ is the average error

3. Iterate until it converges





$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i\theta_i - y_i)^2$$

Numerical Solution: Gradient Descent

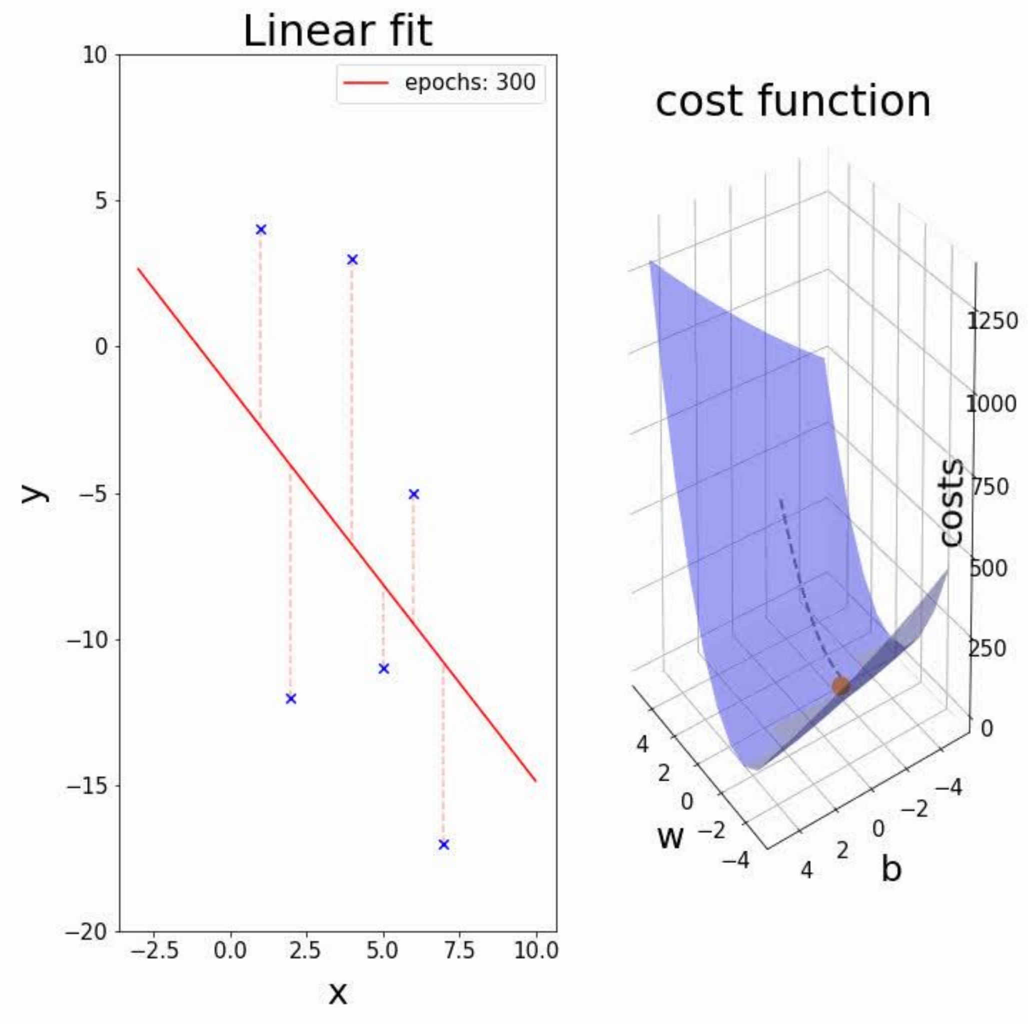
1. Select θ randomly
2. Calculate $\hat{y}$ for all training examples
3. Calculate the gradient of the total loss:

$$\nabla J(\theta_{t-1}) = \nabla \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

4. Update :

$$\theta_t = \theta_{t-1} - \alpha \nabla J(\theta_{t-1})$$

5. Iterate until it converges



Gradient Descent: Learning Rate

1. Update :

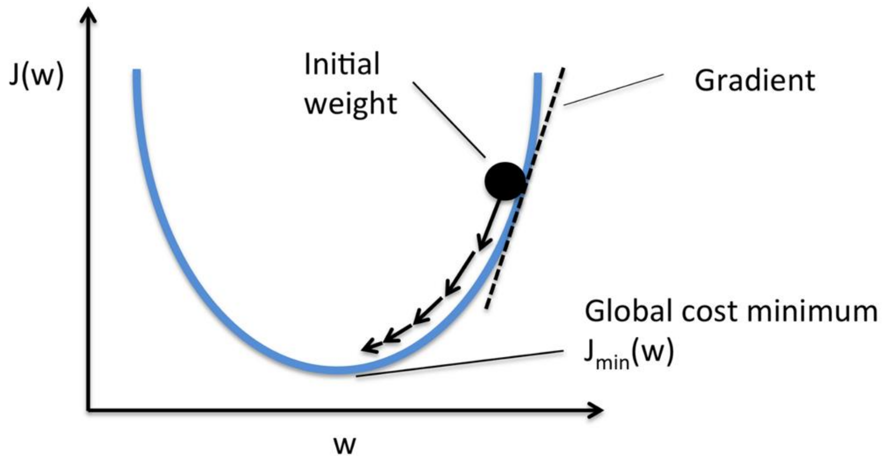
A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

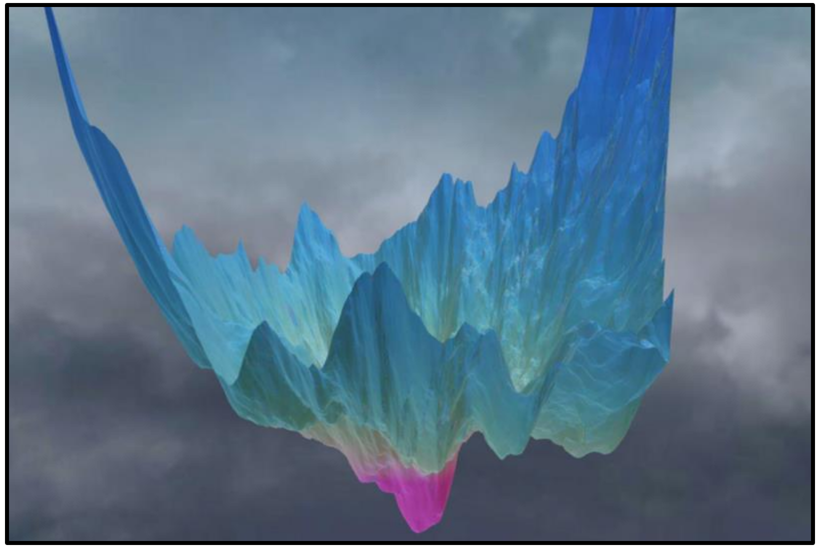
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Gradient Descent

Easy visualization in 2D



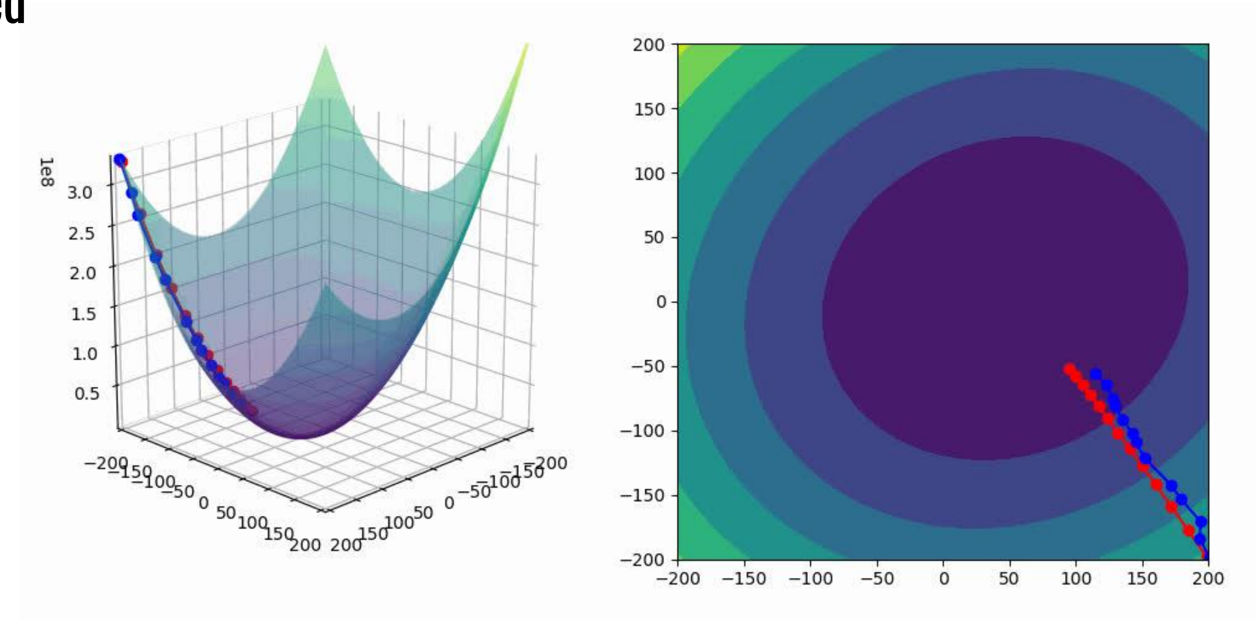
But can be much harder: (demo)
https://losslandscape.com/explorer



More details in the next lecture

Stochastic Gradient Descent (SGD)

- Processing the whole dataset in one batch is straightforward, may not fit into the memory
- Go through the training data in randomly selected batches
- A single pass through on the training data called **epoch**
- batch size can be:
  - Dataset size (**full-batch**)
  - 1, 2, 4 , 8, 16, … (**mini-batch**)



"SGD is like a drunkard trying to find his way home."
— Geoffrey Hinton

Image from: https://towardsdatascience.com/stochastic-gradient-descent-for-machine-learning-clearly-explained-cadcc17d3d11

SGD with Momentum

- To get a smoother trajectory and reduce oscillations in valleys lets introduce **momentum**
- β controls the smoothing (weighted average)

4. Update:

$$\mathbf{m}_t = \beta \cdot \mathbf{m}_{t-1} + (1-\beta)\nabla J(\theta_{t-1})$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \mathbf{m}_{t-1}$$

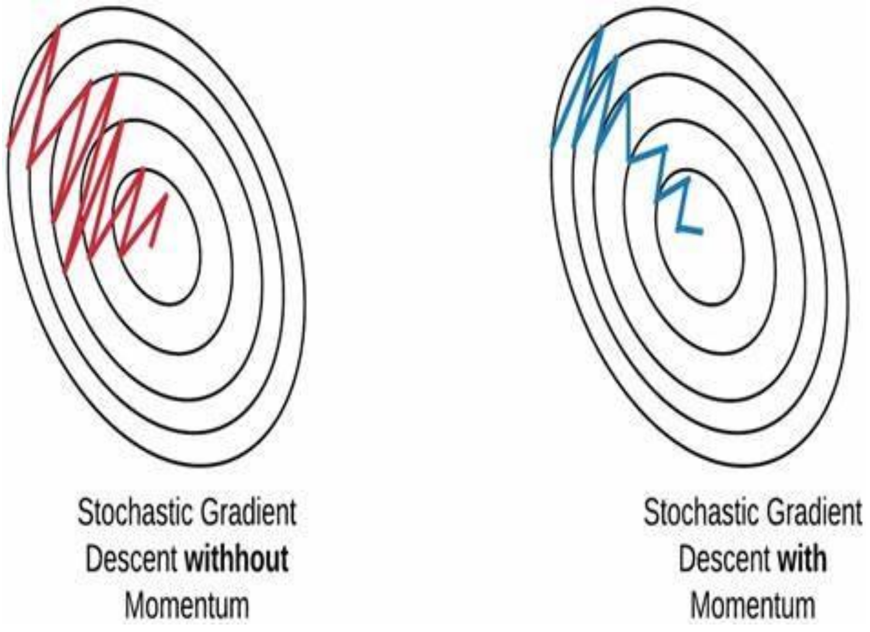
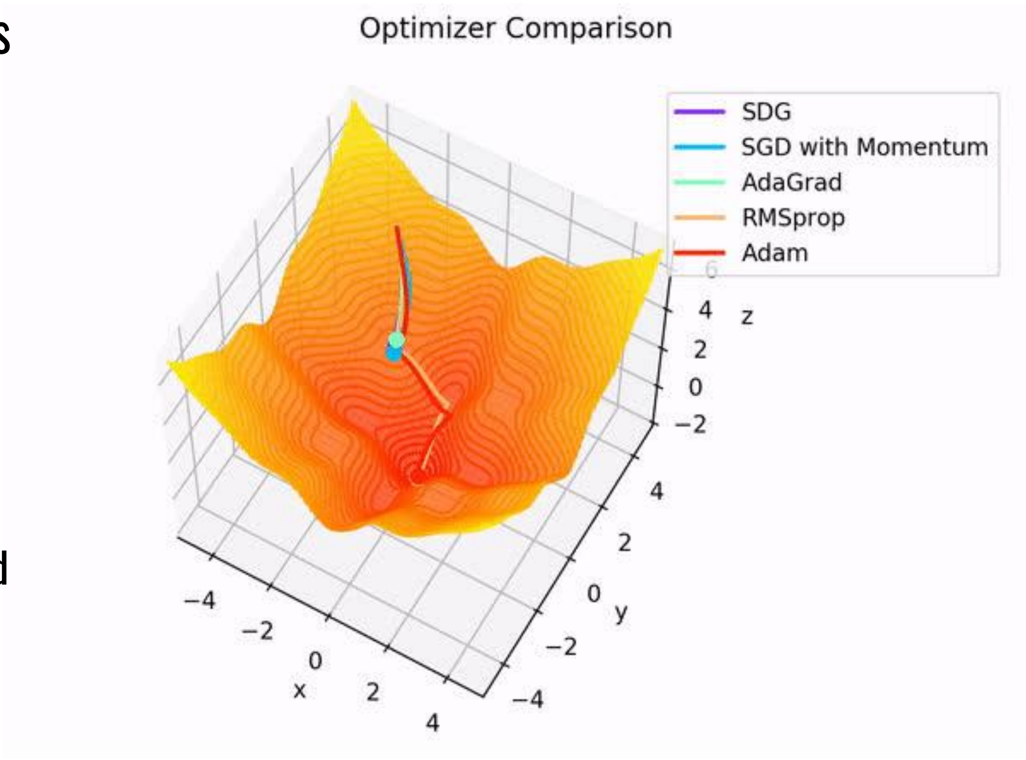


Image from: https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/

Adaptive Moment Estimation (Adam)

- Gradient descent makes large adjustments to parameters with large gradients
- **Solution**: Normalize the gradients so we move fix distances (learning rate)



4. Update:

$$\mathbf{m}_t = \beta_0 \cdot \mathbf{m}_{t-1} + (1-\beta_0)\nabla J(\theta_{t-1})$$
$$\mathbf{v}_t = \beta_1 \cdot \mathbf{v}_{t-1} + (1-\beta_1)\nabla J(\theta_{t-1})^2$$

Measuring gradient and applying momentum

$$\hat{\mathbf{m}}_t = \mathbf{m}_t/(1-\beta_1^t)$$
$$\hat{\mathbf{v}}_t = \mathbf{v}_t/(1-\beta_2^t)$$
$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{\mathbf{m}}_{t-1}/(\sqrt{\hat{\mathbf{v}}_t} + \epsilon)$$

Normalize

Image from: https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/

Deep Network Development

Lecture 2.

Linear Regression & Artificial Neural Networks

Budapest, 21st February 2025

1 Linear Regression

2 Artificial Neural Networks

3 Backpropagation

Backpropagation algorithm

Forward Propagation:

- The NN tries to make its best guess about the correct output
- We run the input data through its functions

Backward Propagation:

- The NN adjusts its parameters based on the error in its guess

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Chain Rule

The chain rule is a formula that expresses the derivative of the composition of two differentiable functions:

$$h(x) = f\big(g(x)\big), \qquad h'(x) = f'\big(g(x)\big)g'(x)$$

Or, equivalently

$$h' = (f \circ g)' = (f' \circ g) \cdot g'$$

The chain rule may also be expressed in Leibniz's notation:

$$\frac{\partial h(x)}{\partial x} = \frac{\partial h(x)}{\partial q} \cdot \frac{\partial q}{\partial z} \cdot \frac{\partial z}{\partial x}, \qquad q \coloneqq f(z), z \coloneqq g(x)$$

Training our Regression Model

We have our model: $\hat{y} = \theta_0 + \theta_1 x = \theta x$

Loss function: $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\theta_i x_i - y_i)^2$

Update rule: $\theta_t = \theta_{t-1} - \alpha\nabla J(\theta_{t-1})$

$$\nabla J(\theta_{t-1}) = \nabla\frac{1}{n}\sum_{i=1}^{n}L(\hat{y}_i, y_i)$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm

Pytorch's autograd keeps a record of data and all executed operations in a directed acyclic graph (**DAG**)

We create a DAG of our executed operations, where the arrows are pointing in the direction of the **forward pass**

- The leaf nodes represents our parameters
- The nodes represents the operations
- The root node represents our loss function

The backward pass kicks off when we call **.backward()** on our root node (loss)

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases}$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases}$$

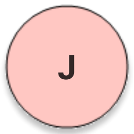
1. Let's create the directed acyclic graph of the operations

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

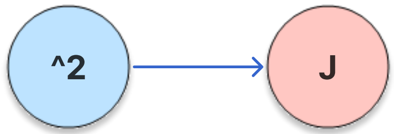


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

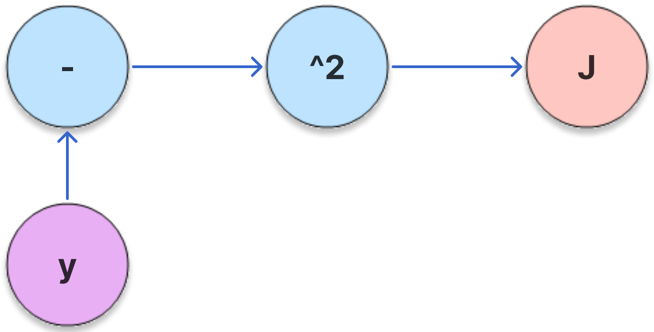


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

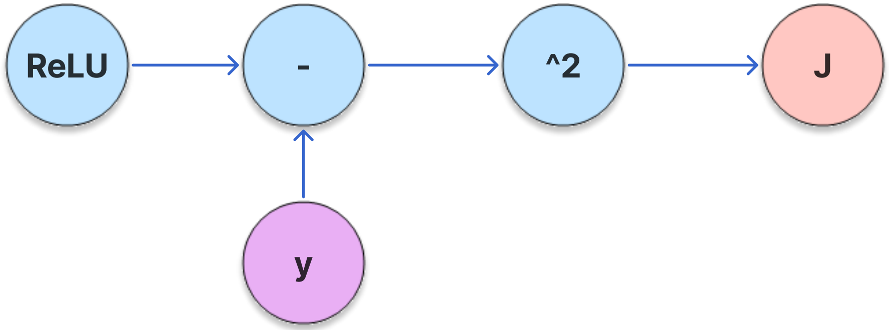


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \le 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

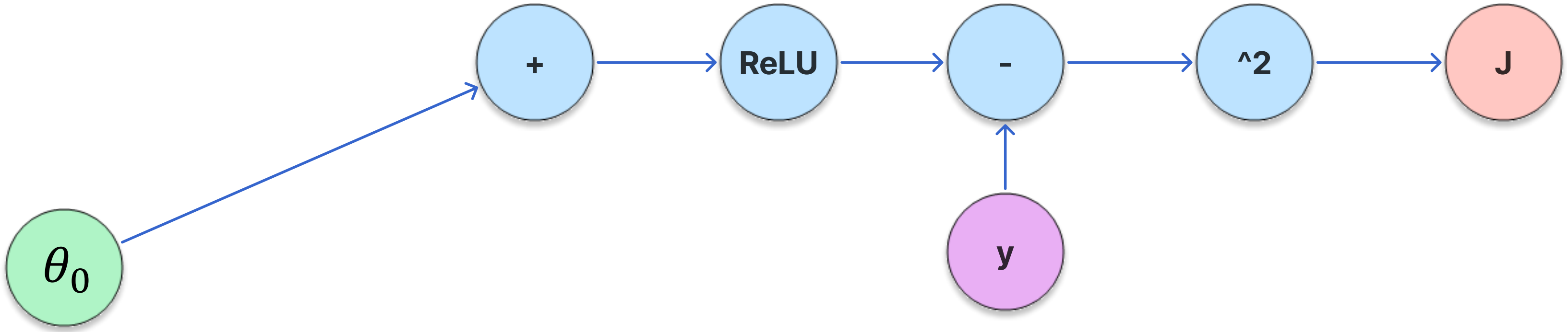


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

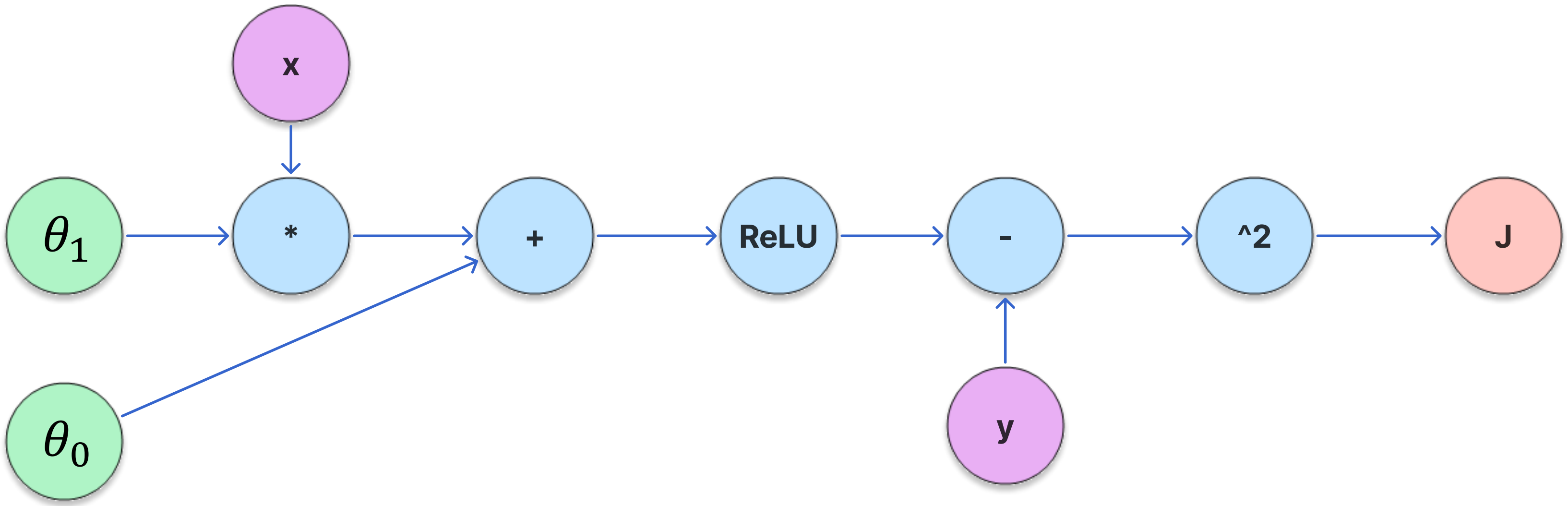


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

1. Let's create the directed acyclic graph of the operations

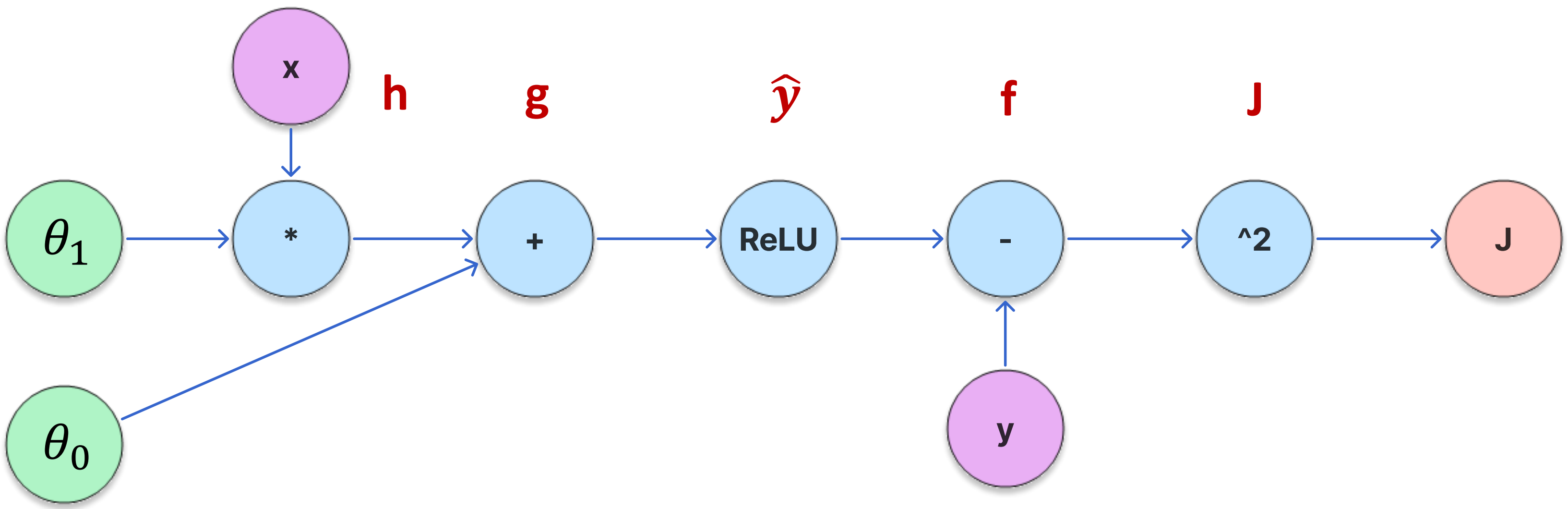


https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

$$J(\theta) = (\hat{y} - y)^2, \qquad \hat{y} = ReLU(\theta_0 + x\theta_1), \qquad ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

2. Name these nodes



https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

2. Name these nodes

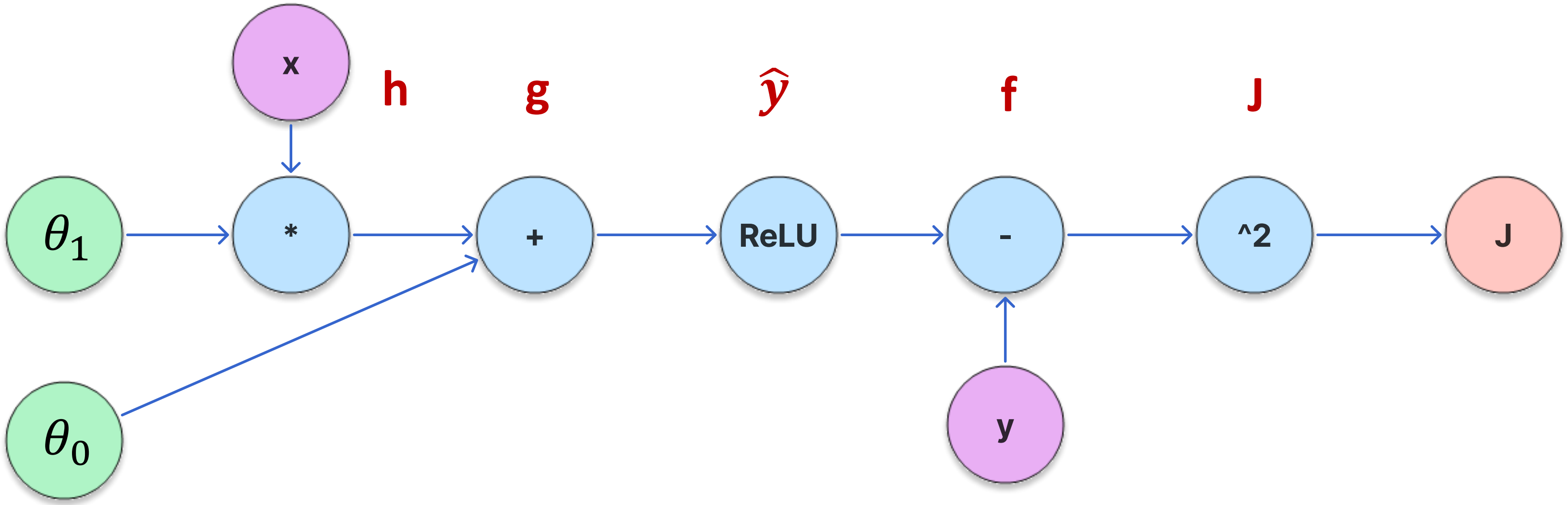
$$J = f^2$$

$$f = \hat{y} - y$$

$$\hat{y} = ReLU(g)$$

$$g = h + \theta_0$$

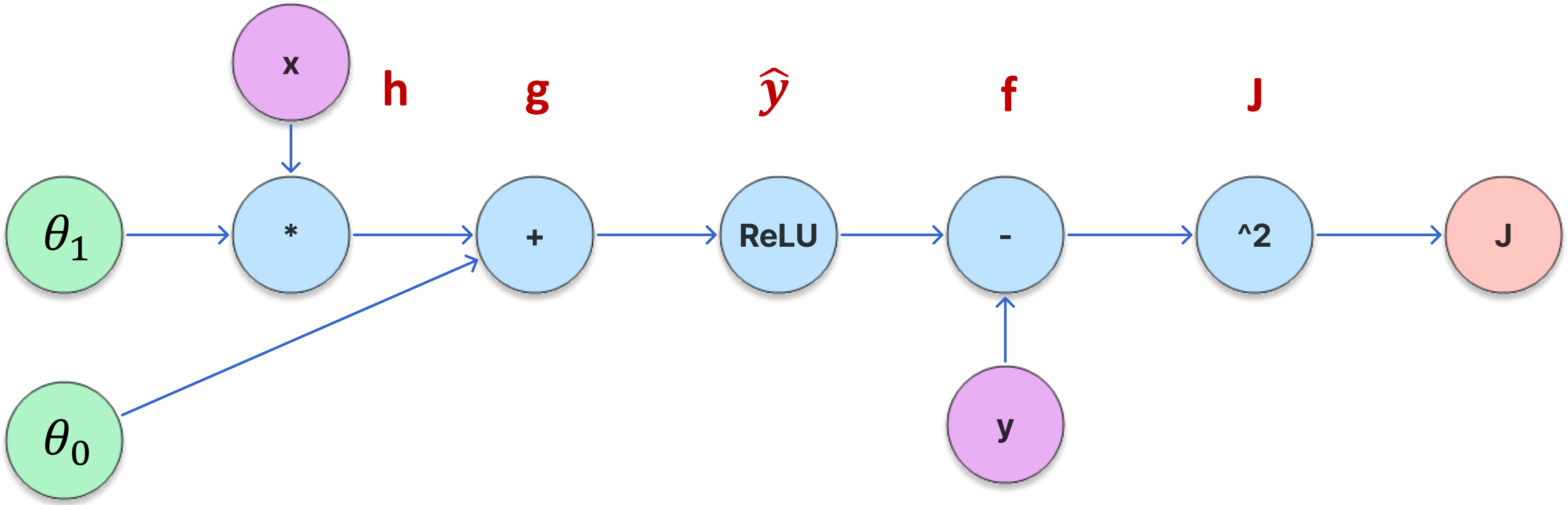
$$h = \theta_1 x$$



https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation



$$J = f^2$$

$$f = \hat{y} - y$$

$$\hat{y} = ReLU(g)$$

$$g = h + \theta_0$$

$$h = \theta_1 x$$

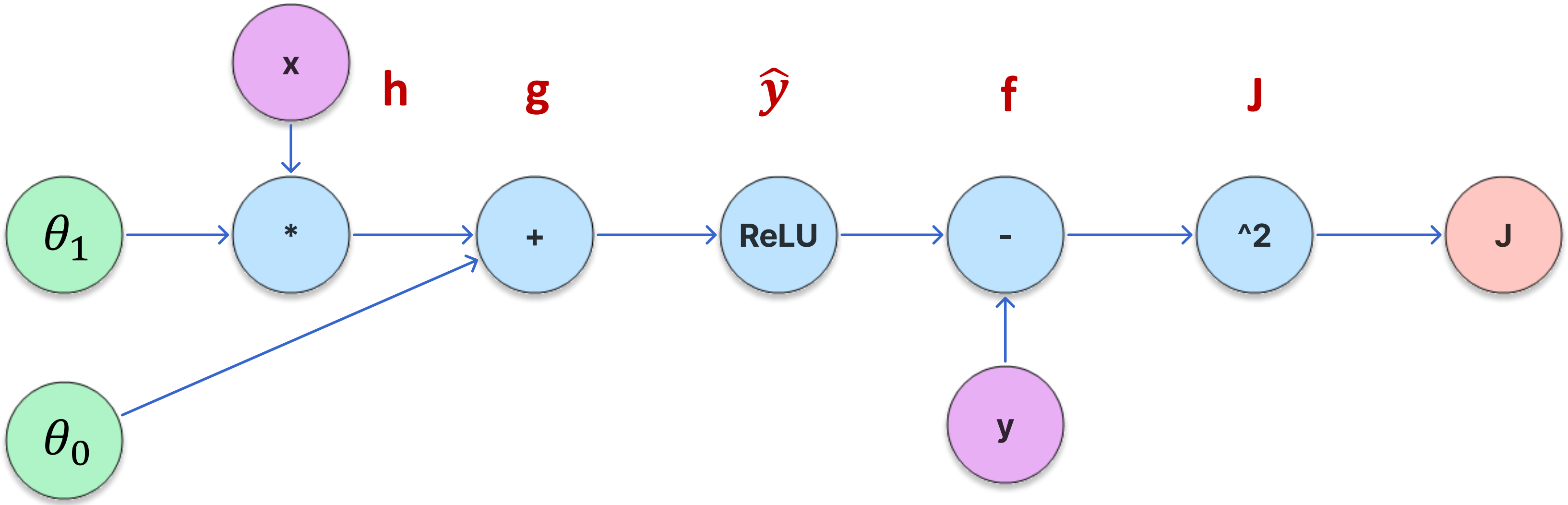
$$\frac{\partial J}{\partial \theta_0} = ?$$

$$\frac{\partial J}{\partial \theta_1} = ?$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

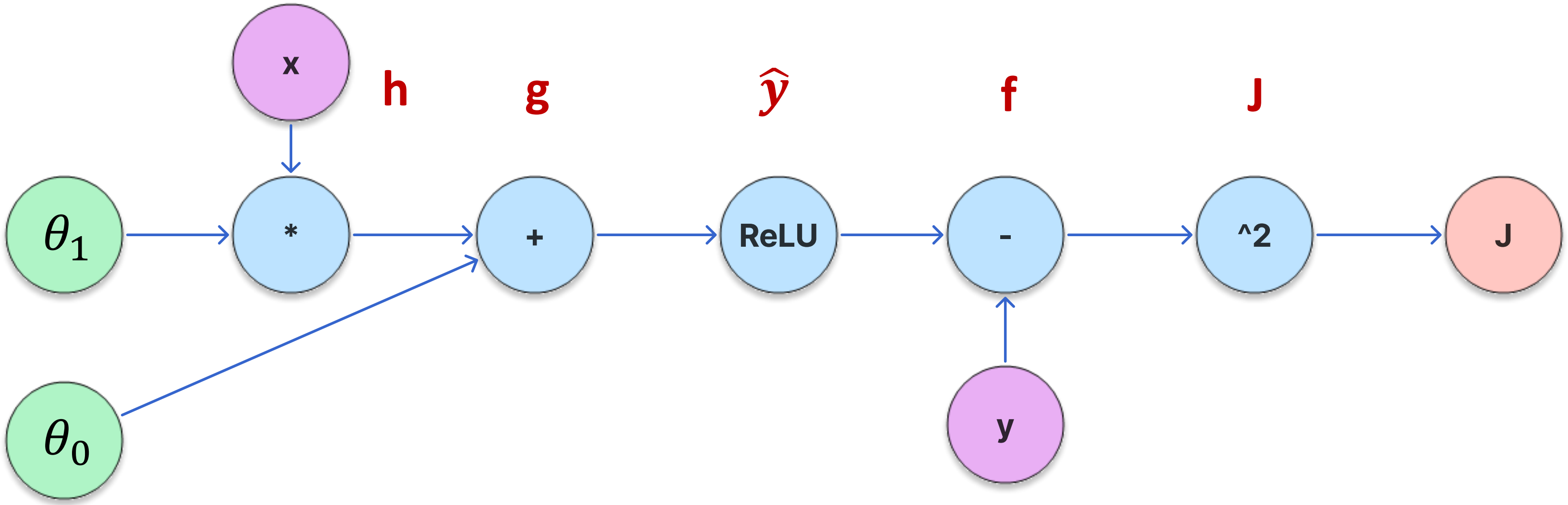
$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1 \qquad \frac{\partial J}{\partial \theta_0} = ?$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x \qquad \frac{\partial J}{\partial \theta_1} = ?$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

3. Symbolic differentiation



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass

$\frac{\partial J}{\partial J} = 1$

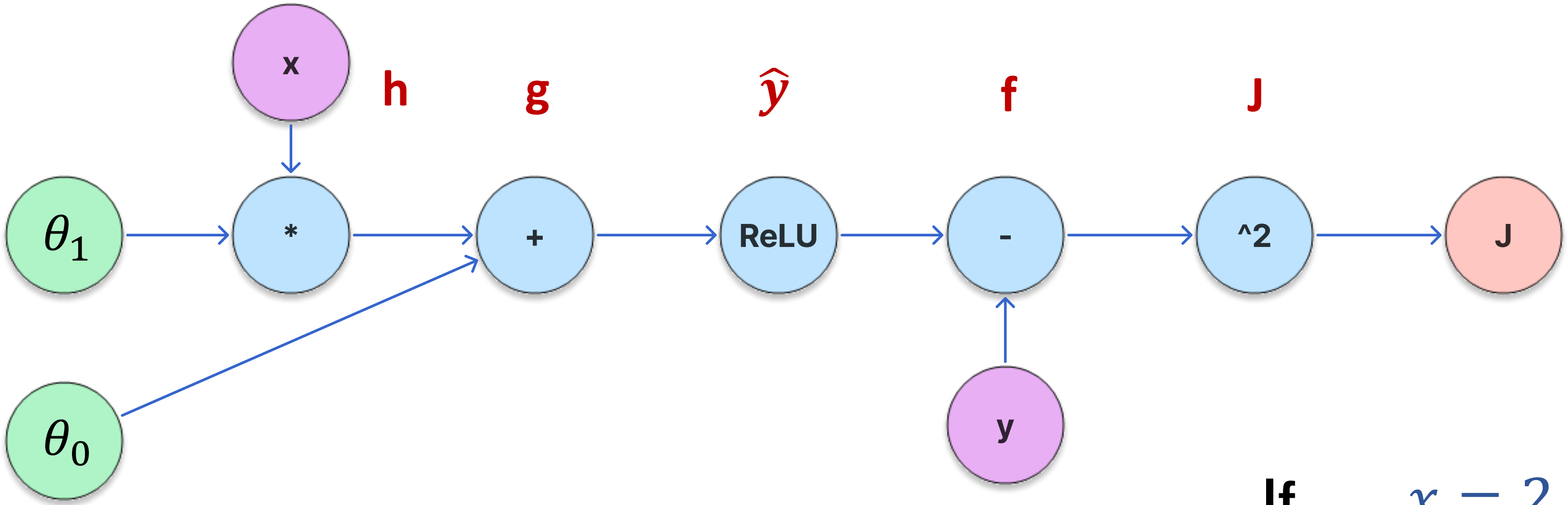
$J = f^2$ $\quad \frac{\partial J}{\partial f} = 2f$

$f = \hat{y} - y$ $\quad \frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\quad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$

$g = h + \theta_0$ $\quad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\quad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$
$y = 1$

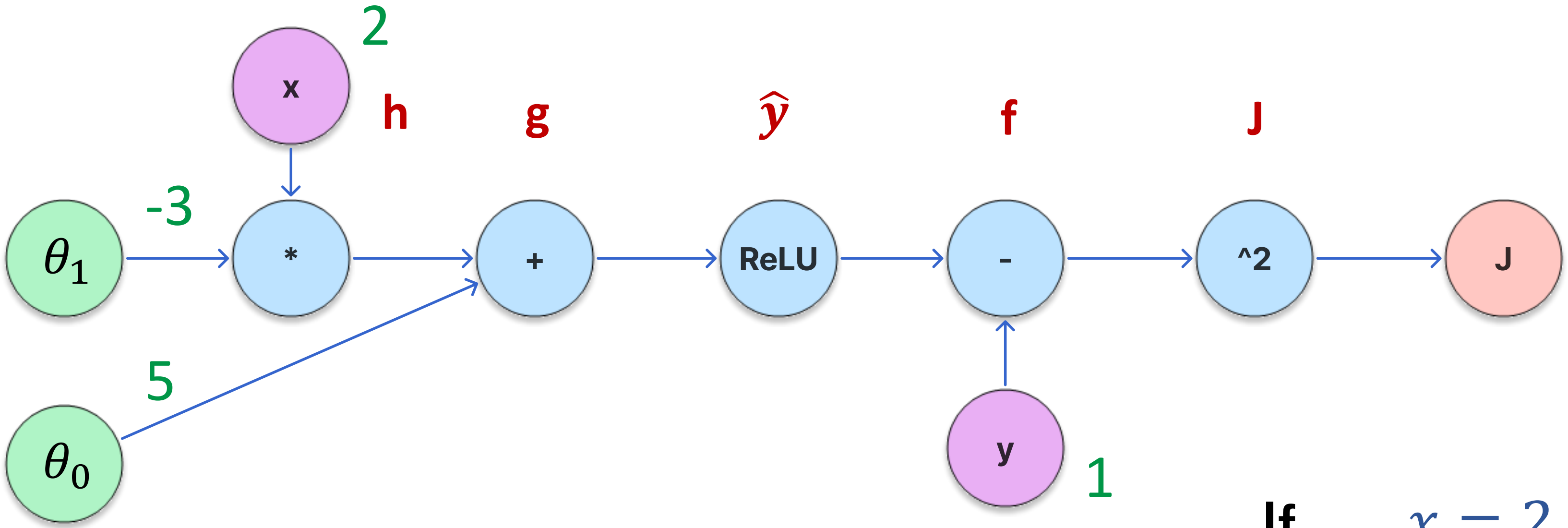
$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$, $y = 1$

$\theta_0 = 5$

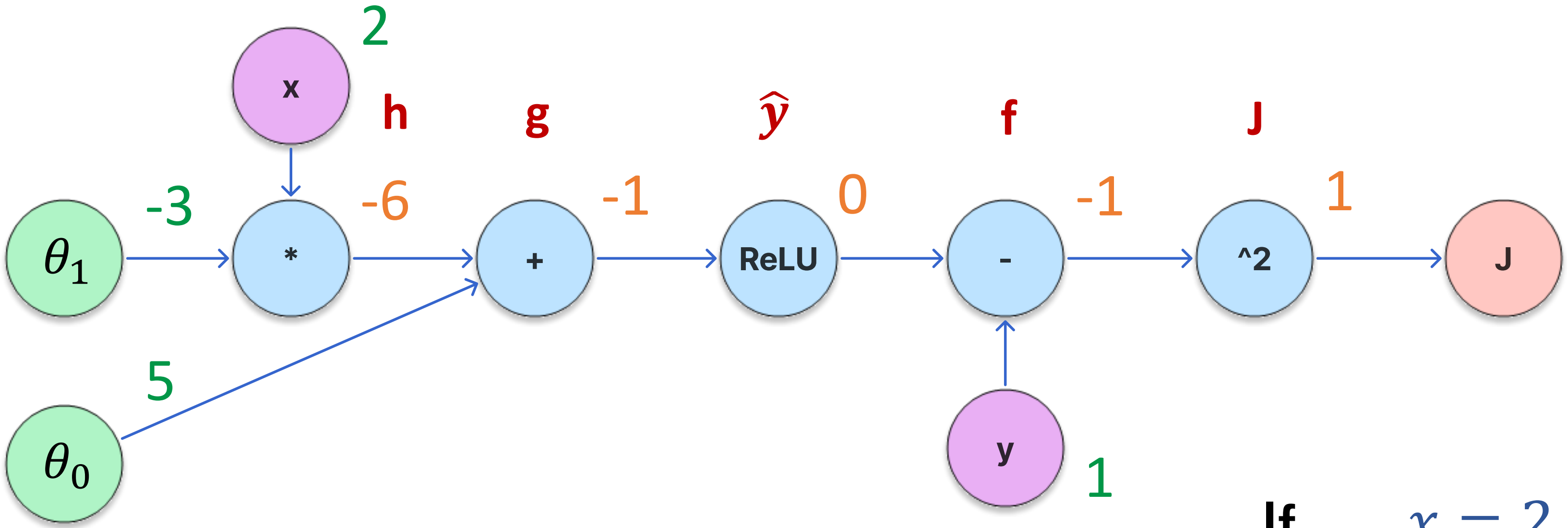
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – forward pass



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

If
$$x = 2$$
$$y = 1$$

$$\theta_0 = 5$$
$$\theta_1 = -3$$

$$\alpha = 0.1$$

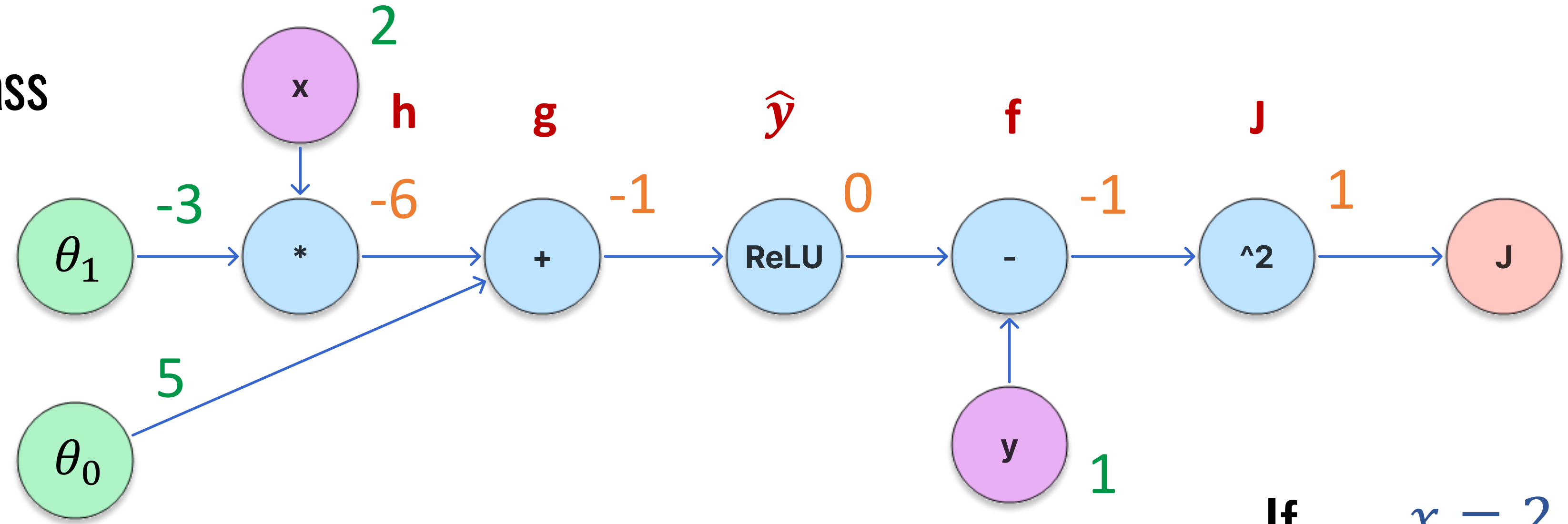
$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – backward pass



$$\frac{\partial J}{\partial J} = 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$
$y = 1$

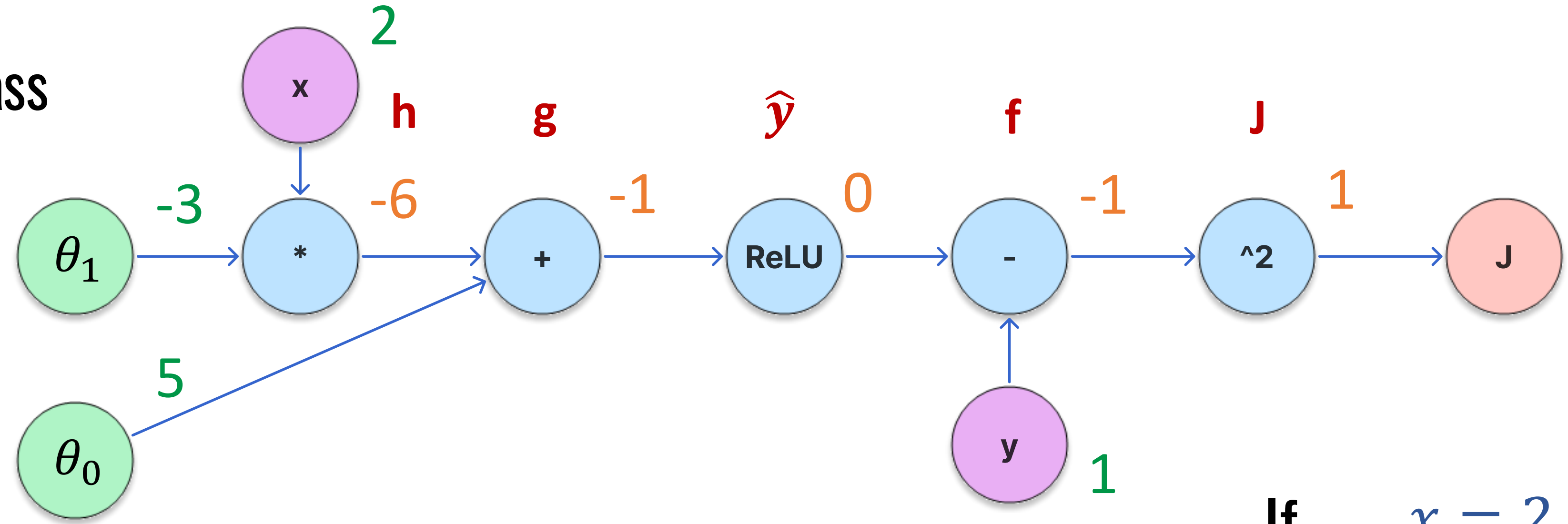
$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

4. Substitution – backward pass



$$\frac{\partial J}{\partial J} = 1 \qquad 1$$

$$J = f^2 \qquad \frac{\partial J}{\partial f} = 2f \qquad -2$$

$$f = \hat{y} - y \qquad \frac{\partial f}{\partial \hat{y}} = 1$$

$$\hat{y} = ReLU(g) \qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases} \qquad 0$$

$$g = h + \theta_0 \qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$$

$$h = \theta_1 x \qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$$

-3 2

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$, $y = 1$

$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

5. Update weights

What happened? Did we learn?

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 0$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 0$$

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 0$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -3 - 0.1 \cdot 0$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -3$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

3. Symbolic differentiation

$$\frac{\partial J}{\partial J} = 1$$

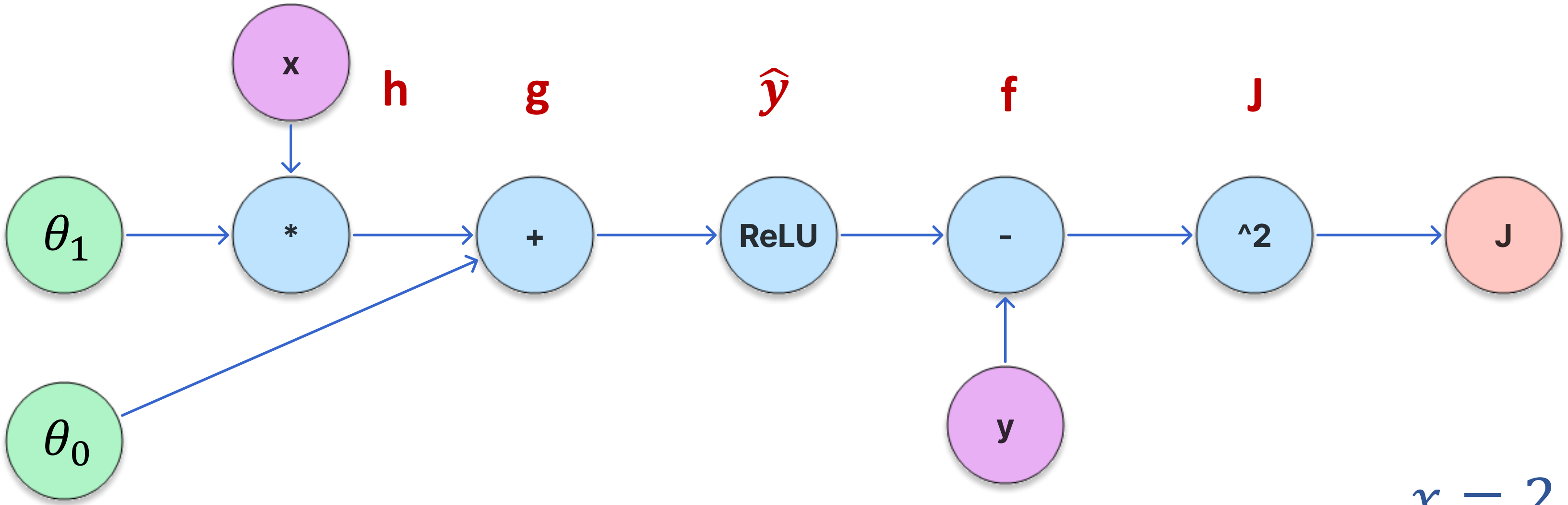
$J = f^2$ $\qquad \frac{\partial J}{\partial f} = 2f$

$f = \hat{y} - y$ $\qquad \frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\qquad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$

$g = h + \theta_0$ $\qquad \frac{\partial g}{\partial h} = 1 \qquad \frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\qquad \frac{\partial h}{\partial x} = \theta_1 \qquad \frac{\partial h}{\partial \theta_1} = x$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – forward pass

$\frac{\partial J}{\partial J} = 1$

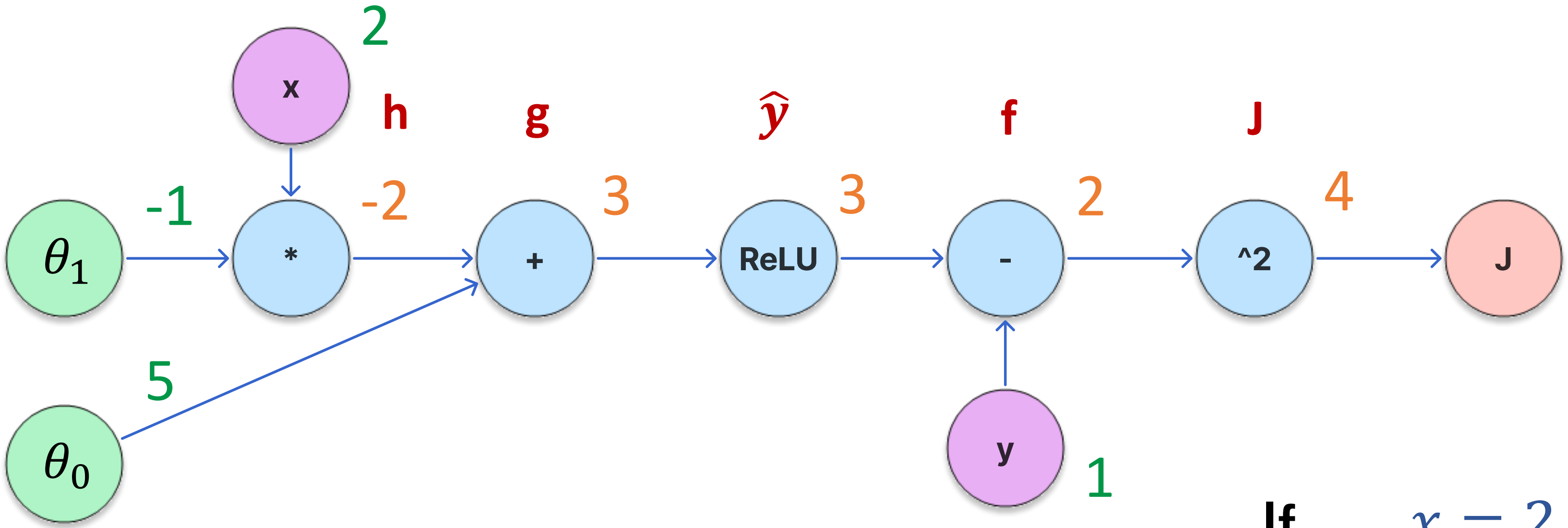
$J = f^2$ $\quad \frac{\partial J}{\partial f} = 2f$

$f = \hat{y} - y$ $\quad \frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\quad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$

$g = h + \theta_0$ $\quad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\quad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass

$\frac{\partial J}{\partial J} = 1$ 1

$J = f^2$ $\quad \frac{\partial J}{\partial f} = 2f$ 4

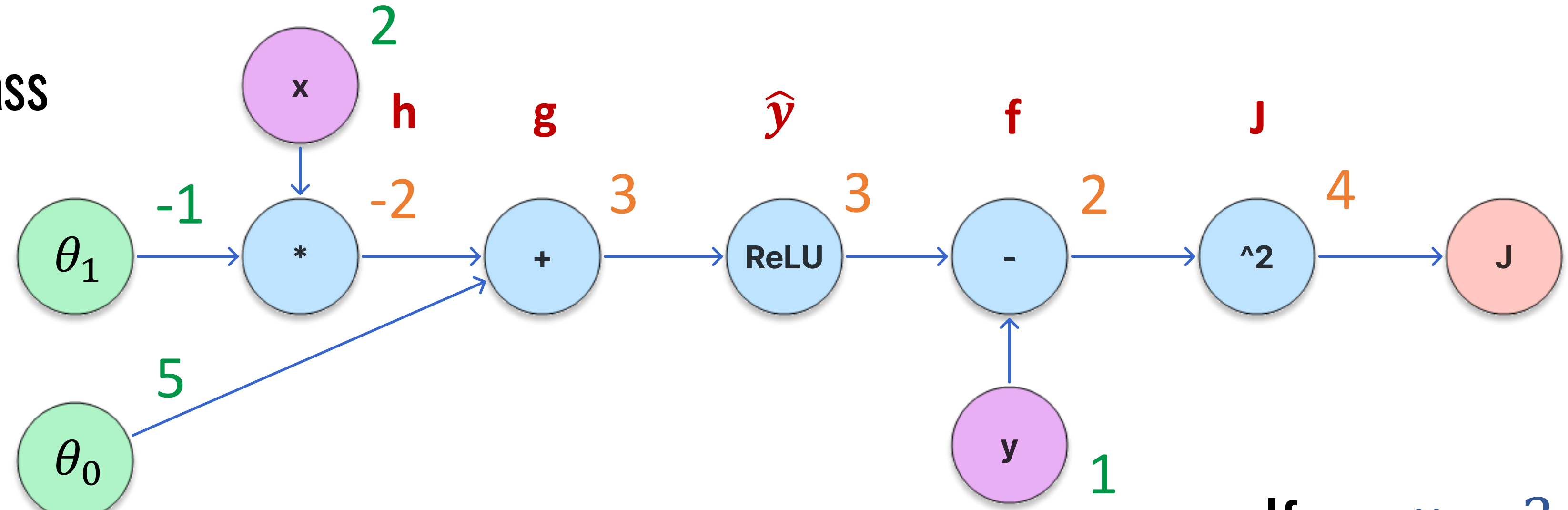
$f = \hat{y} - y$ $\quad \frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\quad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$ 1

$g = h + \theta_0$ $\quad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\quad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$

-1 2



$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

If $\quad x = 2$

$y = 1$

$\theta_0 = 5$

$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass

$\frac{\partial J}{\partial J} = 1$ 1

$J = f^2$ $\frac{\partial J}{\partial f} = 2f$ 4

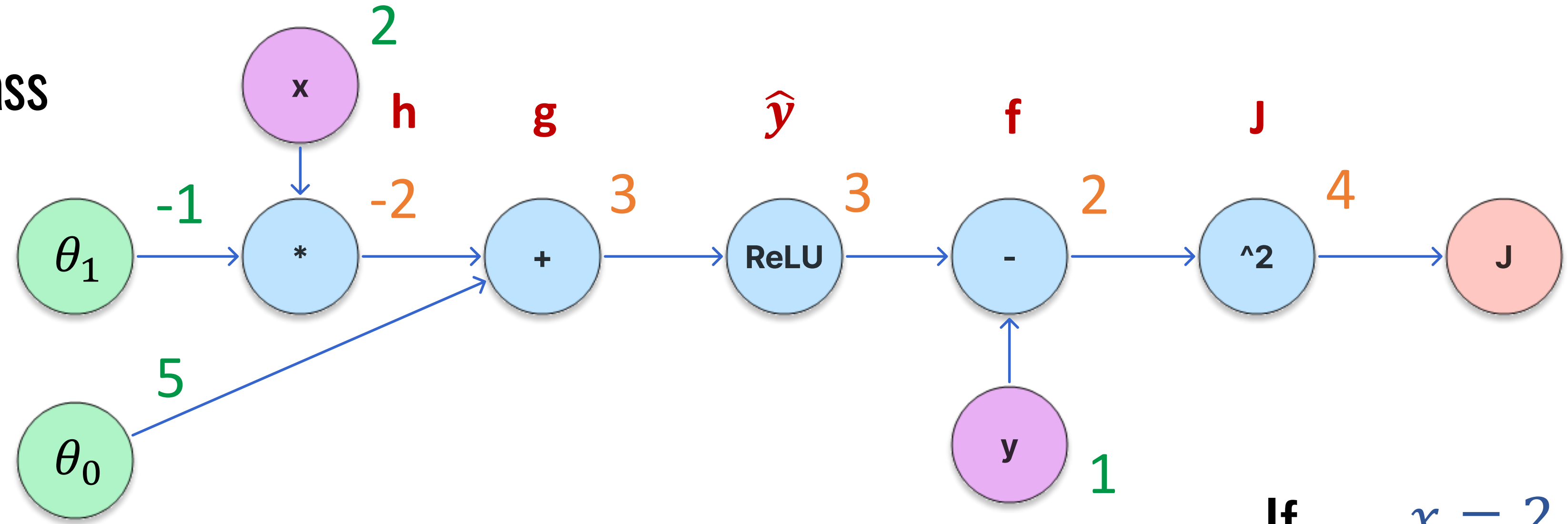
$f = \hat{y} - y$ $\frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$ 1

$g = h + \theta_0$ $\frac{\partial g}{\partial h} = 1$ $\frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\frac{\partial h}{\partial x} = \theta_1$ $\frac{\partial h}{\partial \theta_1} = x$

-1 2



1 4 1 1 1

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$$

1 4 1 1 1 2

If $x = 2$
$y = 1$

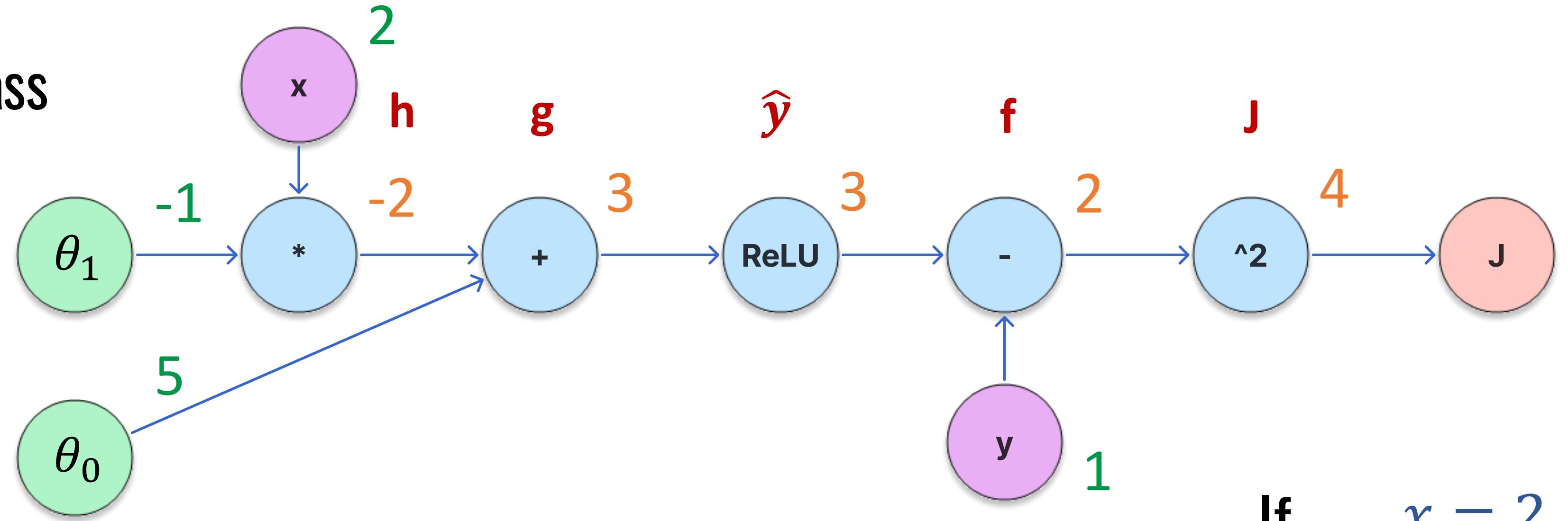
$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (other example)

4. Substitution – backward pass



$\frac{\partial J}{\partial J} = 1$ 1

$J = f^2$ $\quad \frac{\partial J}{\partial f} = 2f$ 4

$f = \hat{y} - y$ $\quad \frac{\partial f}{\partial \hat{y}} = 1$

$\hat{y} = ReLU(g)$ $\quad \frac{\partial \hat{y}}{\partial g} = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$ 1

$g = h + \theta_0$ $\quad \frac{\partial g}{\partial h} = 1 \quad \frac{\partial g}{\partial \theta_0} = 1$

$h = \theta_1 x$ $\quad \frac{\partial h}{\partial x} = \theta_1 \quad \frac{\partial h}{\partial \theta_1} = x$

-1 2

1 4 1 1 1

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 8$$

1 4 1 1 1 2

If $x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Backpropagation algorithm (example)

5. Update weights

What happened? Did we learn?

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial \theta_0} = 4$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial J} \cdot \frac{\partial J}{\partial f} \cdot \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \theta_1} = 8$$

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = 5 - 0.1 \cdot 4 = 4.6$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = -1 - 0.1 \cdot 8 = 1.8$$

$x = 2$
$y = 1$

$\theta_0 = 5$
$\theta_1 = -1$

$\alpha = 0.1$

https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Summary

- Linear regression is a supervised learning method
- It tries to find the best-fitting linear model that minimizes the error
- Used when the output is continuous (not discrete)

- Can be solved with analytical or numerical solution

<u>Analytical (Normal equation)</u>

  - Closed-form, provides a direct solution

<u>Numerical (Gradient Descent)</u>

  - Iterative approach, provides an approximation

- Fully Connected Networks (Feed Forward Networks) are inspired by the biological neural networks in the brain (but there are some differences)
- Can be used in tasks like Linear Regression where we optimize the weights and biases

Summary

In Artificial Neural Network we introduced non-linearity, we don't have an exact solution, so we use the iterative approach for optimizing:

- **Gradient Descent:** we calculate the loss on the whole dataset and adjust the parameters
- **Stochastic Gradient Descent**: we split our dataset into random mini-batches (and train, test, validation sets), we calculate the loss on these mini-batches, update based on these
- **Stochastic Gradient Descent with Momentum**: we smooth out the updates by applying a weighted average of the gradients
- **Adaptive Momentum**: we normalize the gradients, so the updates are uniform

Backpropagation algorithm:

- Calculating the gradient for our deep learning model.
- Forward pass – prediction – loss – backward pass.
- Calculating symbolic differentiation.

Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

That’s all for today!