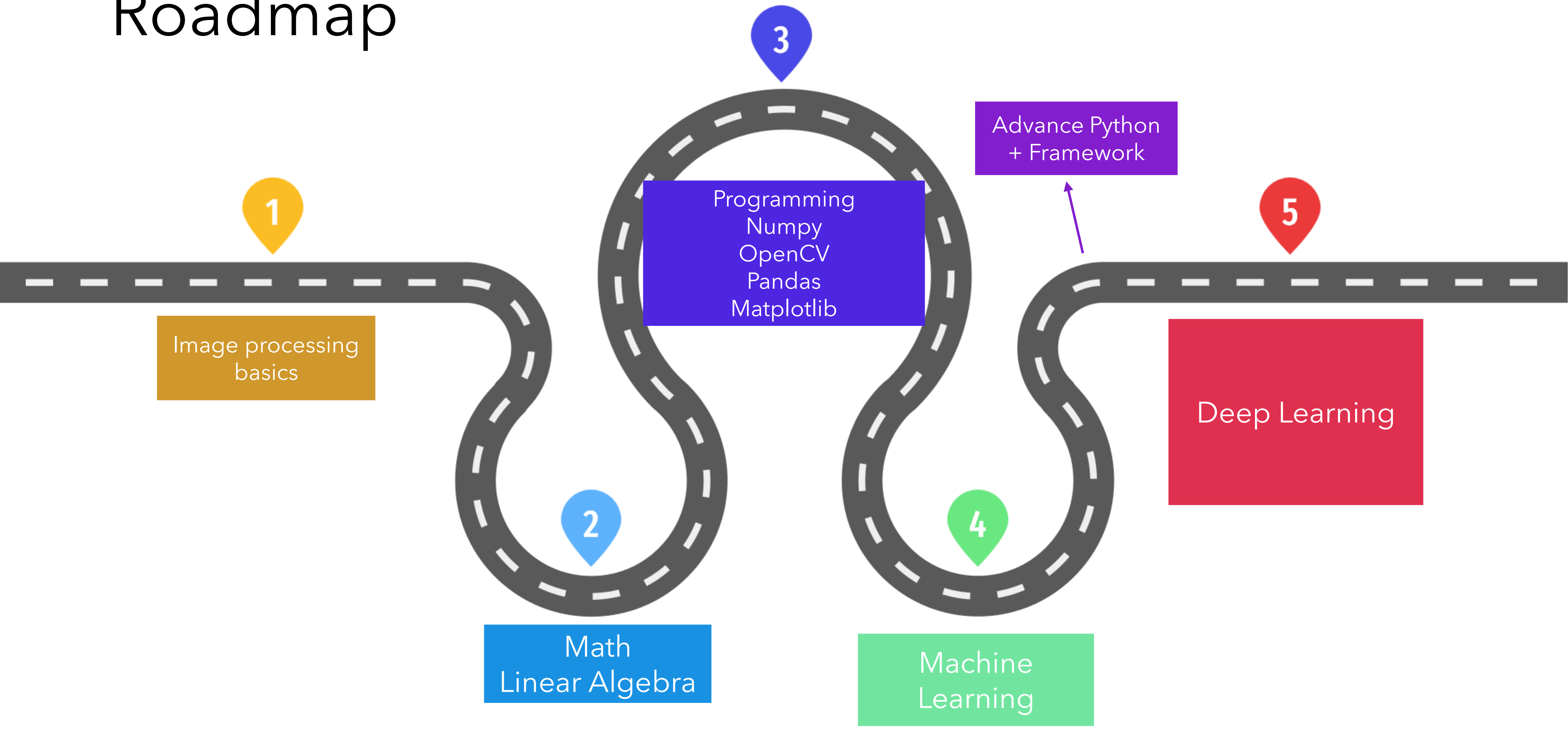
Computer Vision

CVI620

Session 16
03/2025

Roadmap



1
Image processing basics
2
Math
Linear Algebra
3
Programming
Numpy
OpenCV
Pandas
Matplotlib
Advance Python
+ Framework
4
Machine Learning
5
Deep Learning

What is Left?

11 sessions

1. Optimization and Loss Function
2. Code + Logistic Regression
3. ML and Images
4. Perceptron and Neural Networks
5. Deep Neural Networks
6. Convolution Neural Networks (CNN)
7. Advanced CNNs
8. Project
9. Segmentation
10. Introduction to object detection and image generation methods with AI
11. Project

Agenda

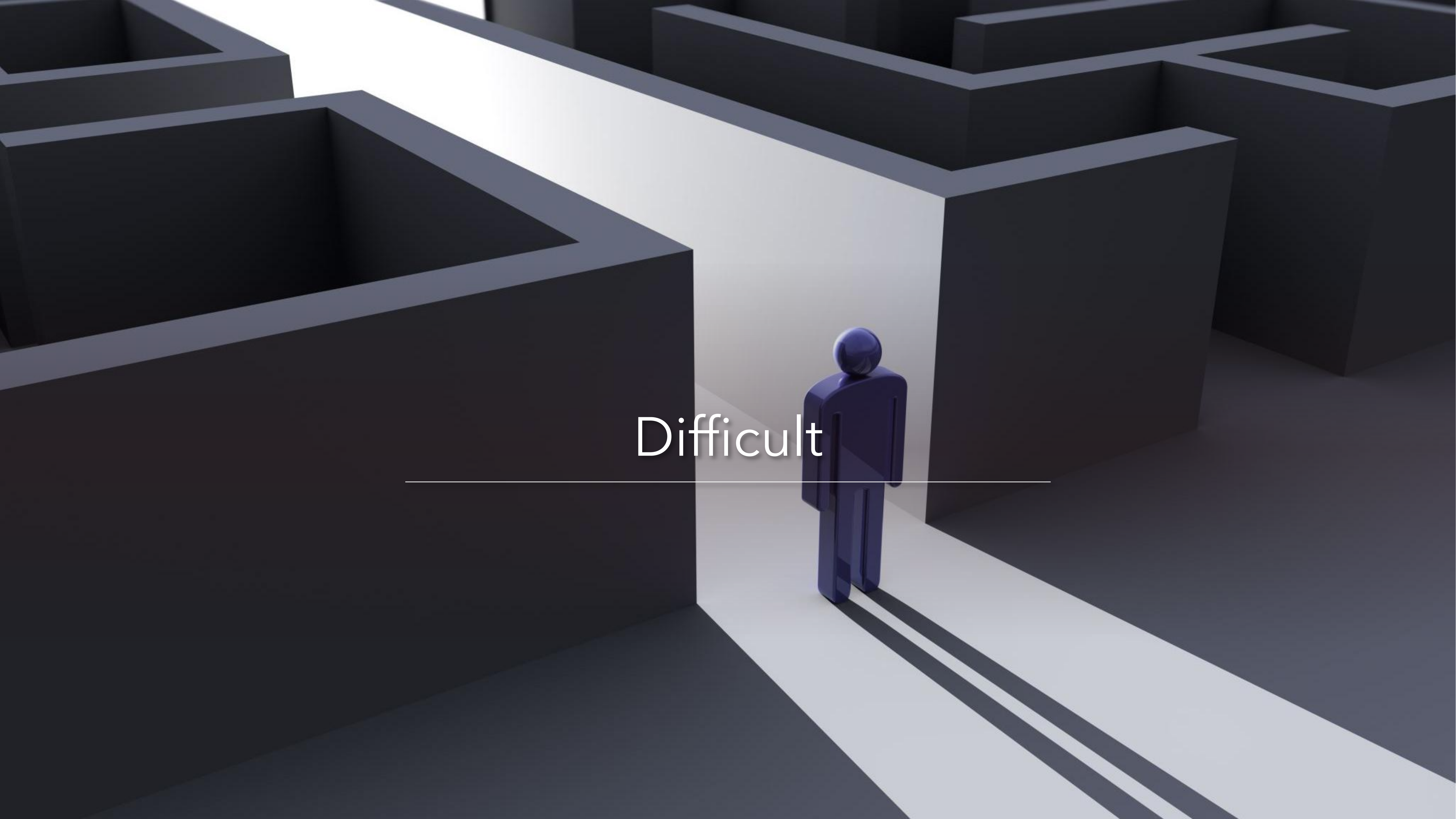
01 Multiple Linear Regression

02 Optimization

03 Gradient Descend

04 Example Code

Difficult



Questions from Session 15 Videos?

Regression

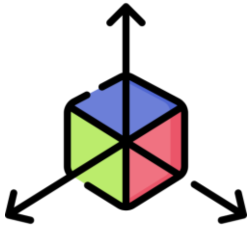
i) Simple

linear

nonlinear

i) Multiple

linear



Polynomial

nonlinear

Simple vs Multiple

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Simple vs Multiple

Simple Linear Regression	$y = b_0 + b_1{}^*x_1$
Multiple Linear Regression	$y = b_0 + b_1{}^*x_1 + b_2{}^*x_2 + \ldots + b_n{}^*x_n$

Still calculate derivative and solve variable values!

Multiple Linear Regression

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error


dataset = pd.read_csv('S16/petrol_consumption.csv')
X = dataset[['Petrol_tax', 'Average_income', 'Paved_Highways', 'Population_Driver_licence(%)']]
y = dataset['Petrol_Consumption']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

regressor = LinearRegression()
regressor.fit(X_train, y_train)
coeff_df = pd.DataFrame(regressor.coef_, X.columns, columns=['Coefficient'])
print(coeff_df)


y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
```

Problem

- Some datasets have millions of features -> impossible to calculate
- They are not always linear
- Some problems do not have closed form formulas

Problem

- Some datasets have millions of features -> impossible to calculate
- They are not always linear
- Some problems do not have closed form formals

Solution -> Optimization

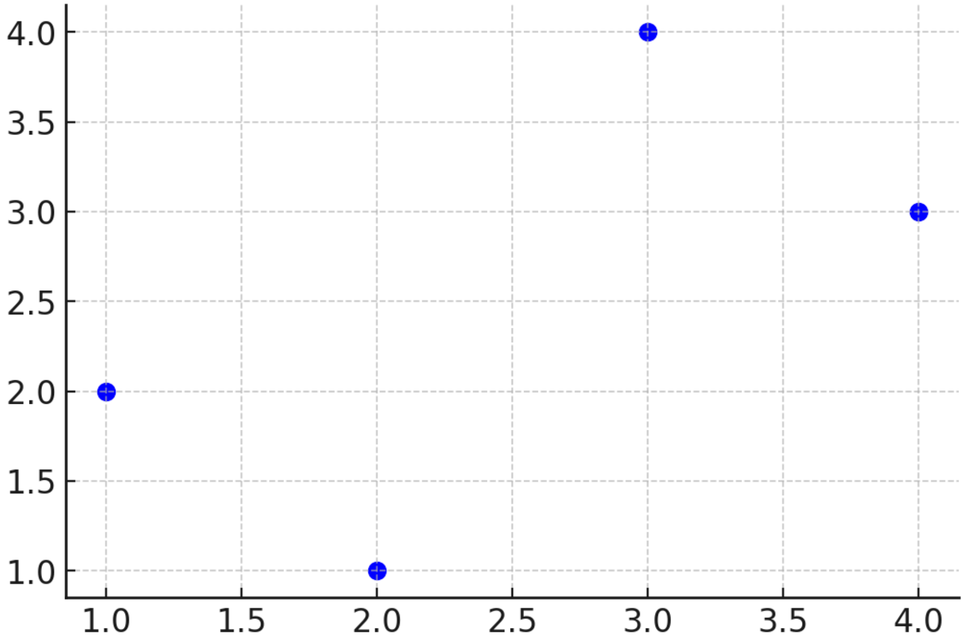
Optimization

Start random in space

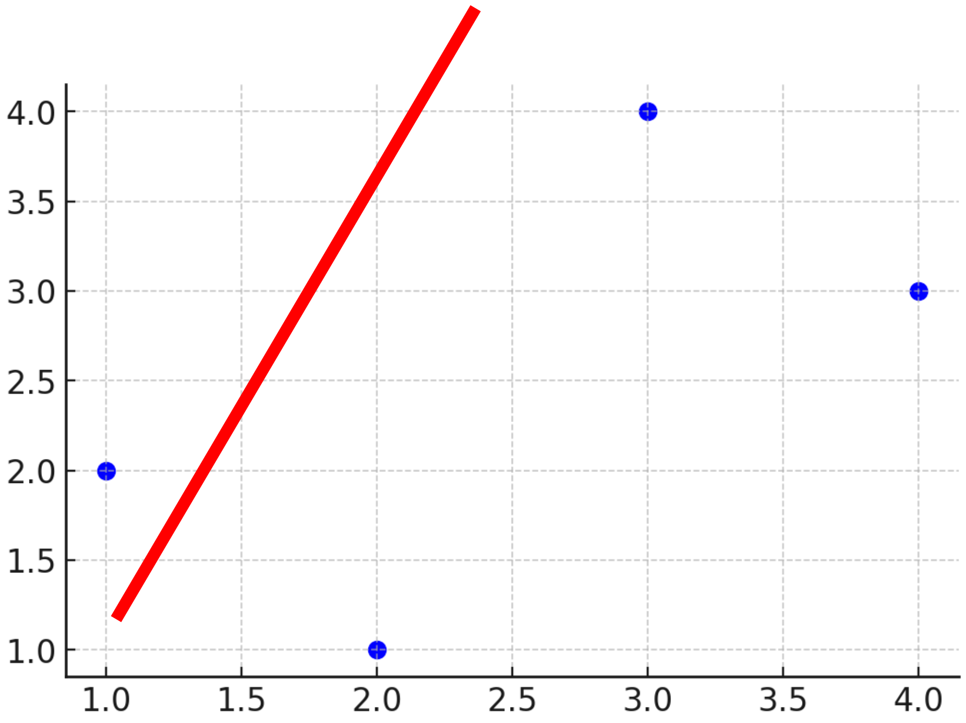
Take gradual steps towards your goal

Not the best best answer but a solution close to the best

Example



Example



What is our goal?

What is our goal?

Minimize loss

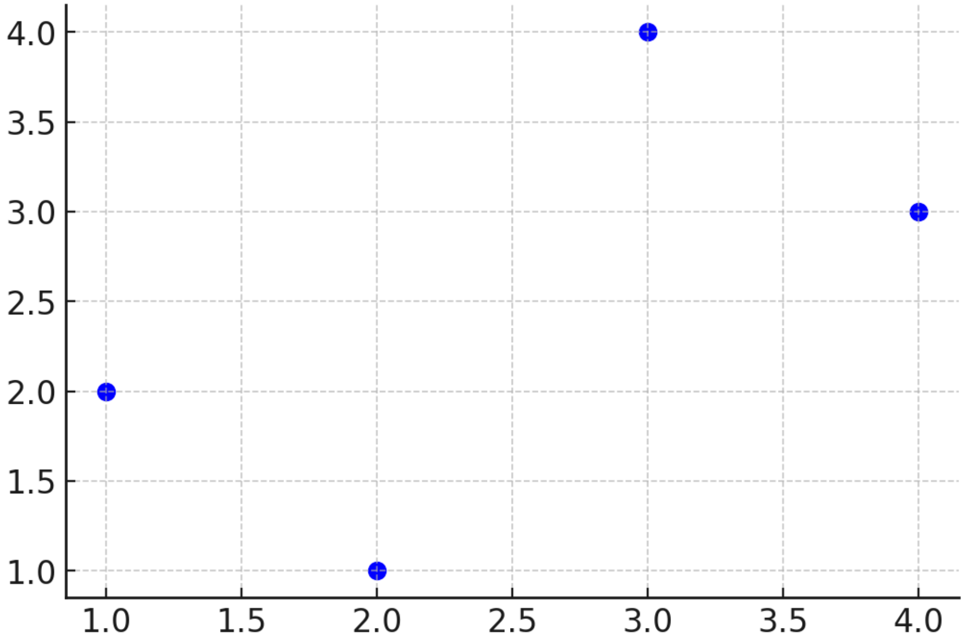
Gain parameter values

What Was Loss?

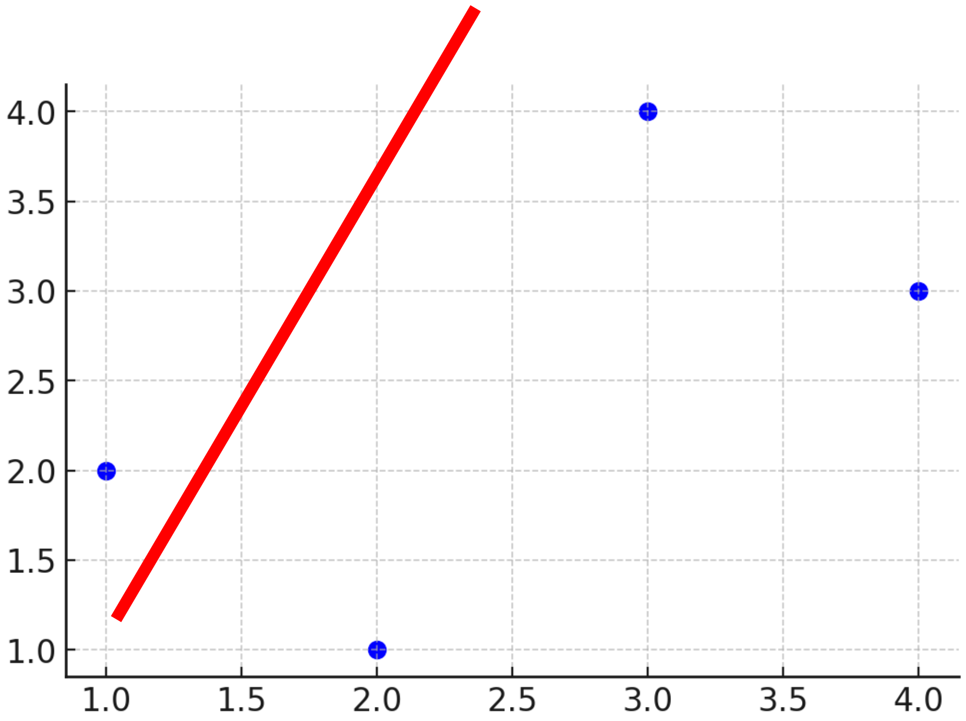
- MSE (mean squared error)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

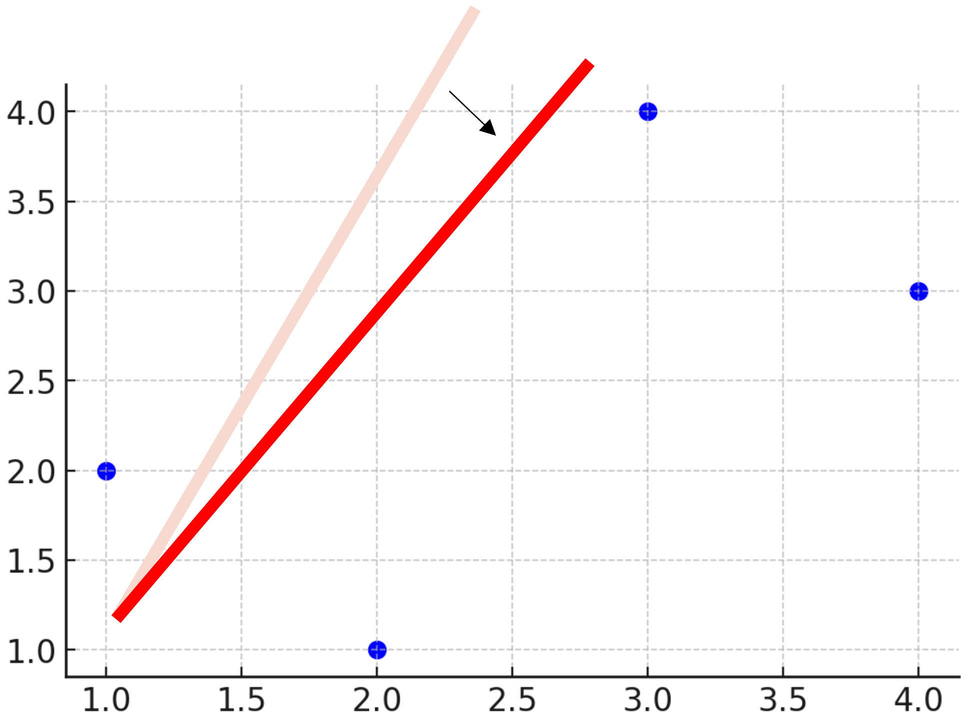
Example



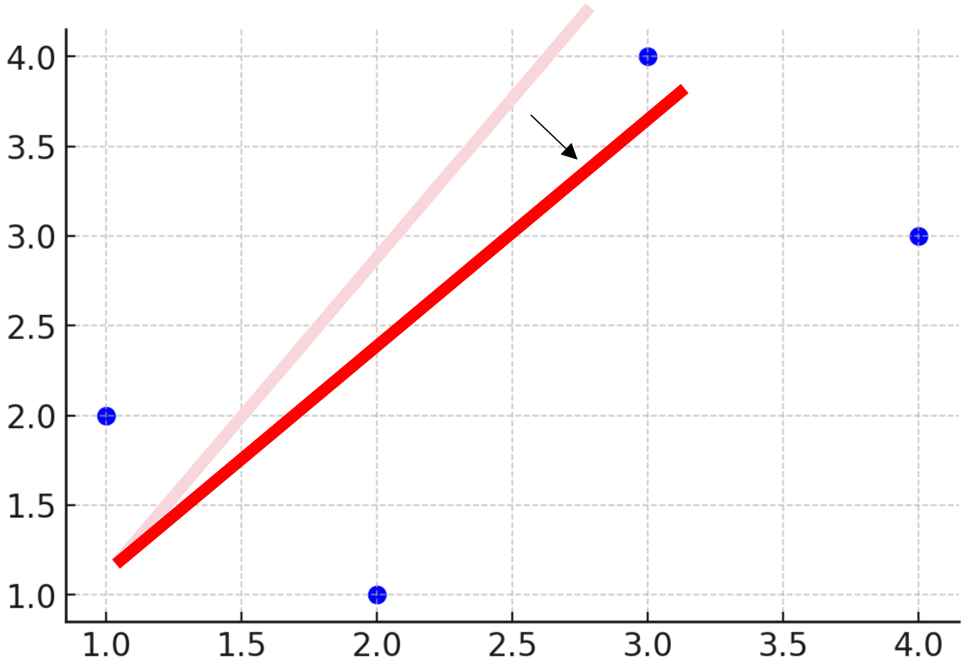
Example



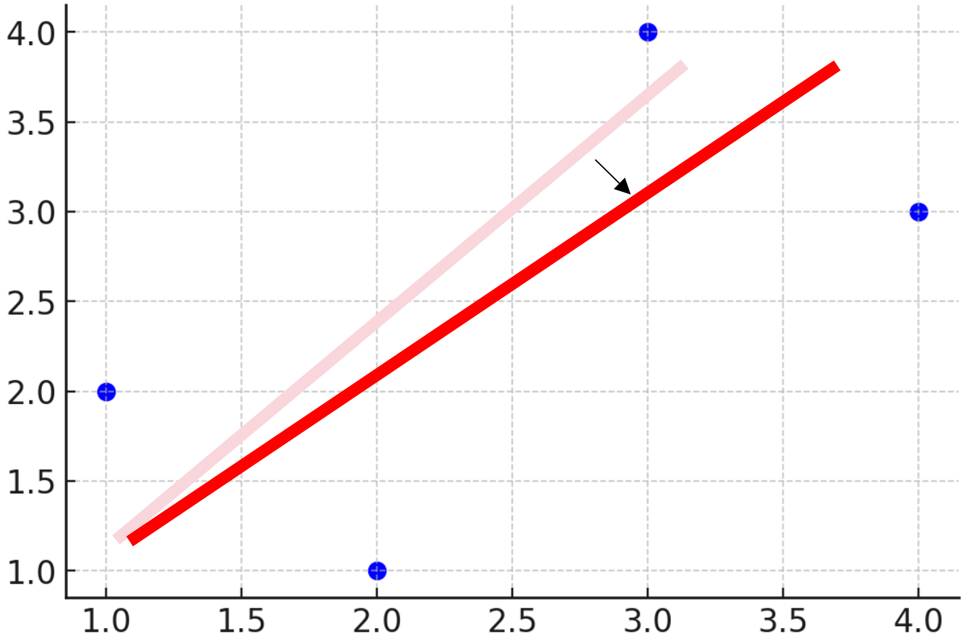
Example



Example



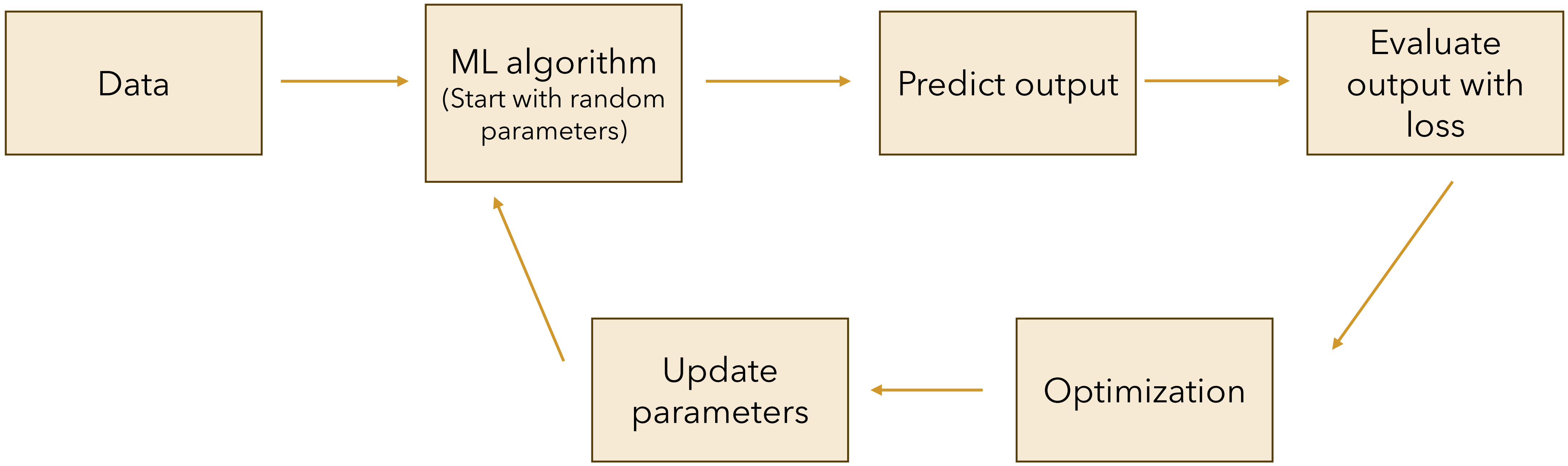
Example



Mathematically How?

- We should solve this mathematically
- But how?

Framework


Data
ML algorithm
(Start with random parameters)
Predict output
Evaluate output with loss
Optimization
Update parameters

Optimization Algorithms

- Gradient Descend
- Stochastic Gradient Descend
- Adam
- AdamW
- RMSProp
- Newton’s Method

Gradient Descend

- The problem was how we get to update the line in LR

Gradient Descend

- The problem was how we get to update the line in LR
- Gradient descend does this with calculating derivatives again!

Gradient Descend

- The problem was how we get to update the line in LR
- Gradient descend does this with calculating derivatives again!
- It updates parameters by moving in the opposite direction of their derivatives with respect to loss!

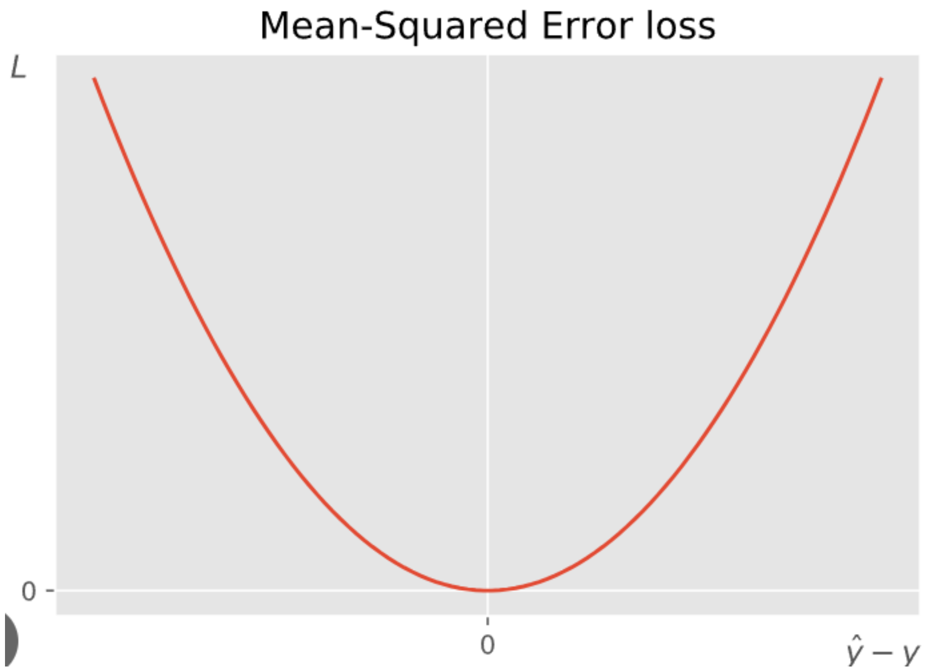
Gradient Descend

- The problem was how we get to update the line in LR
- Gradient descend does this with calculating derivatives again!
- It updates parameters by moving in the opposite direction of their derivatives with respect to loss!

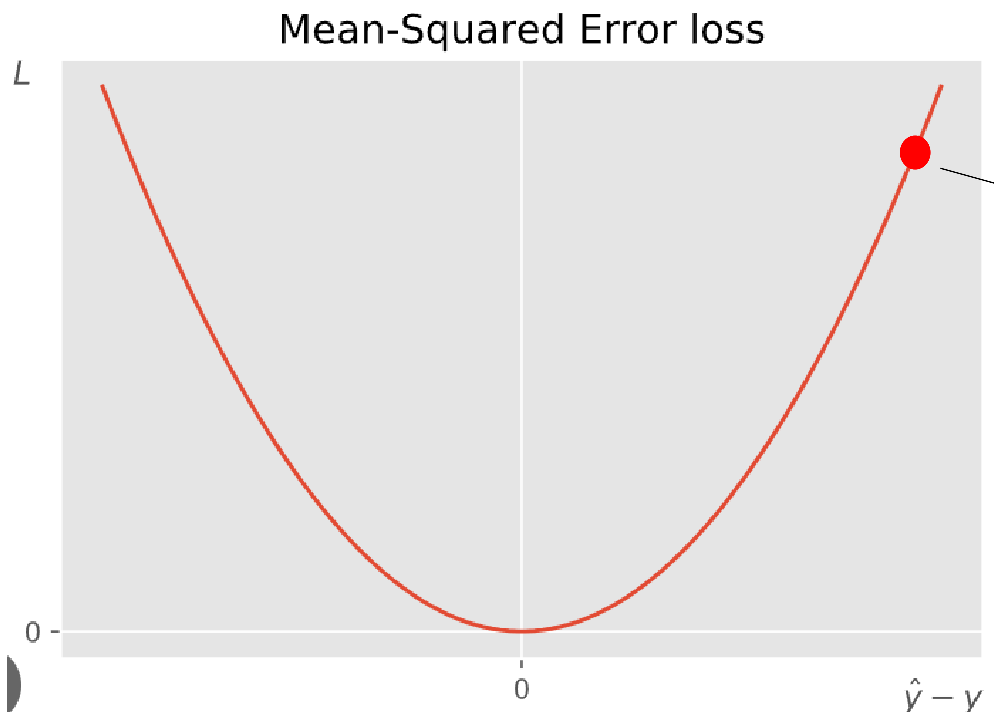
But what do we mean?

opposite direction of parameter derivatives with respect to loss!

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

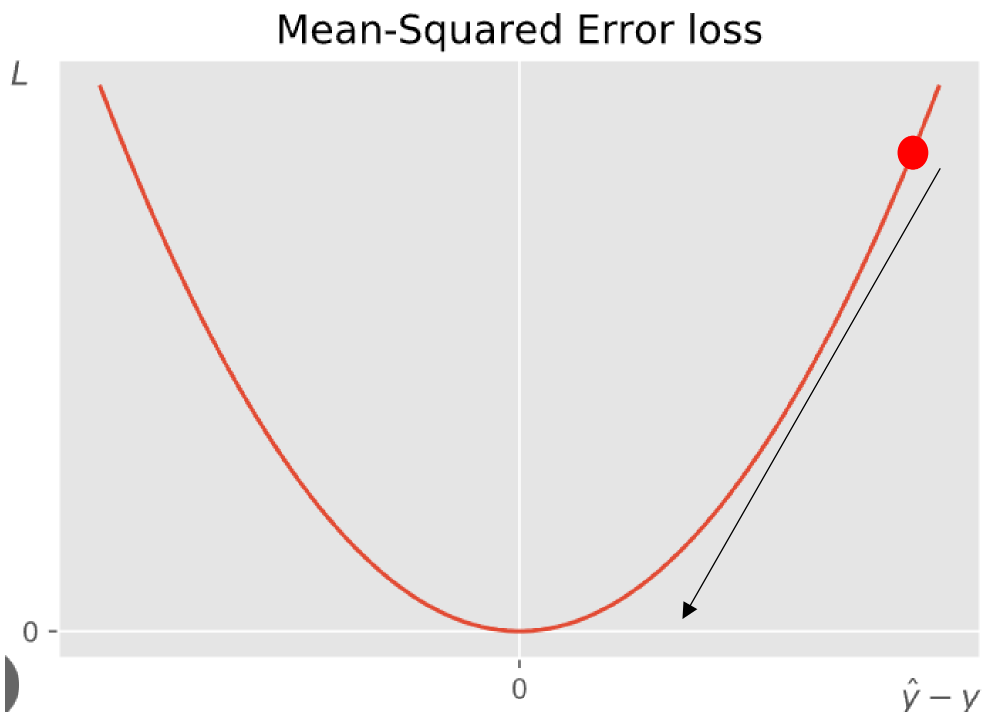


opposite direction of parameter derivatives with respect to loss!



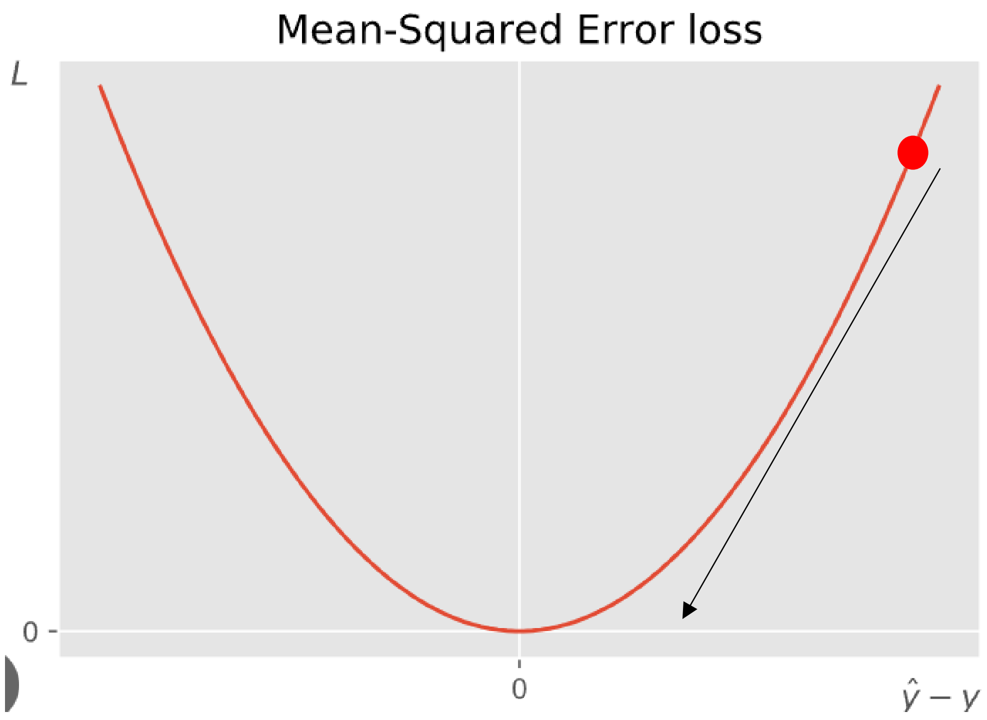
Our first average loss with the random line

opposite direction of parameter derivatives with respect to loss!



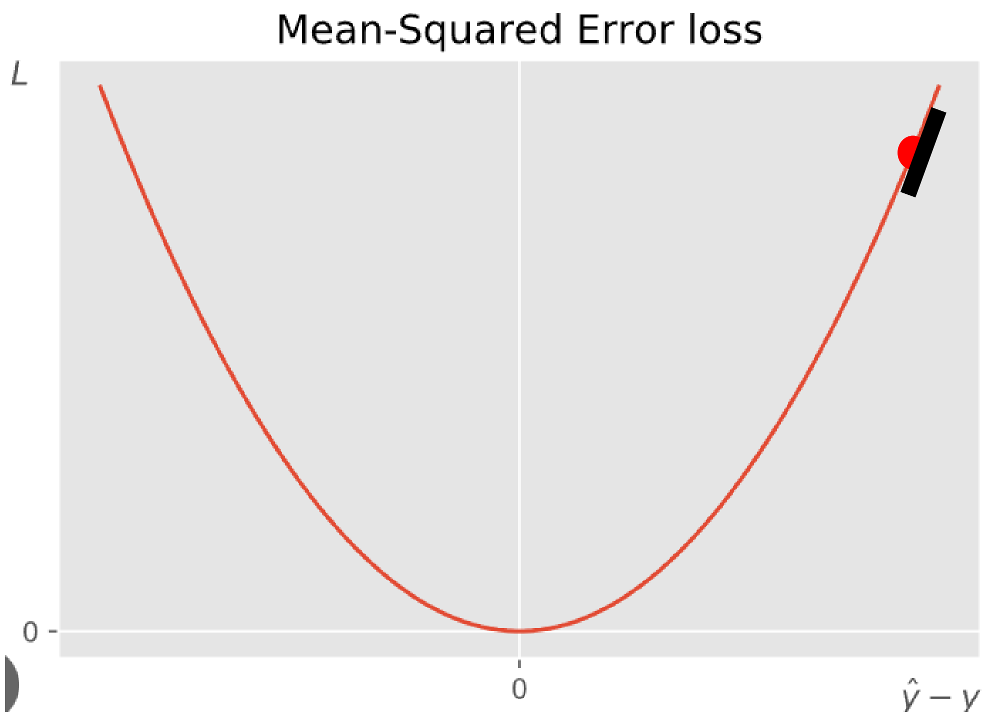
We want this loss to be as close as to 0

opposite direction of parameter derivatives with respect to loss!

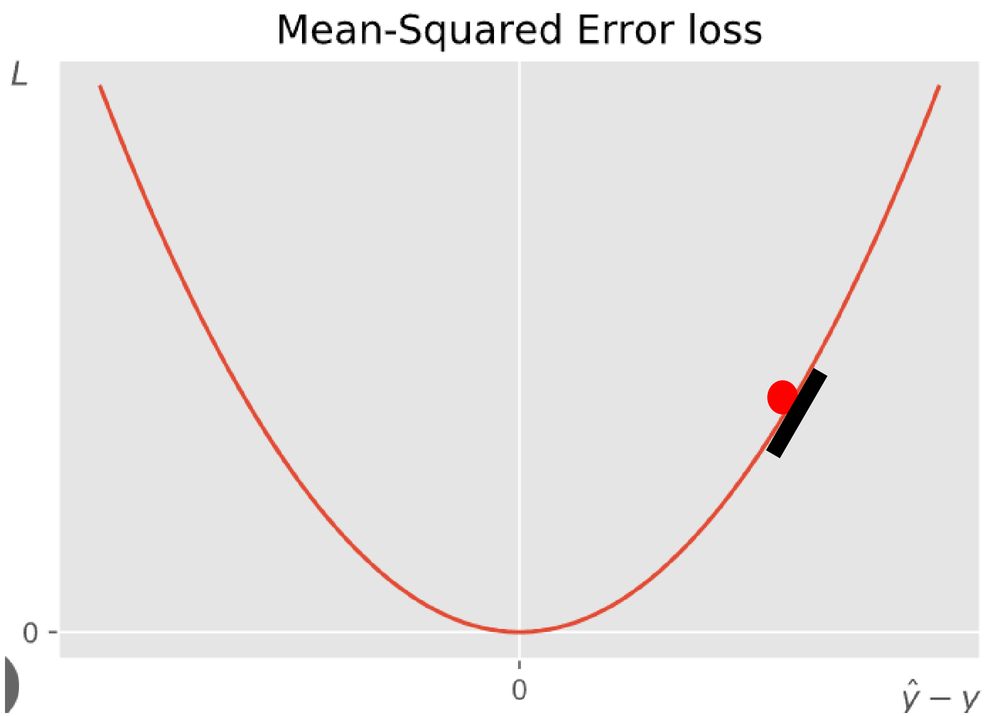


In order to move towards that direction, we have to move in the opposite direction of slope (derivate)

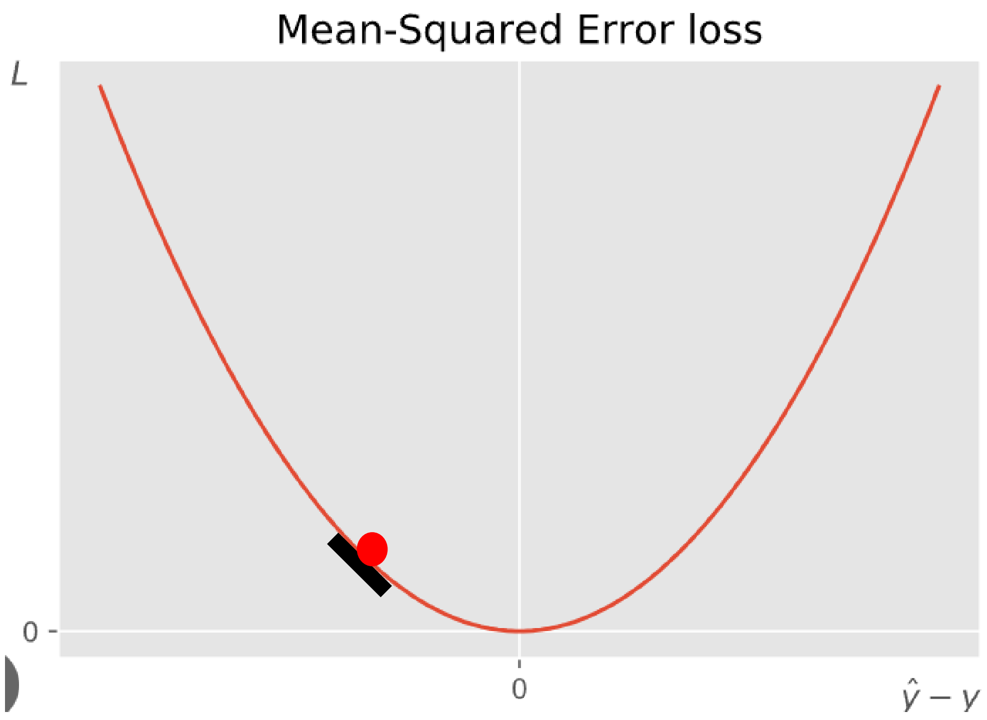
opposite direction of parameter derivatives with respect to loss!



opposite direction of parameter derivatives with respect to loss!

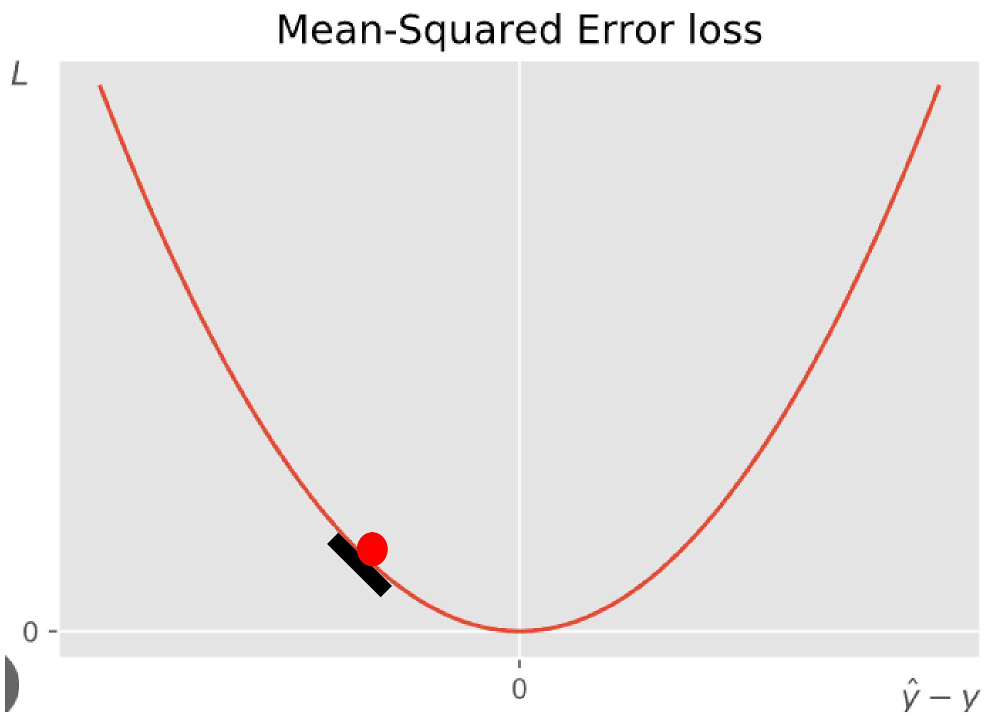


opposite direction of parameter derivatives with respect to loss!



$$w^{+} = w^{-} - \frac{\partial L}{\partial w}$$

opposite direction of parameter derivatives with respect to loss!



$$w^{+} = w^{-} - \alpha \frac{\partial L}{\partial w}$$

Now let's see mathematically!

- opposite direction of **parameter derivatives** with respect to loss!
- Loss is calculated:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Now let's see mathematically!

- opposite direction of **parameter derivatives** with respect to loss!
- Loss is calculated:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Given in data

2x+4

Now let's see mathematically!

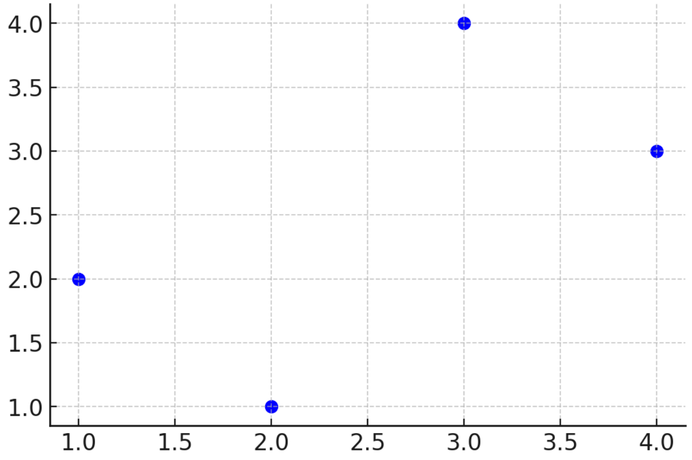
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Given in data

2x+4

Now let's see mathematically!

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

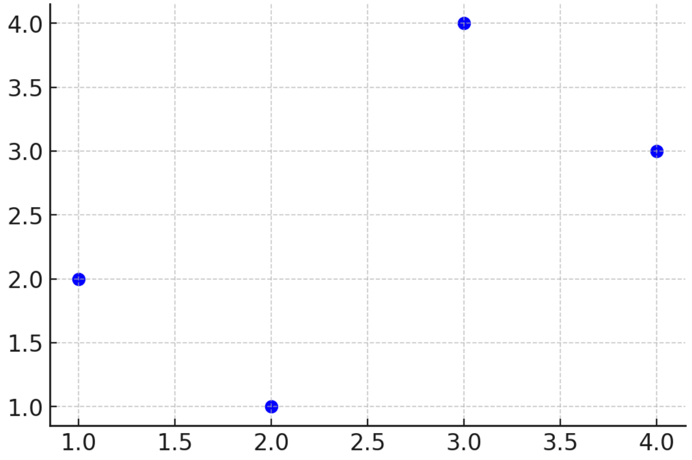


Data = (1,2), (2,1), (3,4), (4,3)
y' = 2x + 4

Loss = [$(2-6)^2$+ $(1-8)^2$+ $(4-10)^2$+ $(3-12)^2$] / 4 = [16 + 49 + 36 + 81] / 4 = 45.5

Now let's see mathematically!

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$



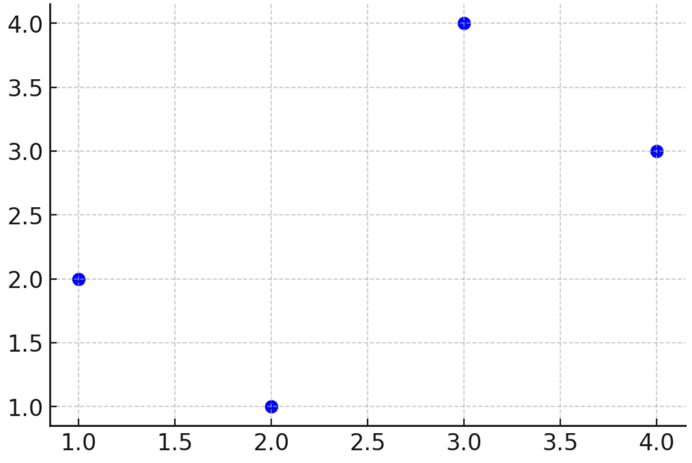
Data = (1,2), (2,1), (3,4), (4,3)
y' = 2x + 4

Loss = [$(2\text{-}6)^2$ + $(1\text{-}8)^2$ + $(4\text{-}10)^2$ + $(3\text{-}12)^2$] / 4 = [16 + 49 + 36 + 81] / 4 = 45.5

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y - (ax + b))^2$$

Now let's see mathematically!

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$



Data = (1,2), (2,1), (3,4), (4,3)
y' = 2x + 4

Loss = [$(2-6)^2$+ $(1-8)^2$+ $(4-10)^2$+ $(3-12)^2$] / 4 = [16 + 49 + 36 + 81] / 4 = 45.5

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

$$E = \frac{1}{n} \sum_{i=0}^{n} (y_i - (mx_i + c))^2$$

$$D_m = \frac{1}{n} \sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n} \sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

$$m = m - \alpha \times D_m$$

$$D_c = \frac{-2}{n} \sum_{i=0}^{n} (y_i - \bar{y}_i)$$

$$c = c - \alpha \times D_c$$

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

$$D_m = \frac{1}{n}\sum_{i=0}^{n}2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - \bar{y}_i)$$

$$m = m - \alpha \times D_m$$

new m = 2 - 0.01 * 2 = 0.02

$$D_c = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)$$

$$c = c - \alpha \times D_c$$

new c = 4 - 0.01 * 2 = 2.02

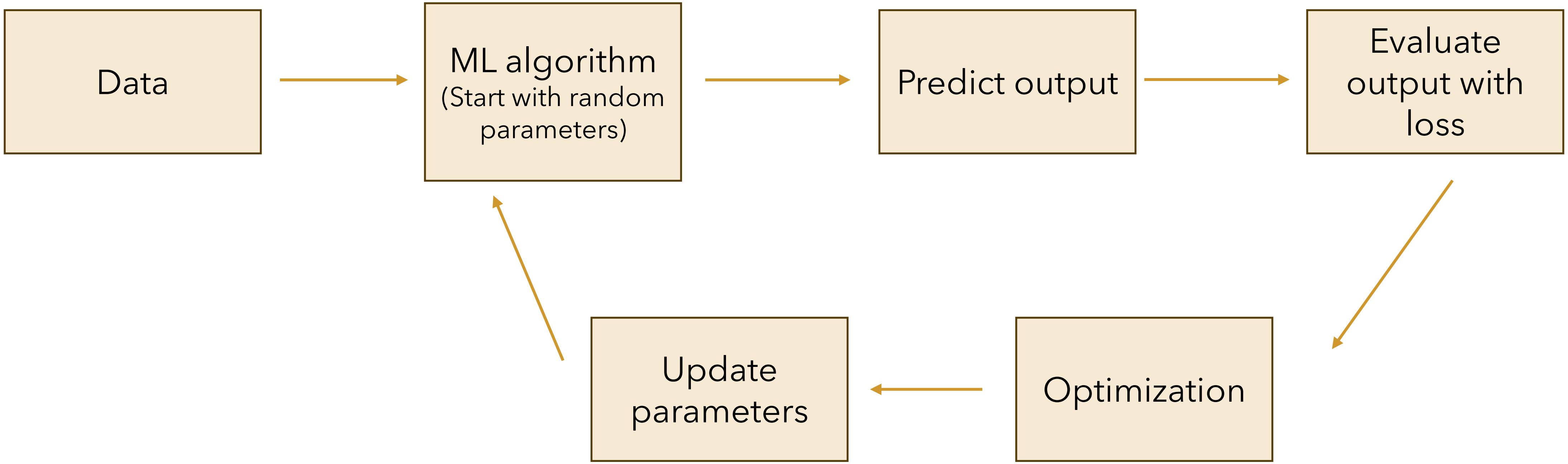
y = 0.02*x + 2.02

New line: $y = 0.02*x + 2.02$

Previous line: $y = 2x+4$

Best fitted line: $y = x + 0.6$

Framework


Data
ML algorithm
(Start with random parameters)
Predict output
Evaluate output with loss
Optimization
Update parameters

Code