UNIVERSITE DE NOUAKCHOTT
FACULTE DES SCIENCES ET TECHNIQUES
FEPARTEMENT INFORMATIQUE
SDD

Machine Learning

Simple Linear Regression

Dr. EL BENANY Mohamed Mahmoud

30.11.24

https://forms.office.com/r/yYL28qWb3m

Supervised Machine Learning

Régression

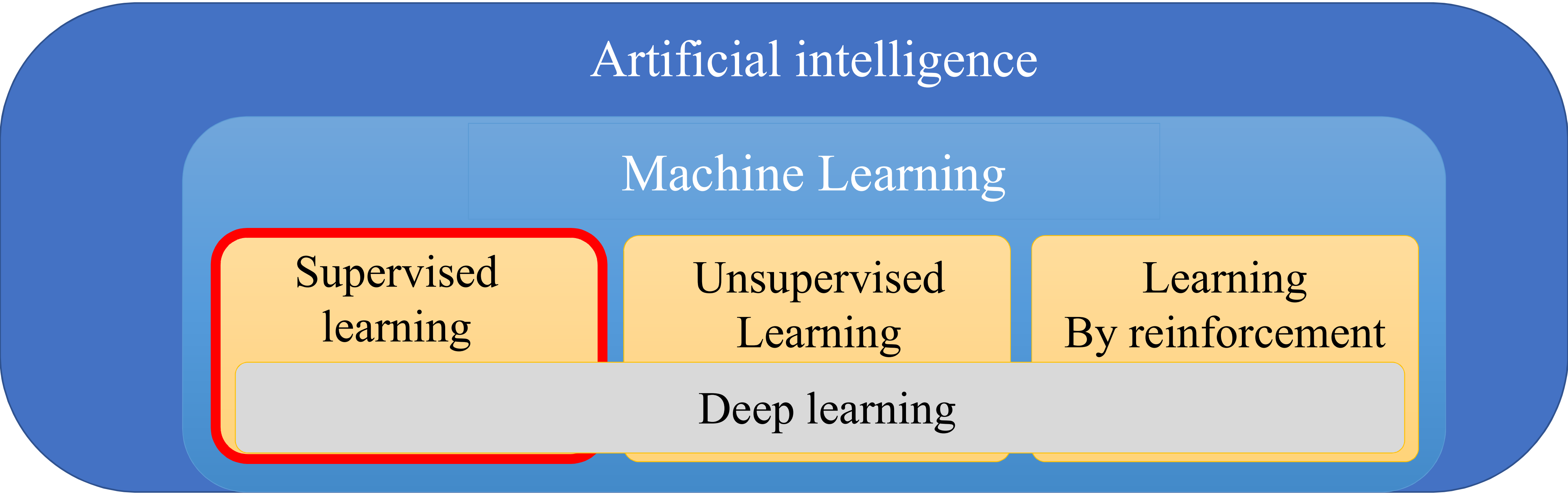
- Simple linear regression
- Multiple linear regression
- Ridge and Lasso Regression
- Polynomial regression
- Decision Tree and Random Forest Regression

Classification

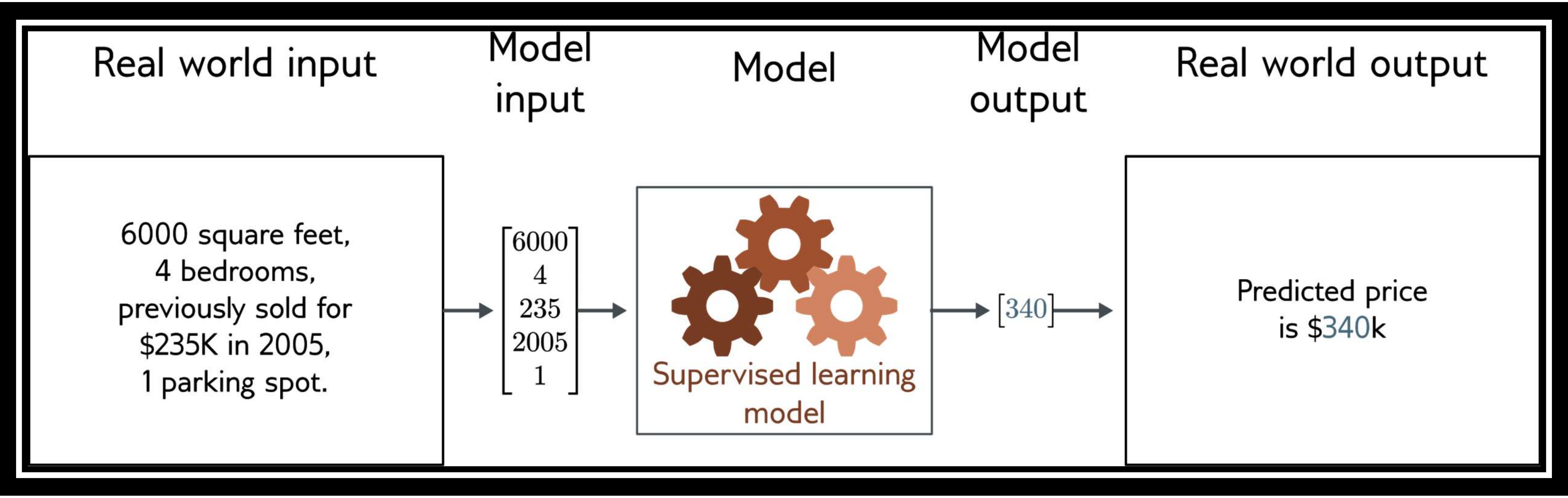
- Logistic Regression Classification
- Decision Tree Random Forest Classification VS Random
- Naive Bayesian Classifiers for Ranking
- Classification of support vector machines
- Vector Machine Kernel Support
- K-Nearest Neighbor Classifier
- Evaluation of classification models

UNSUPERVISED LEARNING

- **Clustering**
- **Dimensionality Reduction**
- **Data Analysis**


Artificial intelligence
Machine Learning
Supervised learning
Unsupervised Learning
Learning By reinforcement
Deep learning

Regression



Univariate regression problem (one output, real value)

Supervised learning overview

Supervised learning model = mapping from one or more inputs to one or more outputs

- Model is a mathematical equation

- Computing the outputs from the inputs = inference

- Example:
  - Input is age and milage of secondhand Toyota Prius
  - Output is estimated price of car

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a mathematical equation

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- ~~Model is a mathematical equation~~
- Model is a family of equations
- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a family of equations
- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Training a model = finding parameters that predict outputs "well" from inputs for a training dataset of input/output pairs

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Notation

- Input:

$$\mathbf{x}$$

Variables always Roman letters

Normal = scalar
Bold = vector
Capital Bold = matrix

- Output:

$$\mathbf{y}$$

- Model:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]$$

Functions always square brackets

Normal = returns scalar
Bold = returns vector
Capital Bold = returns matrix

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Notation example

- Input:

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{mileage} \end{bmatrix}$$

Structured or tabular data

- Output:

$$y = \begin{bmatrix} \text{price} \end{bmatrix}$$

- Model:

$$y = \mathrm{f}[\mathbf{x}]$$

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Model

- Parameters:

$$\phi$$

Parameters always Greek letters

- Model :

$$\mathbf{y} = \mathbf{f}[\mathbf{x}, \phi]$$

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Loss function

- Training dataset of I pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

$$L\left[\phi, \mathrm{f}\left[\mathbf{x}_i, \phi\right], \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}\right]$$

or for short:

$$L\left[\phi\right] \longleftarrow$$

Returns a scalar that is smaller when model maps inputs to outputs better

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Training

- Loss function:

$$L[\phi]$$

Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[L[\phi] \right]$$

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Testing

- To test the model, run on a separate test dataset of input / output pairs
- See how well it generalizes to new data

Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

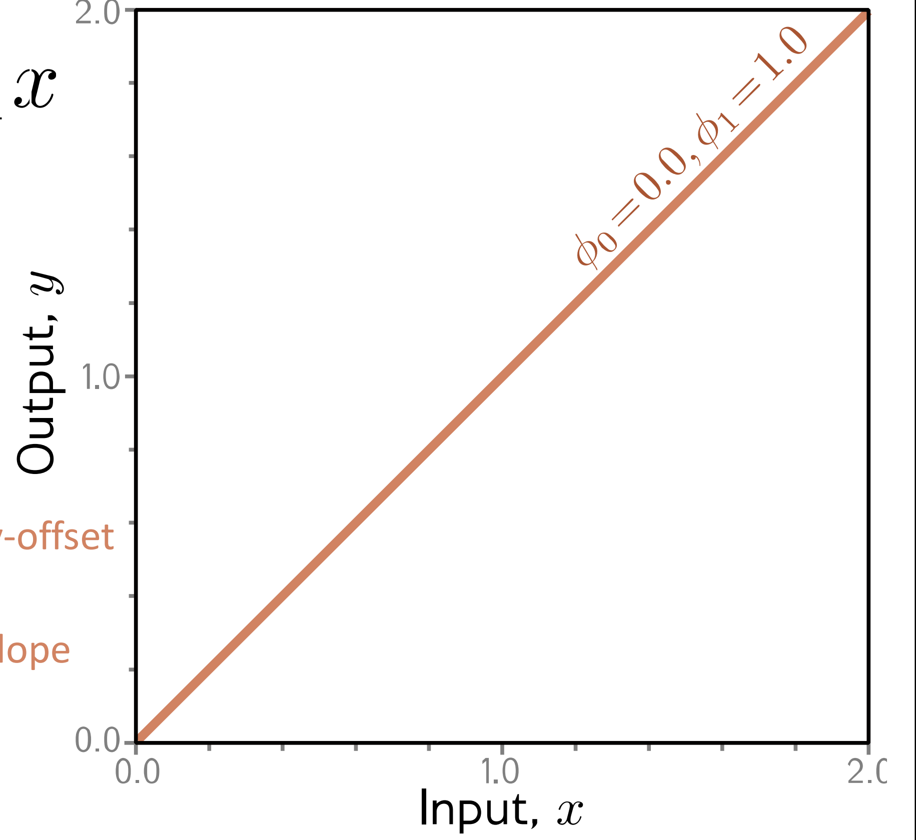
1D Linear regression example

- Model:

$$y = \mathrm{f}[x, \boldsymbol{\phi}] = \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{matrix} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{matrix}$$



Supervised learning

- Overview
- Notation
    - Model
    - Loss function
    - Training
    - Testing
- 1D Linear regression example
    - Model
    - Loss function
    - Training
    - Testing
- Where are we going?

1D Linear regression example

- Model:

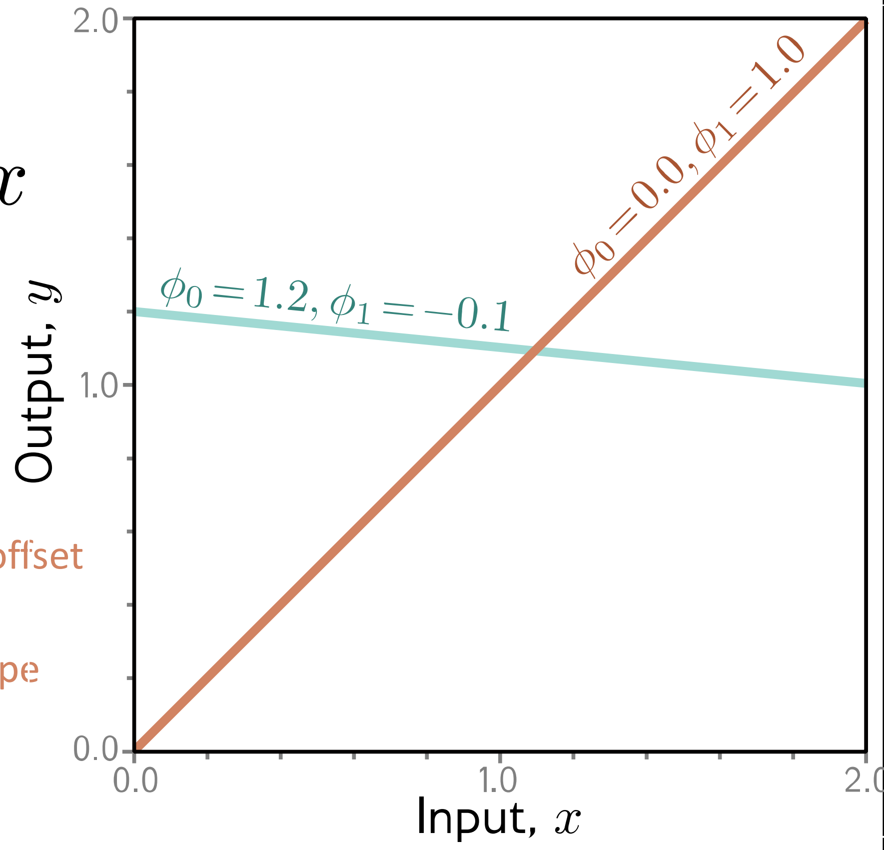
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

y-offset

slope



Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

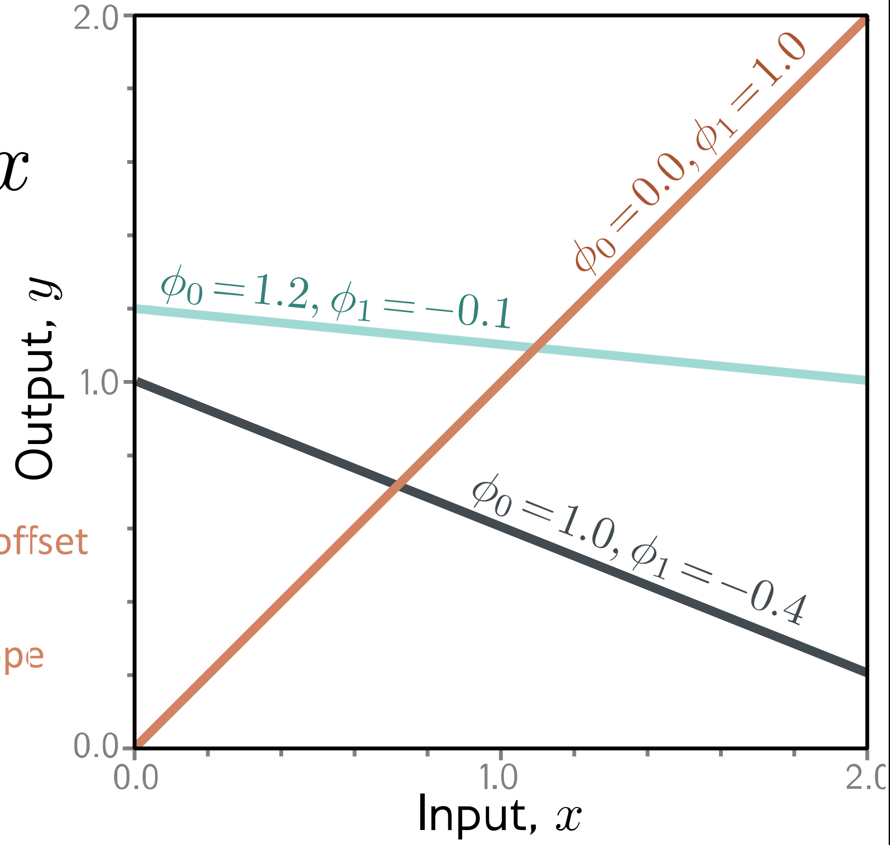
1D Linear regression example

- Model:

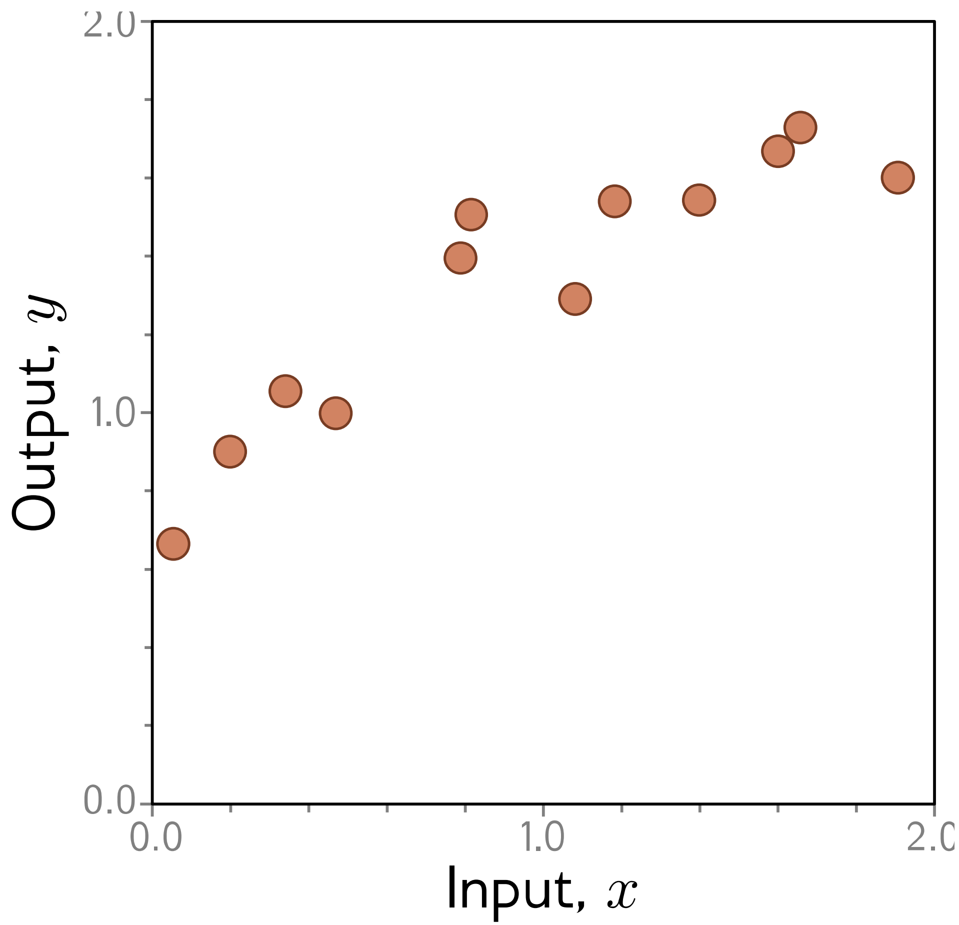
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{matrix} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{matrix}$$



1D Linear regression example



Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Supervised learning

- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

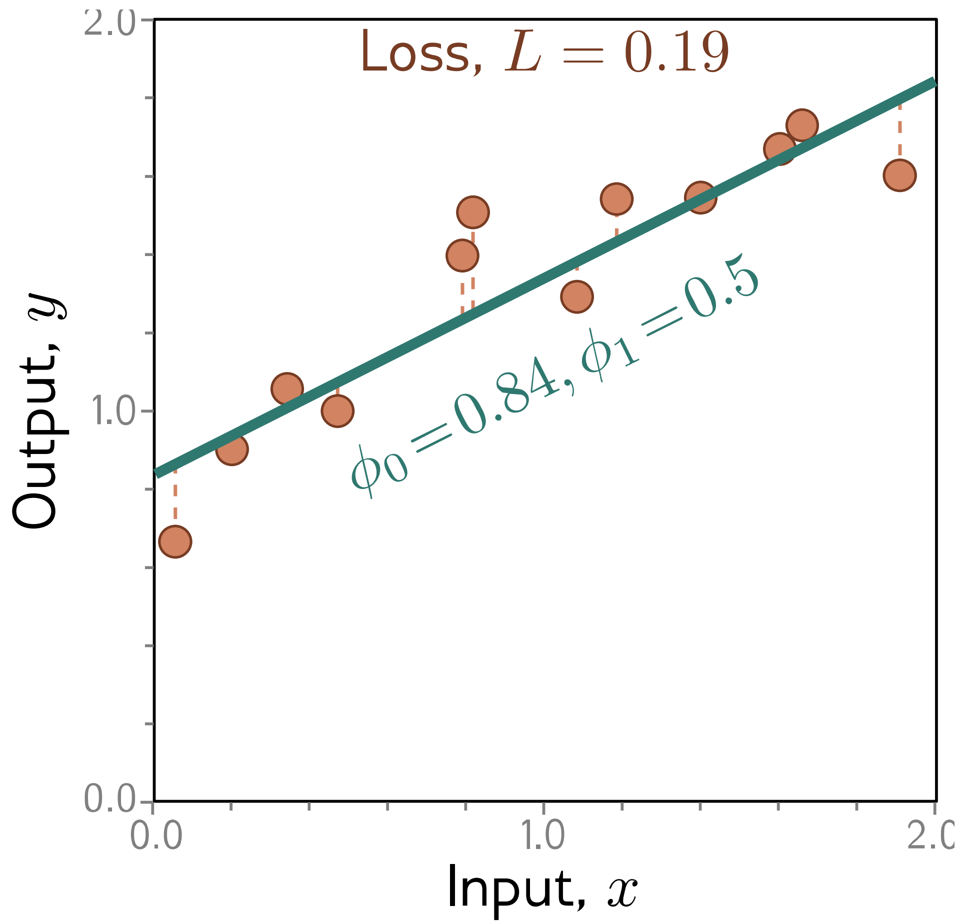
1D Linear regression example

Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"



Supervised learning

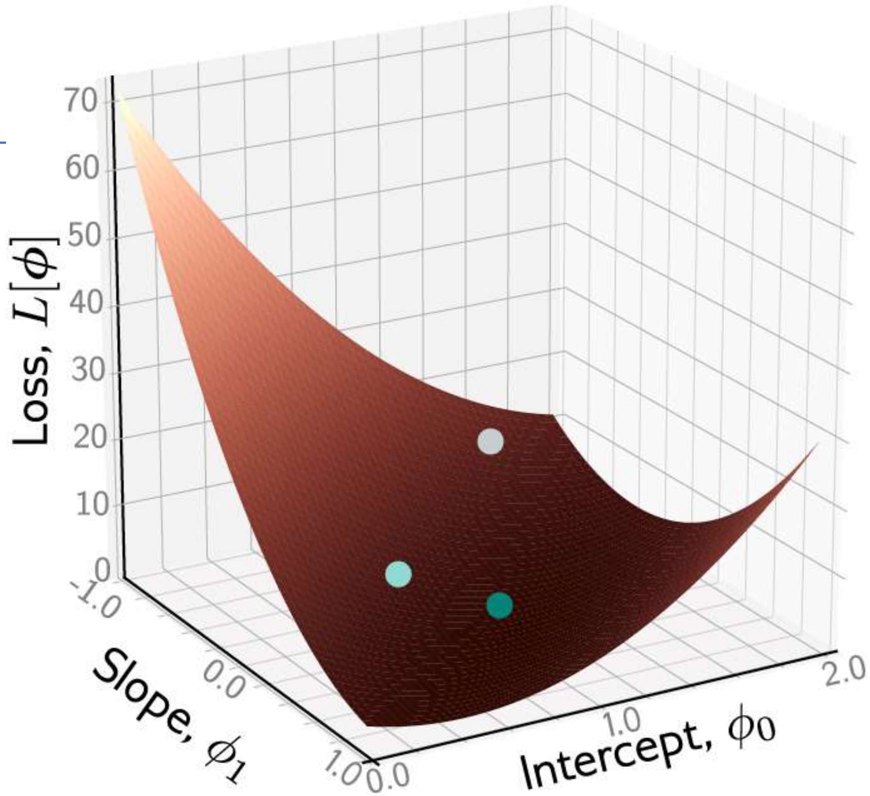
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

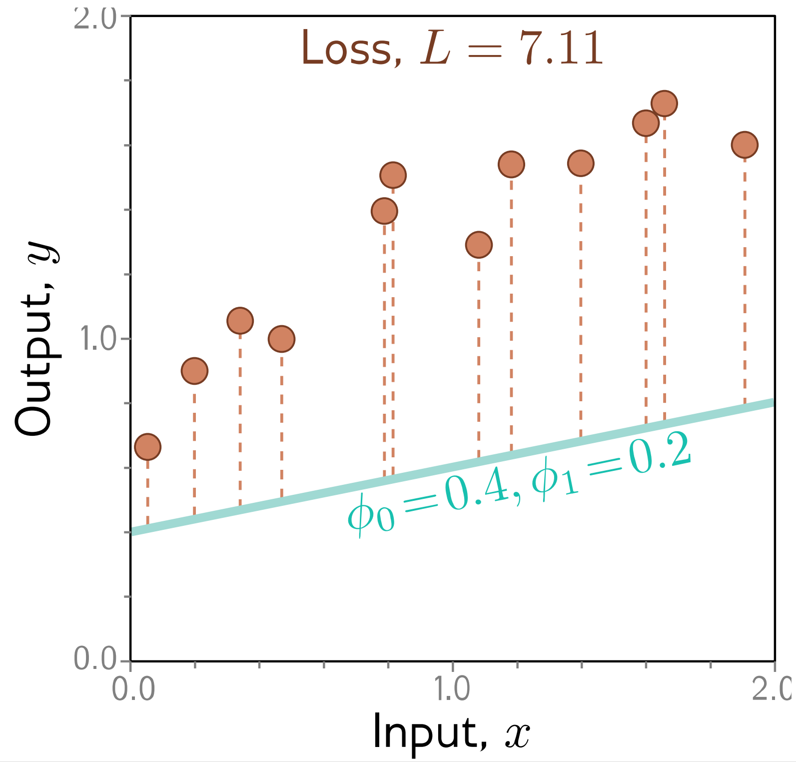
1D Linear regression example

Loss function:

$$L[\phi] = \sum_{i=1}^{I}(\mathrm{f}[x_i, \phi] - y_i)^2$$
$$= \sum_{i=1}^{I}(\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"





Supervised learning

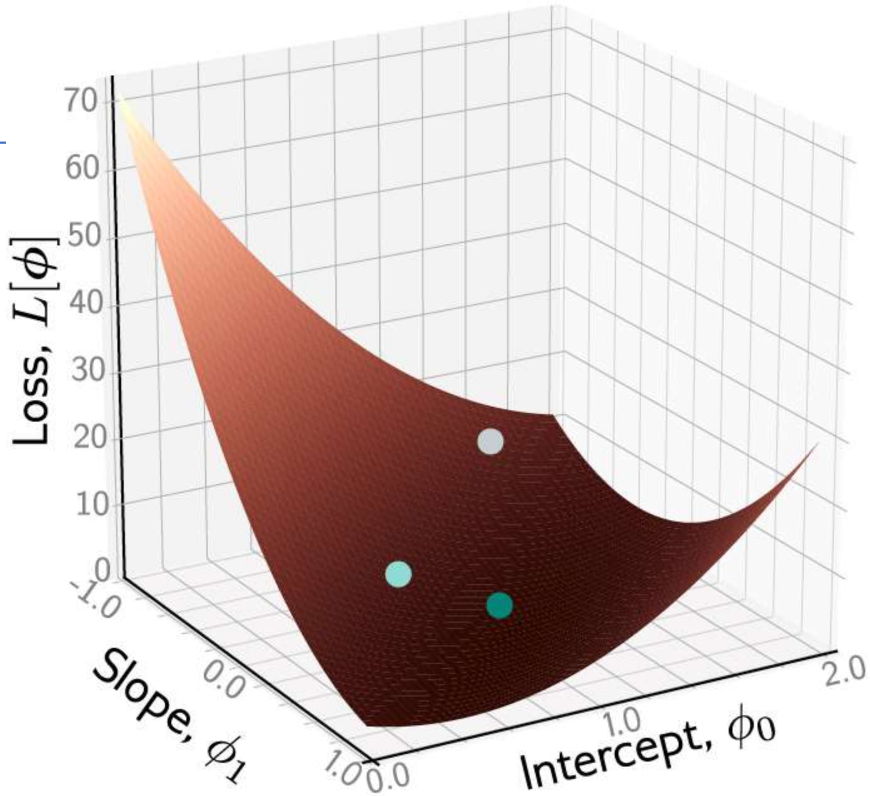
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

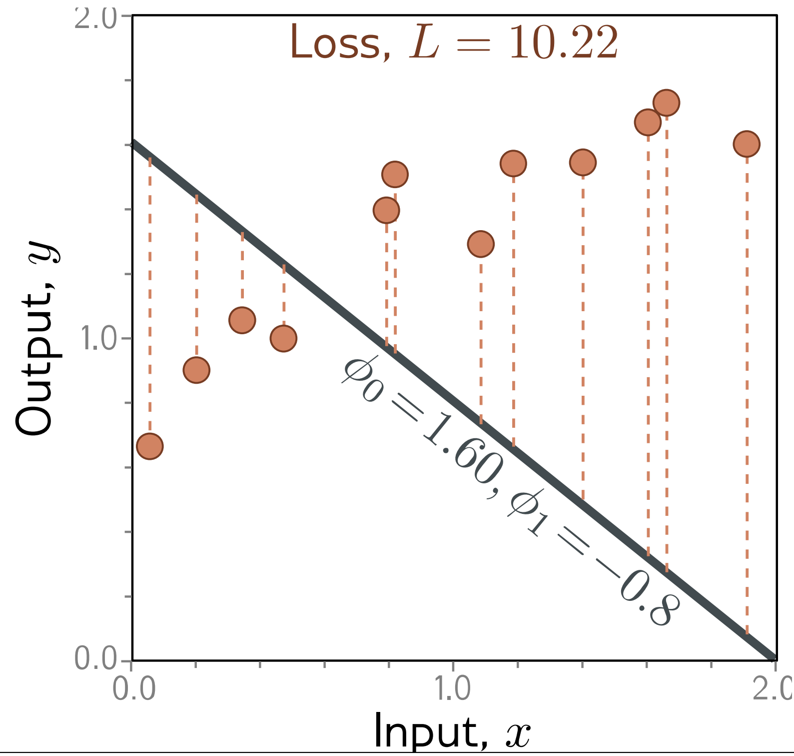
1D Linear regression example

Loss function:

$$L[\phi] = \sum_{i=1}^{I}(\mathrm{f}[x_i, \phi] - y_i)^2$$
$$= \sum_{i=1}^{I}(\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"





Supervised learning

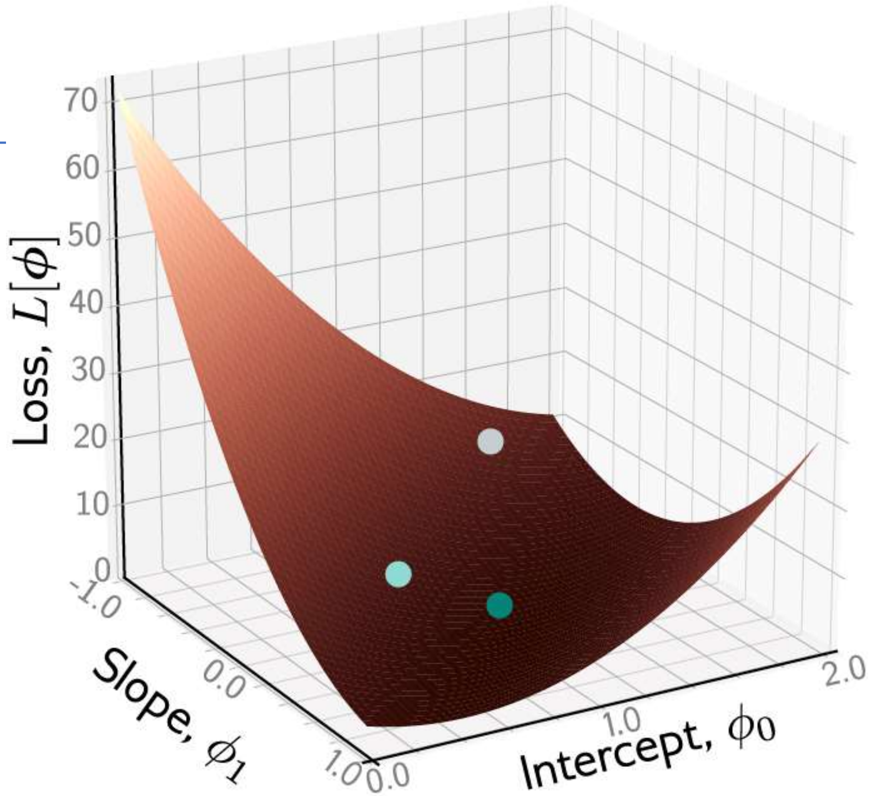
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

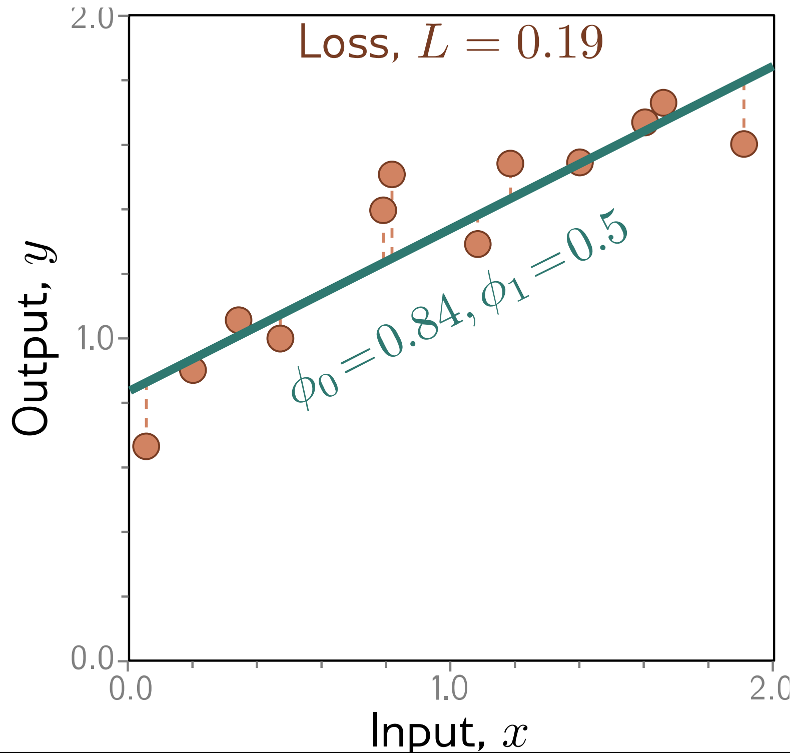
1D Linear regression example

Loss function:

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"





Supervised learning

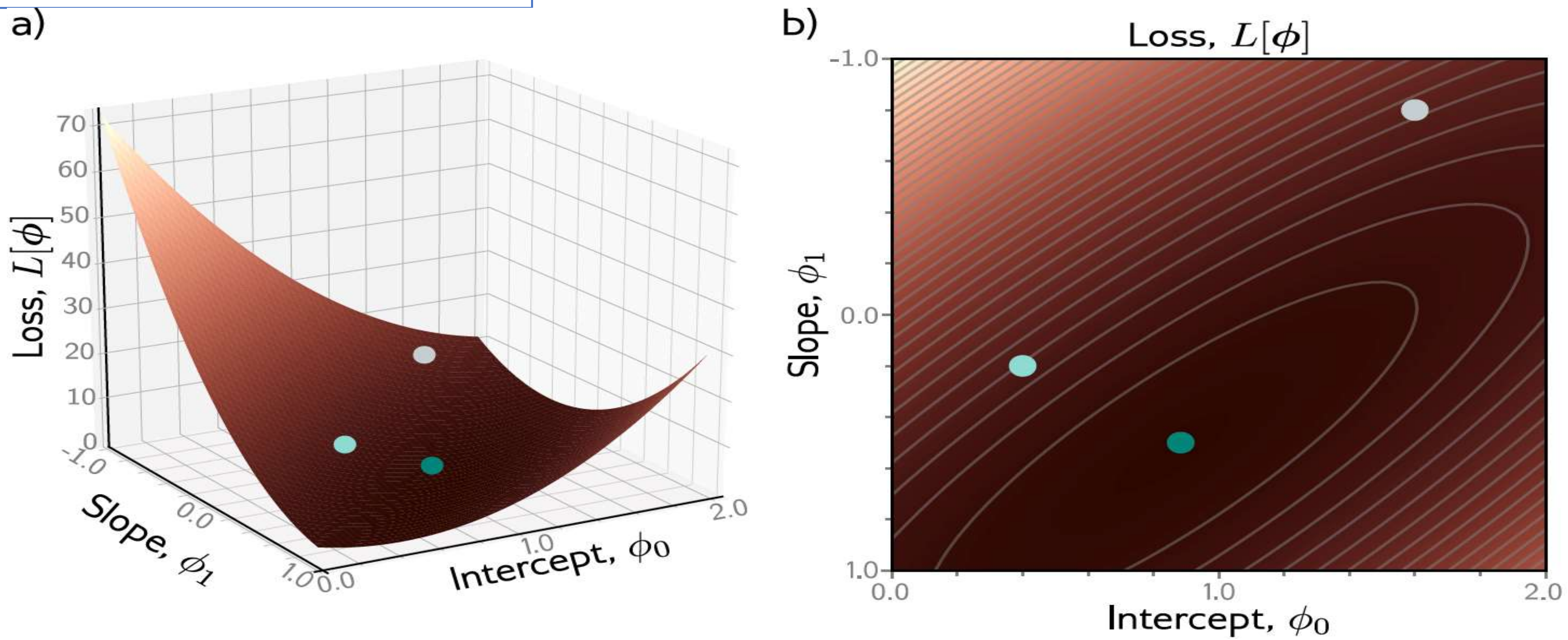
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

1D Linear regression example

Loss function:

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

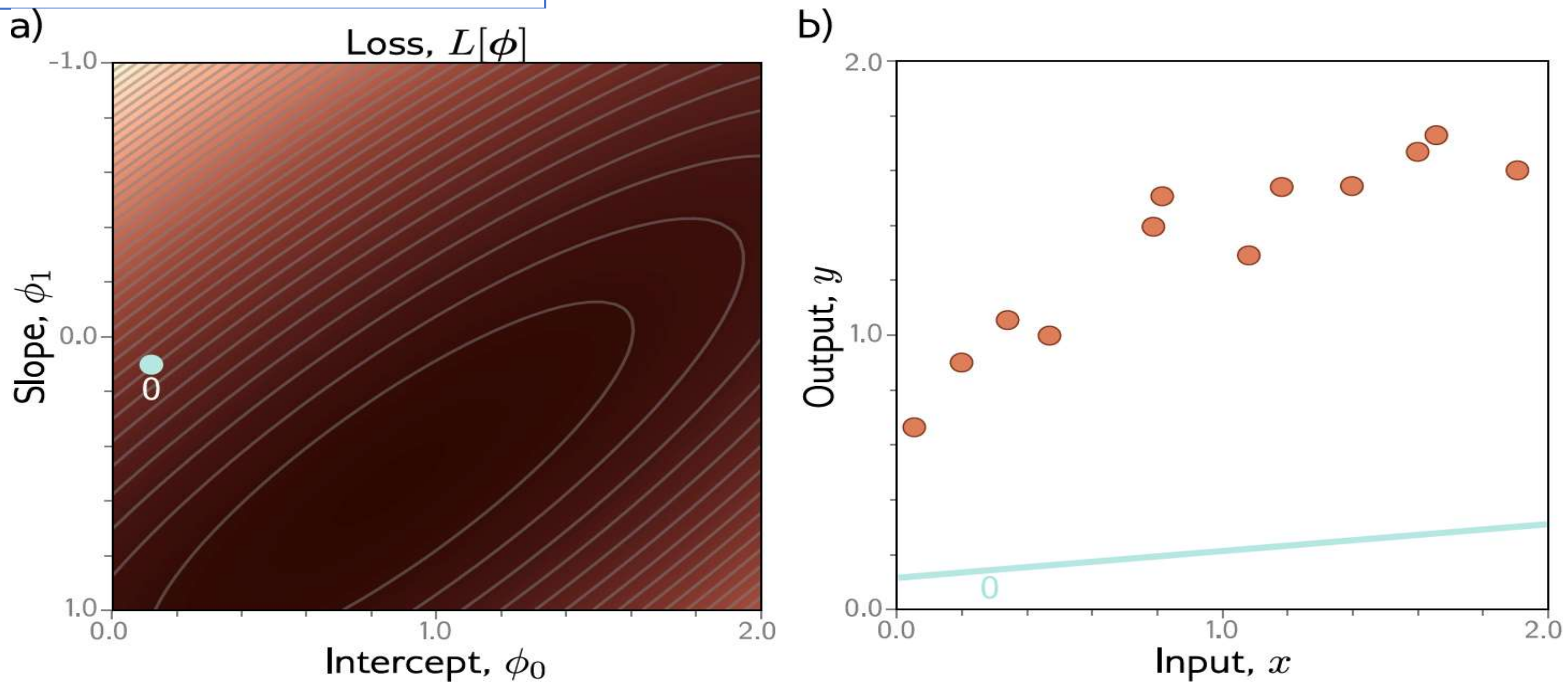
"Least squares loss function"



Supervised learning

Example: 1D Linear regression training

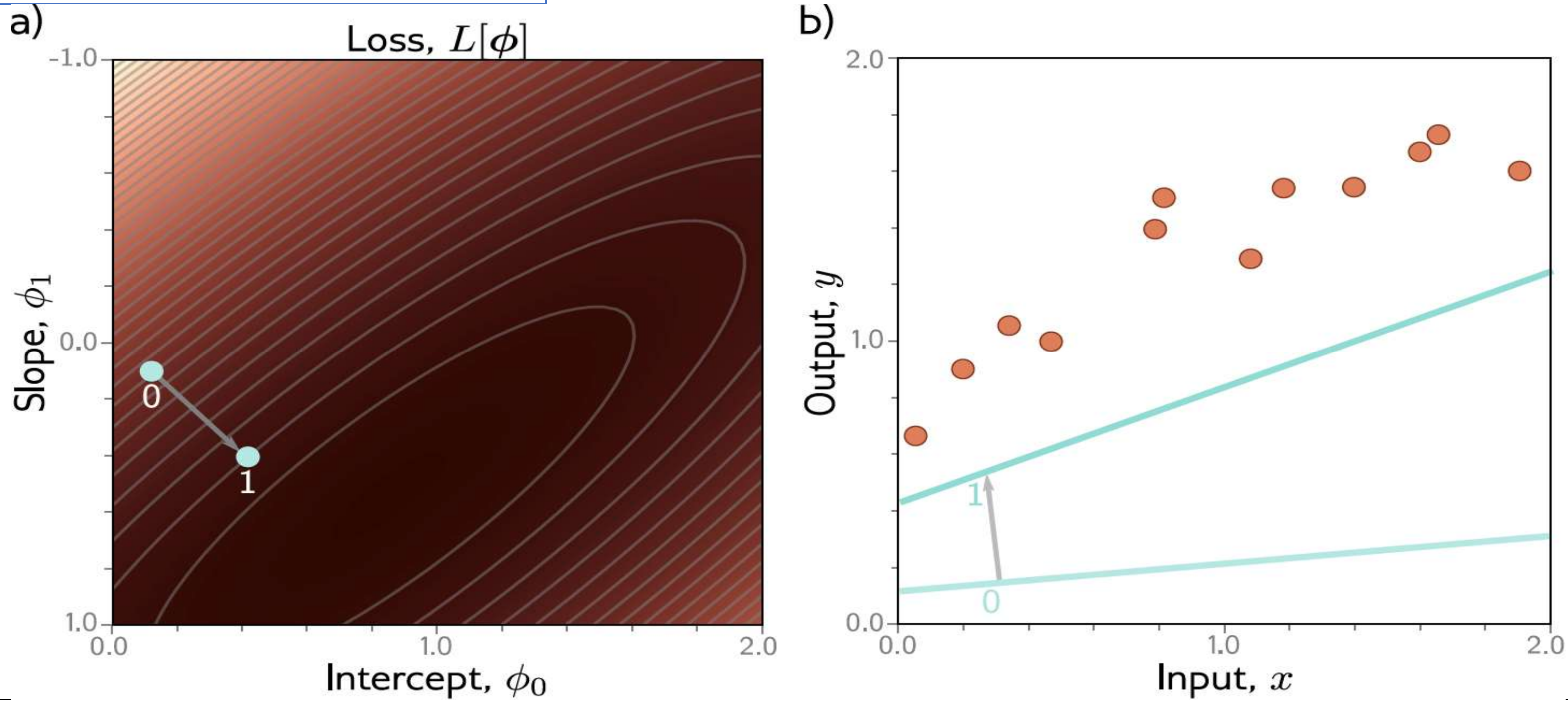
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing



Supervised learning

Example: 1D Linear regression training

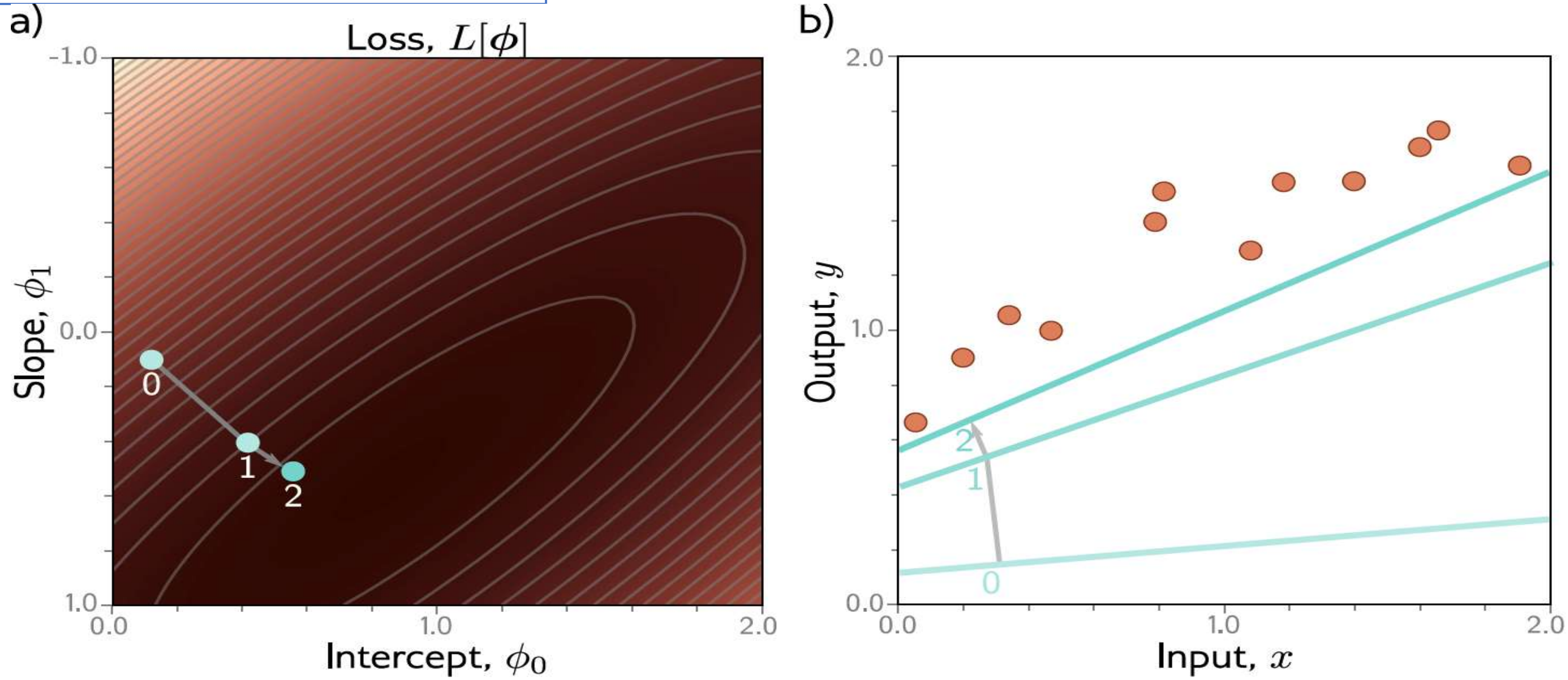
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing



Supervised learning

Example: 1D Linear regression training

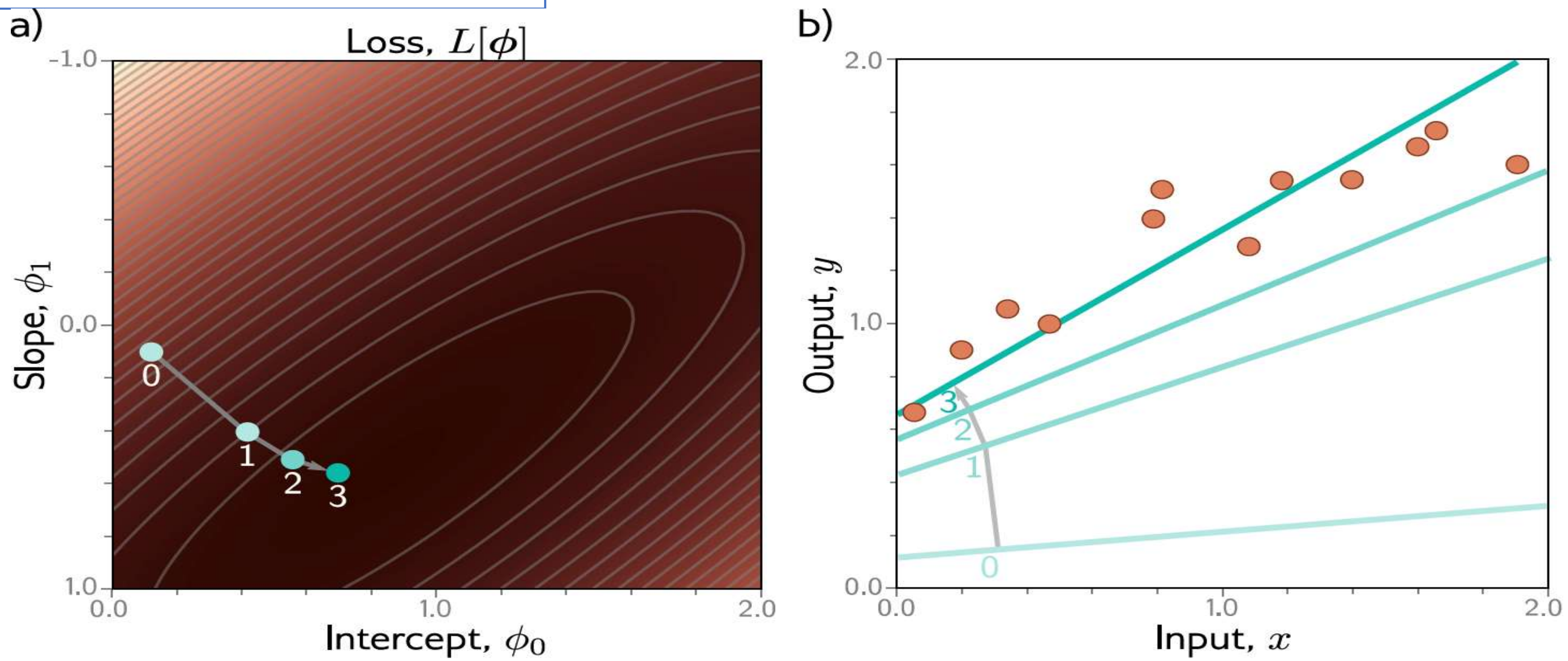
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing



Supervised learning

Example: 1D Linear regression training

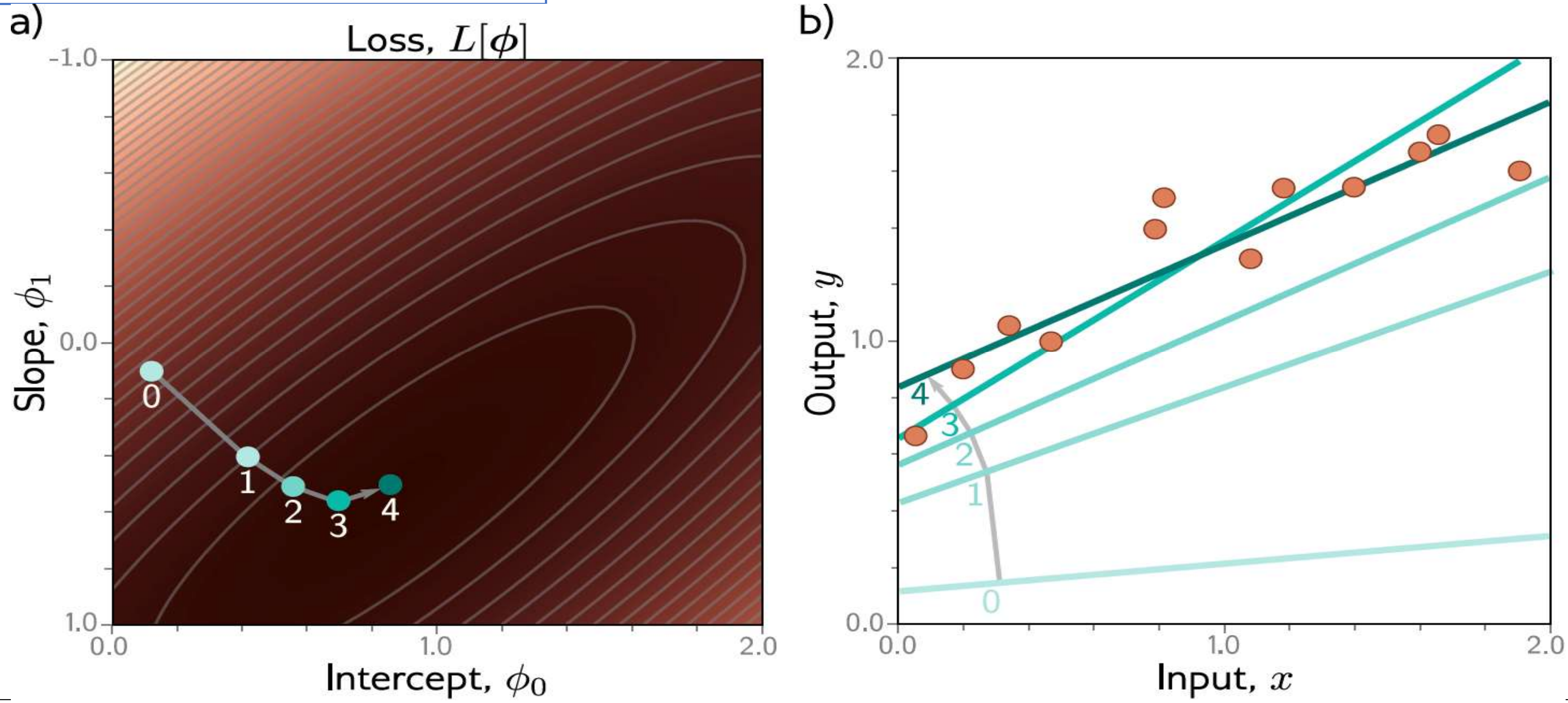
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing



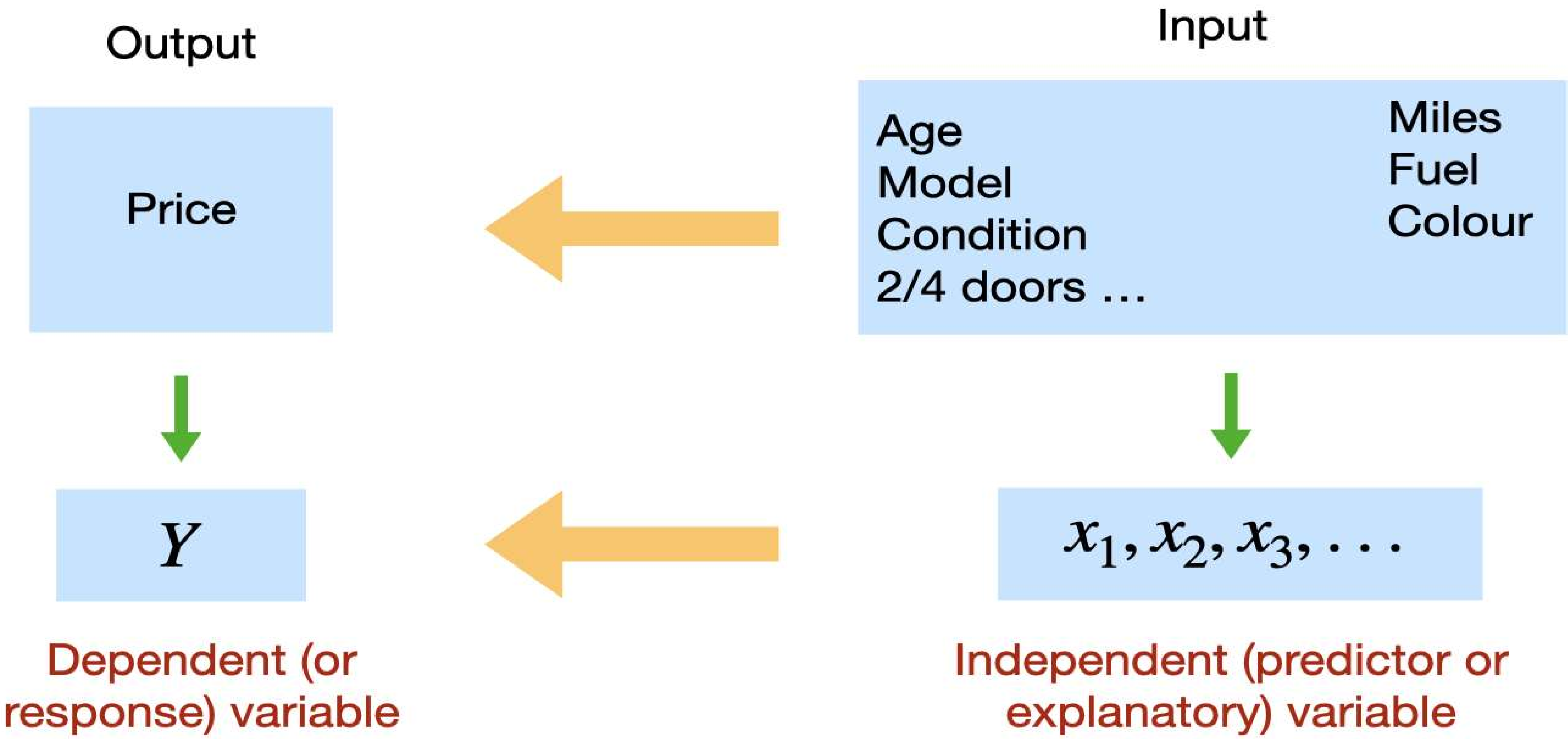
Supervised learning

- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing

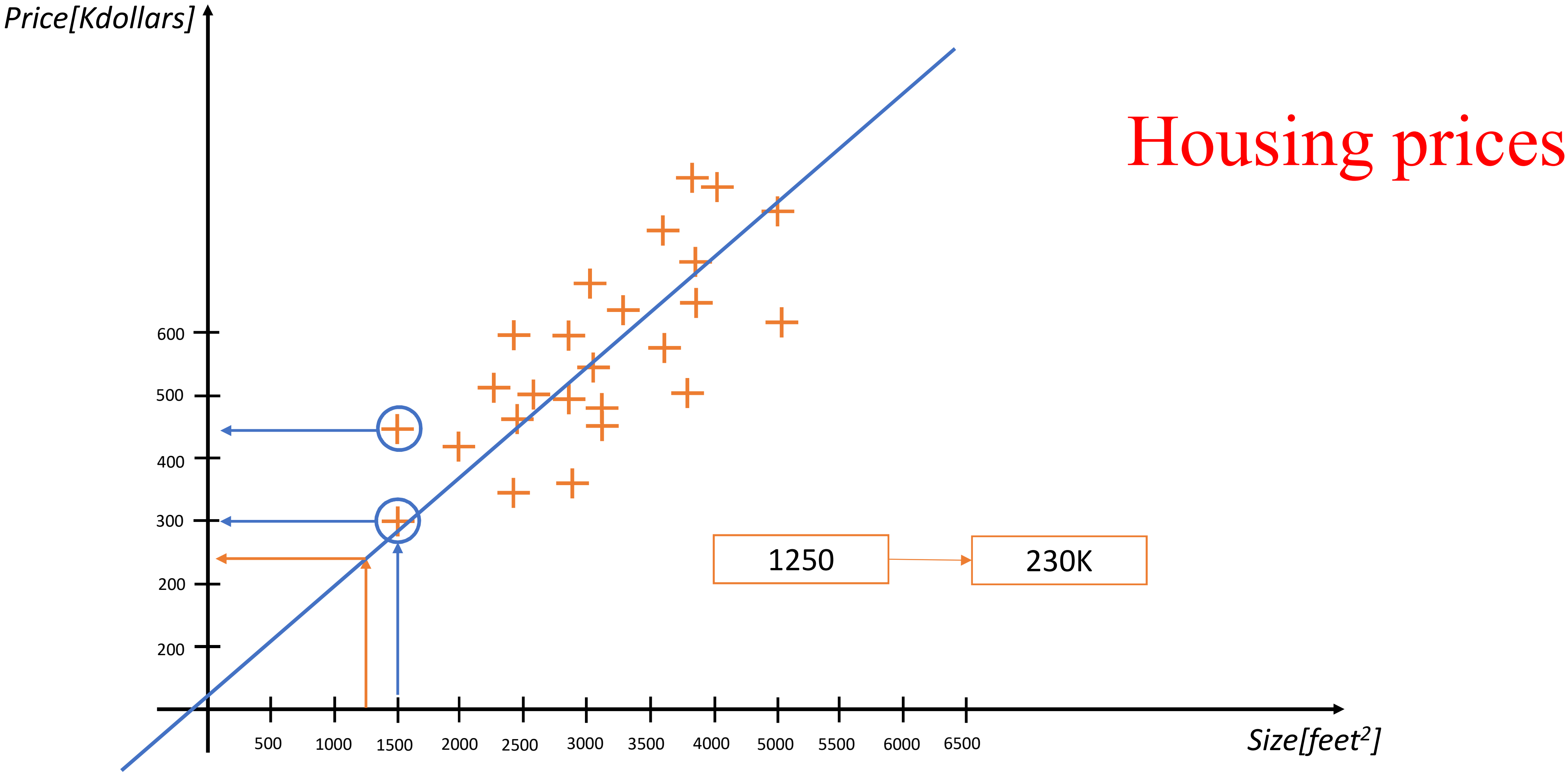
Example: 1D Linear regression training



Linear regression variables



Simple Linear Regression

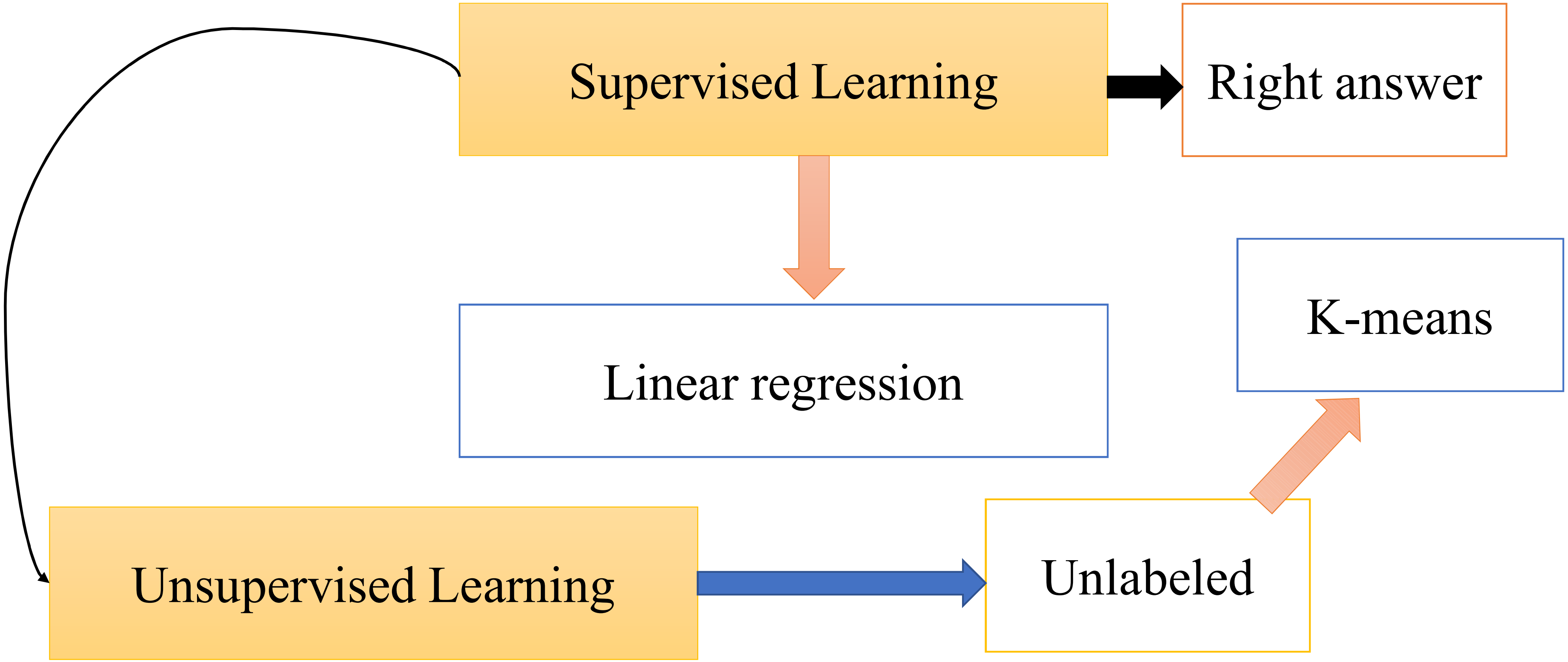


Supervised Learning

- Linear regression:
  - Real values:1.2, 2.5, 3.8, 4.2

- Classification:
  - Discrete values: 1, 2, 3, 4

Machine Learning



Terminologies: Training Data

X

- Input variable
- Features
- Independent variable

Size[feet^2]	*Price in K$*
51000	600
48000	300
38000	403
11000	210
…	…

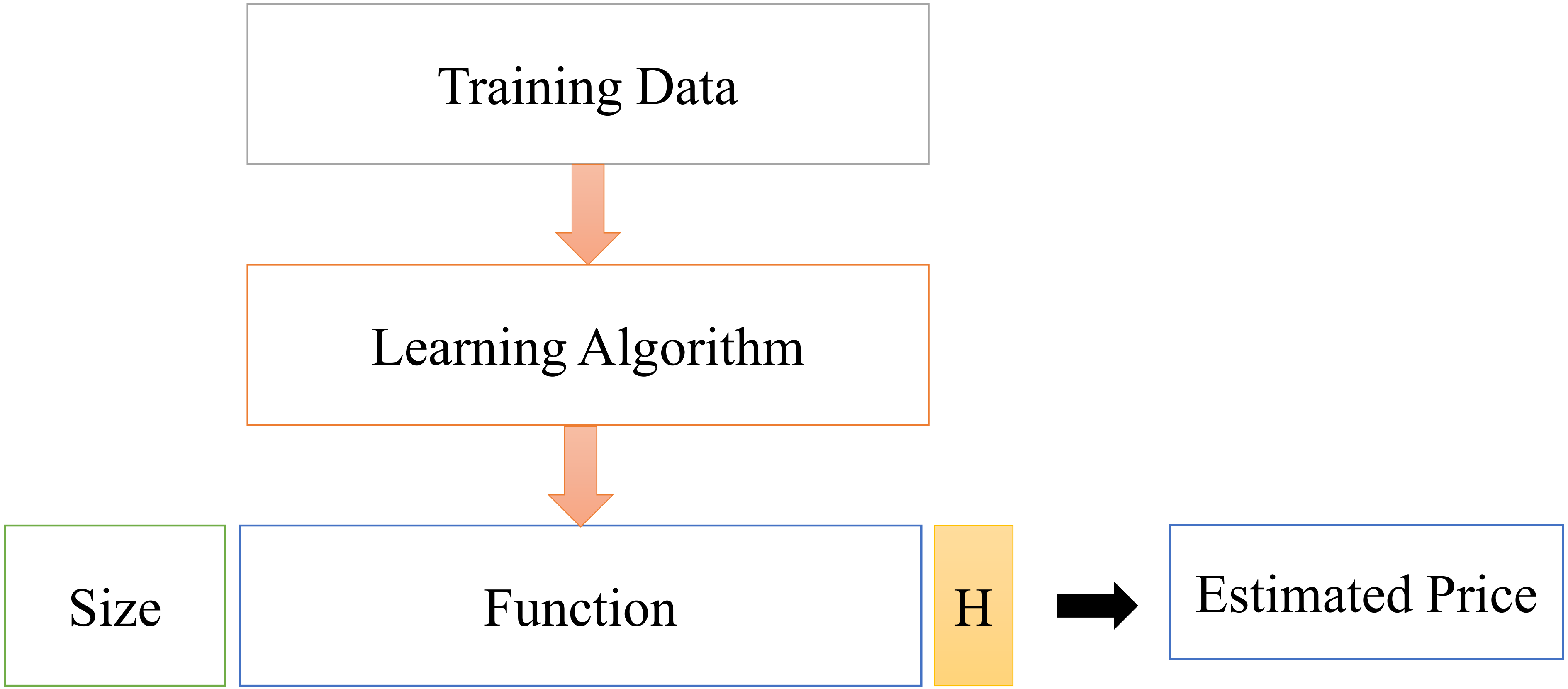
M
50

Y

- Output variable
- Target
- Dependent variable

X^i & Y^i | $X^1 = 51000$ | $X^2 = 48000$ | $Y^1 = 600$ …

Simple Linear Regression
(Housing Data Analysis)



Univariate Linear Regression

Dependent variable

y-offset

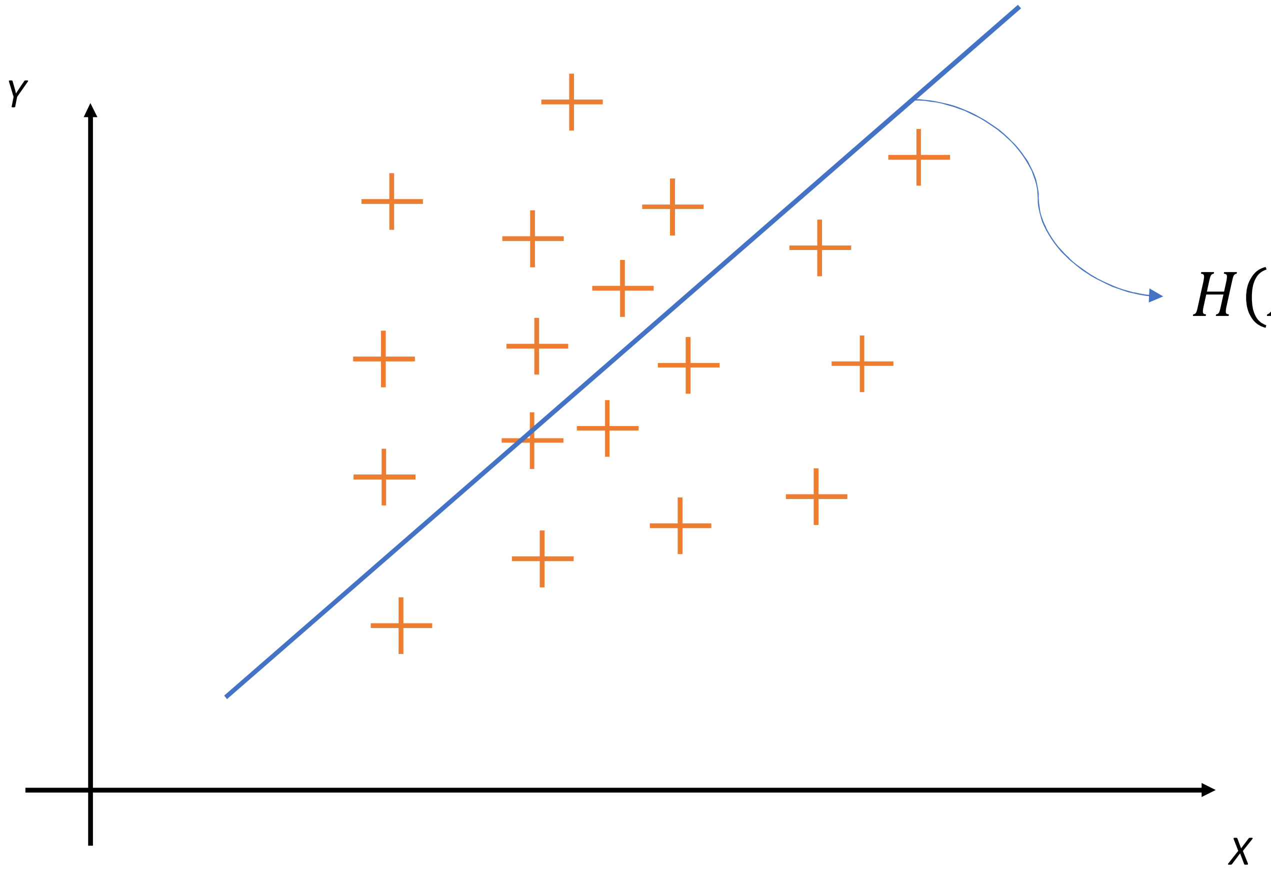
Independent variable

$$H(X) = \theta_0 + \theta_1 * X$$

Slope

$$\phi = \begin{bmatrix} \phi_0 = \theta_0 \\ \phi_1 = \theta_1 \end{bmatrix} \begin{array}{l} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{array}$$

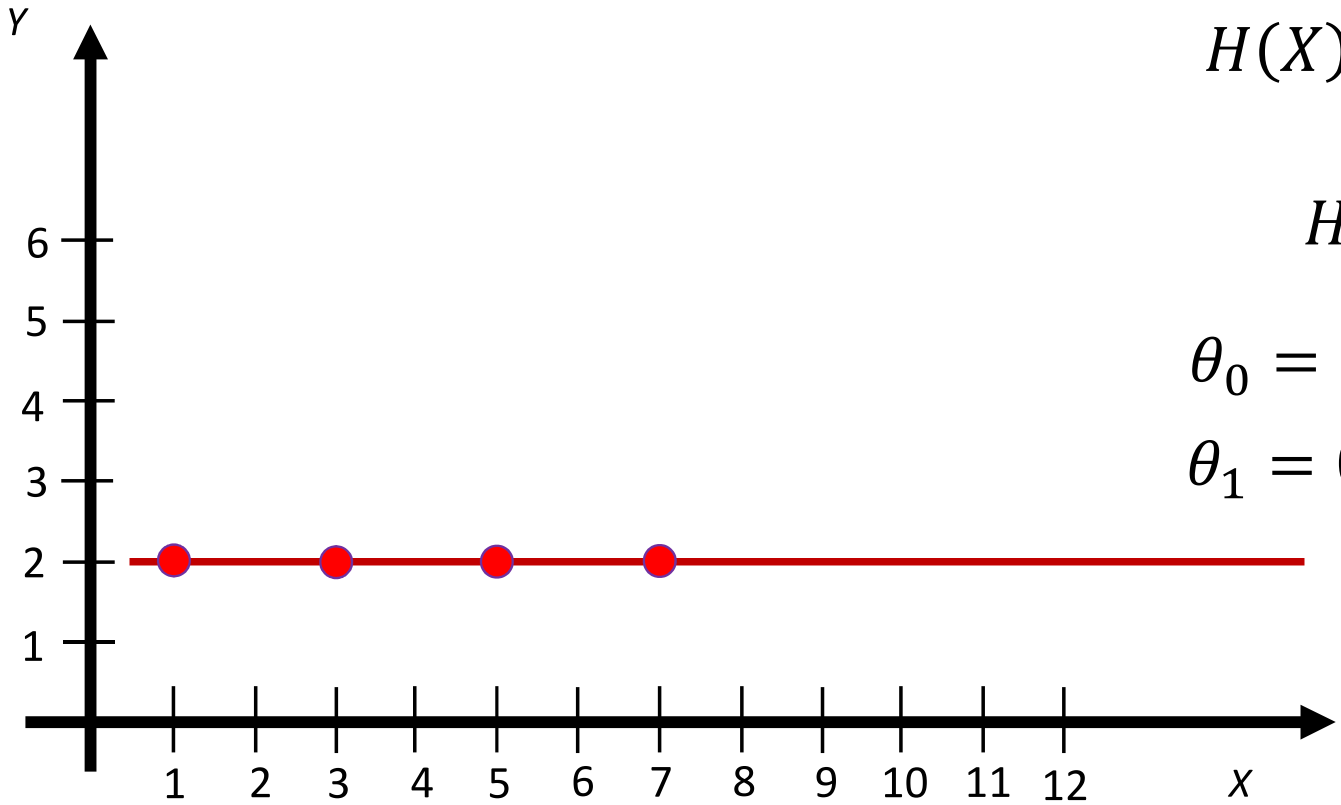
Hypothesis Function (1)



$$H(X) = \theta_0 + \theta_1 * X$$

Parameters

Hypothesis Function (1)



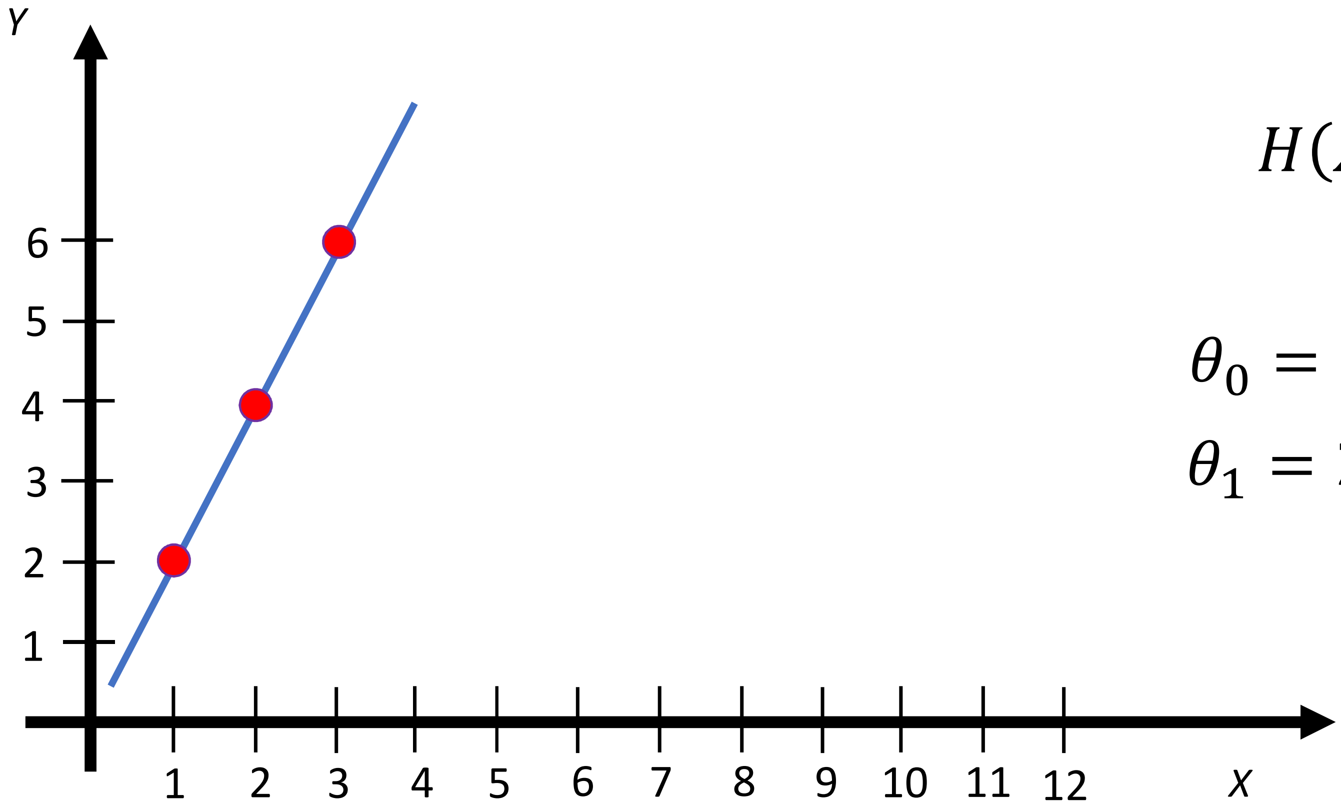
$$H(X) = \theta_0 + \theta_1 * X$$

$$H(X) = 2 + 0$$

$$\theta_0 = 2$$

$$\theta_1 = 0$$

Hypothesis Function (2)

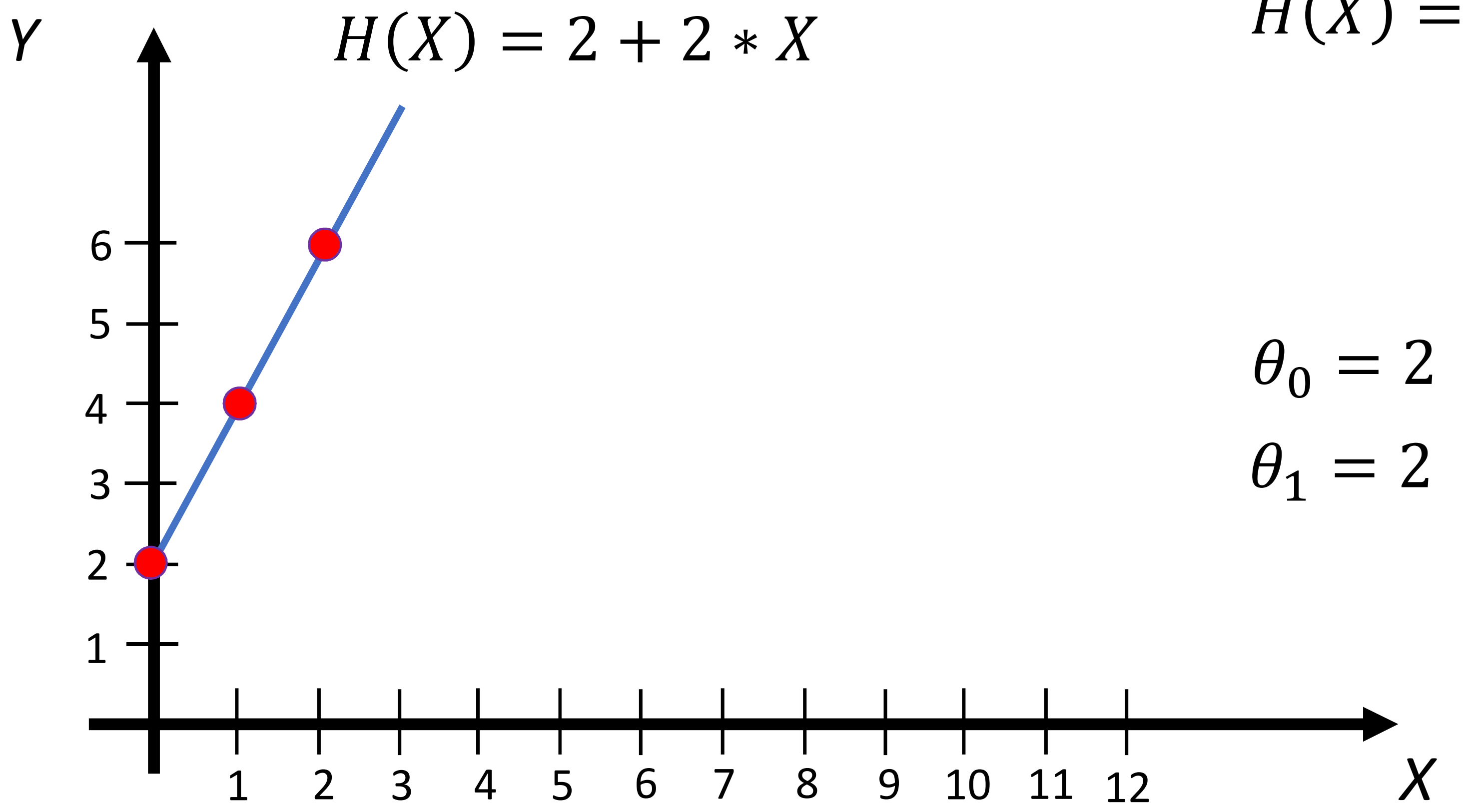


$$H(X) = 0 + 2 * X$$

$$\theta_0 = 0$$

$$\theta_1 = 2$$

Hypothesis Function (3)

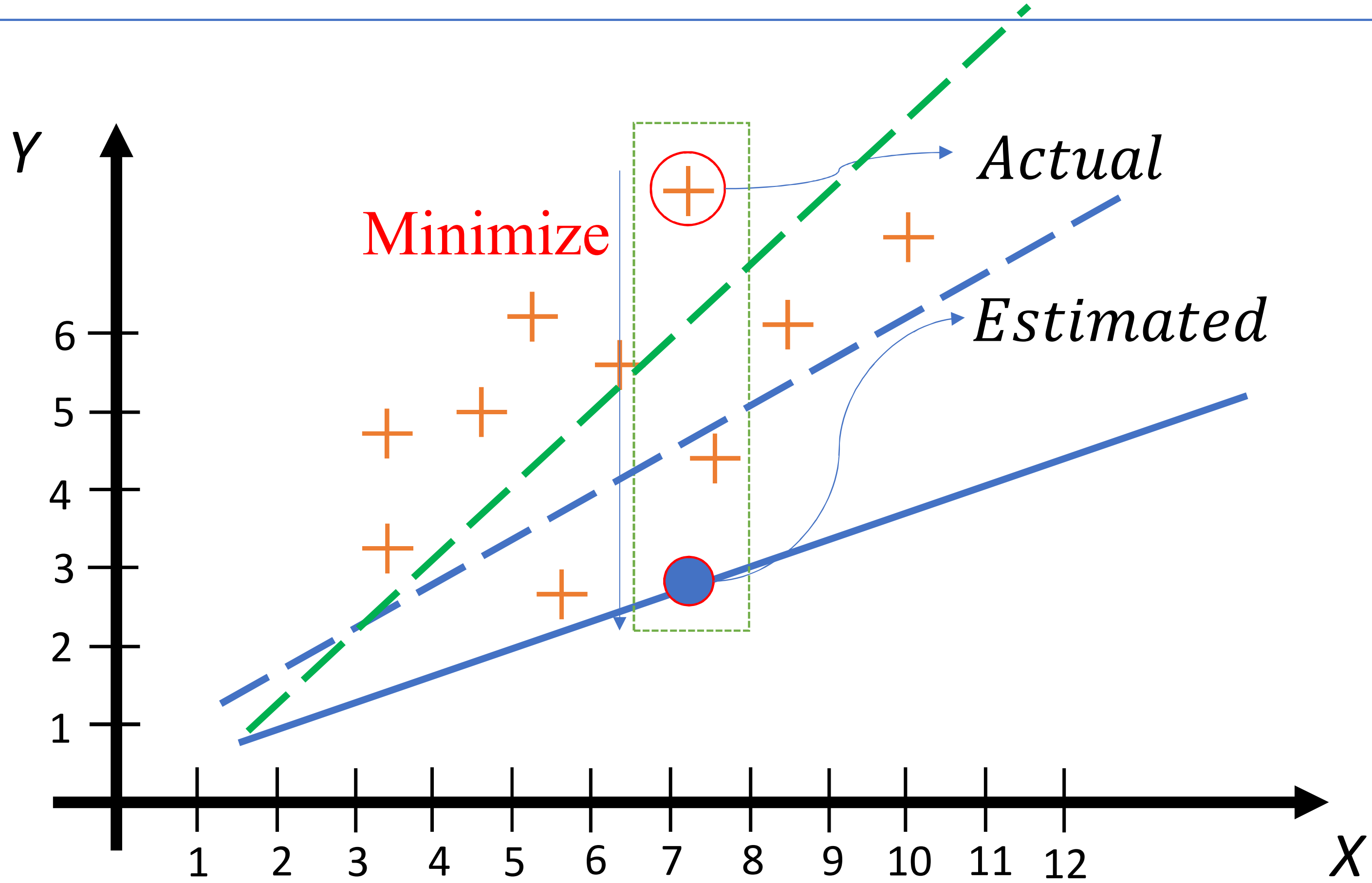


$$H(X) = \theta_0 + \theta_1 * X$$

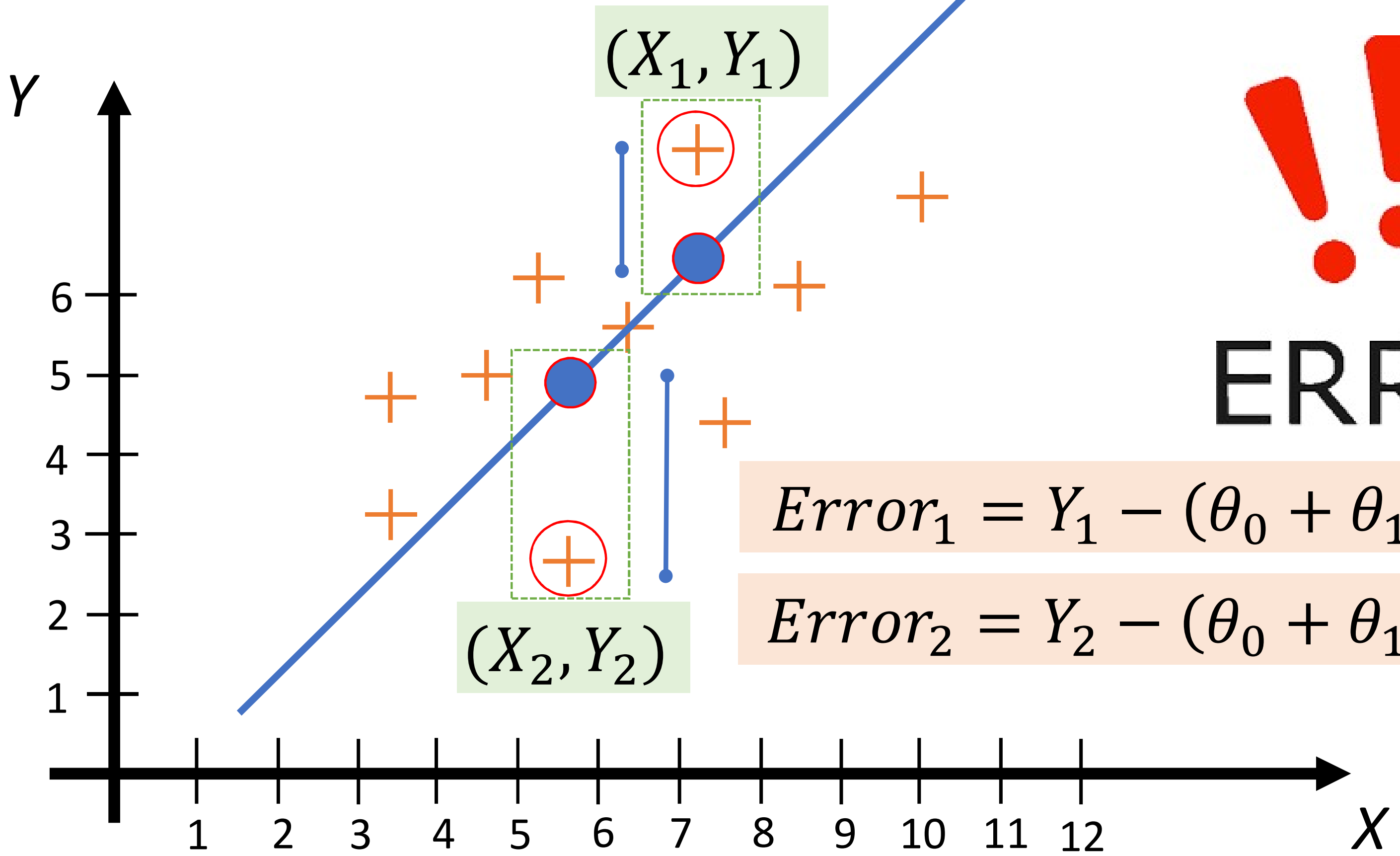
$$\theta_0 = 2$$

$$\theta_1 = 2$$

Errors



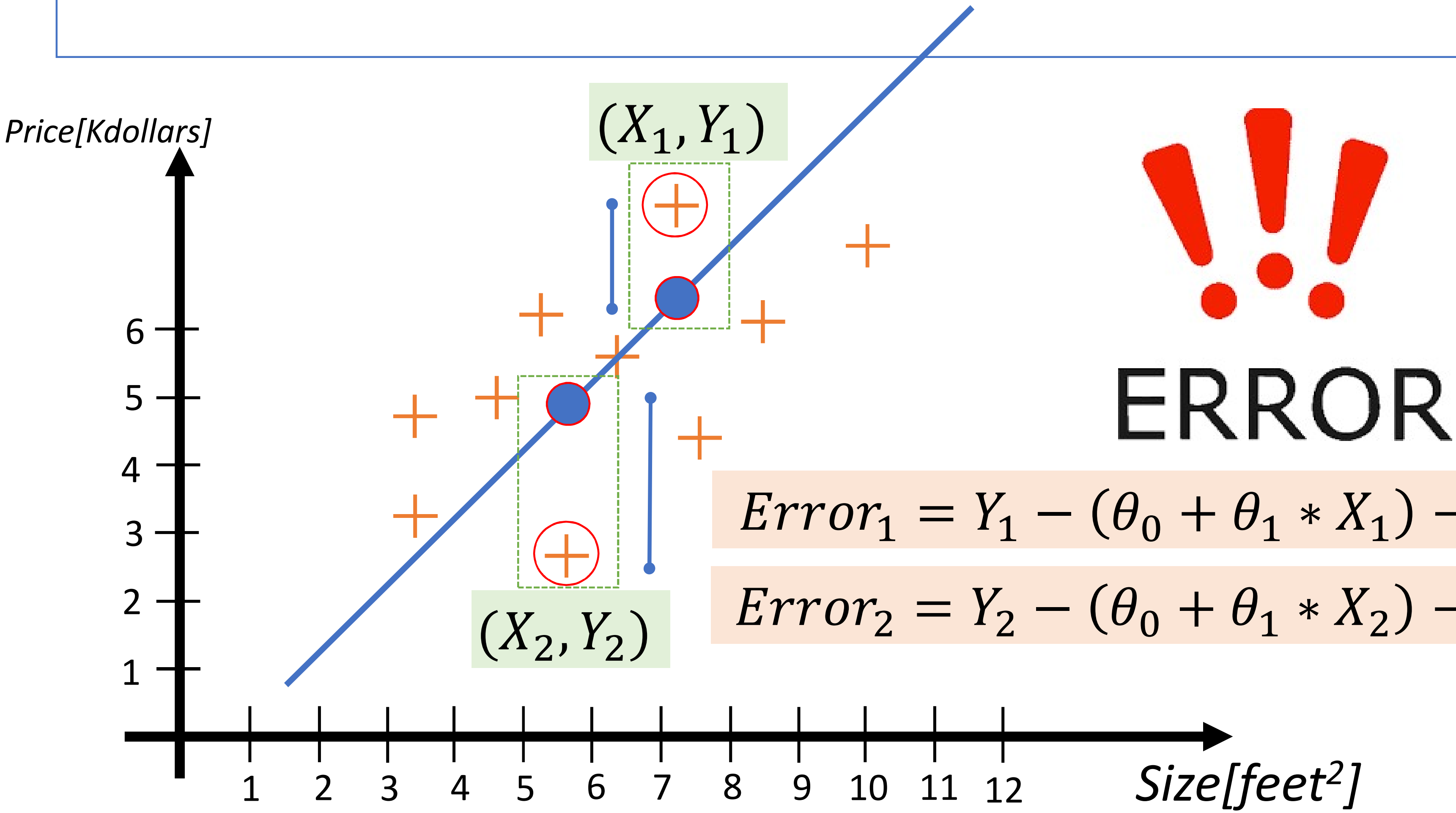
All Errors…



$$Error_1 = Y_1 - (\theta_0 + \theta_1 * X_1) \rightarrow Y_1 - H(X_1)$$

$$Error_2 = Y_2 - (\theta_0 + \theta_1 * X_2) \rightarrow Y_2 - H(X_2)$$

All Errors…



$$Error_1 = Y_1 - (\theta_0 + \theta_1 * X_1) \rightarrow Y_1 - H(X_1)$$

$$Error_2 = Y_2 - (\theta_0 + \theta_1 * X_2) \rightarrow Y_2 - H(X_2)$$

MSE

$$\frac{1}{2m}\sum_{i=1}^{m}\left[H(X^{i}) - Y^{i}\right]^{2}$$

Absolute Error Function?

Square Error Function

Cost function

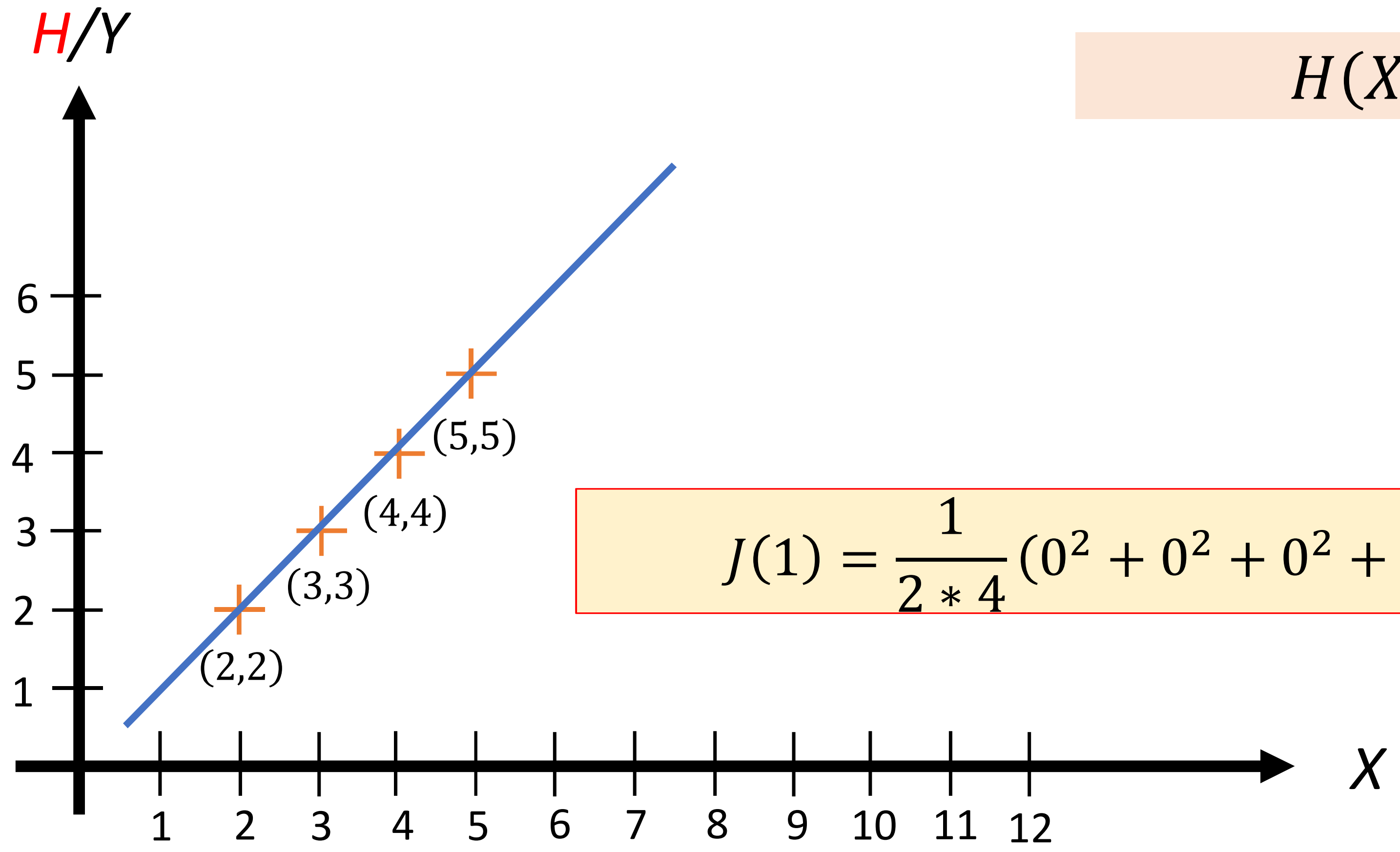
Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left[H\left(X^i\right) - Y^i\right]^2$$

Exemple($\theta_1 = 1$)

$H(X) = \theta_1 * X$ $\quad \theta_1 = 1$

$H(X) = X$



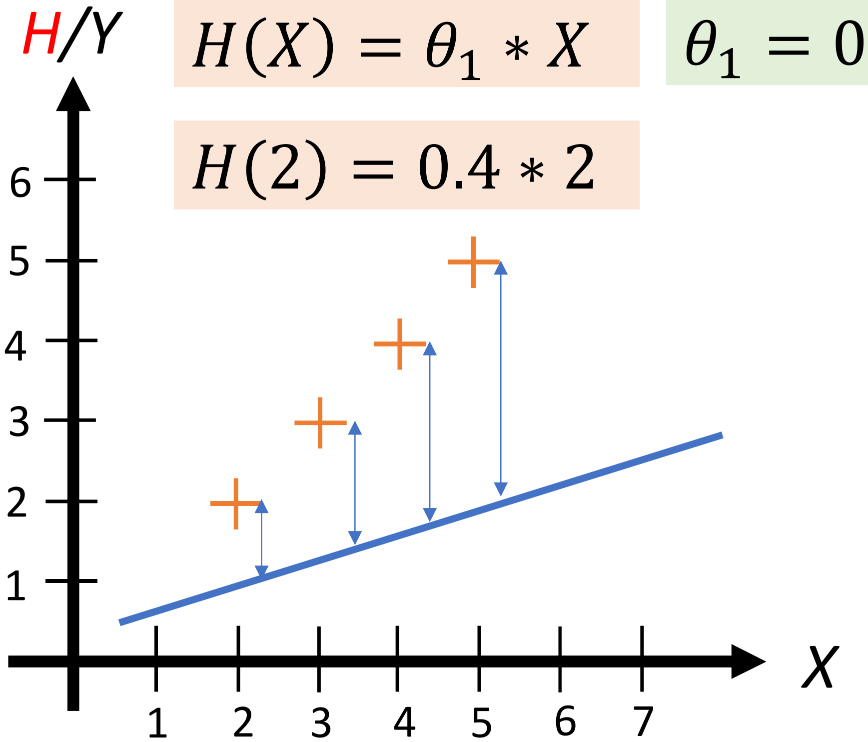
$$J(1) = \frac{1}{2 * 4}(0^2 + 0^2 + 0^2 + 0^2) = 0$$

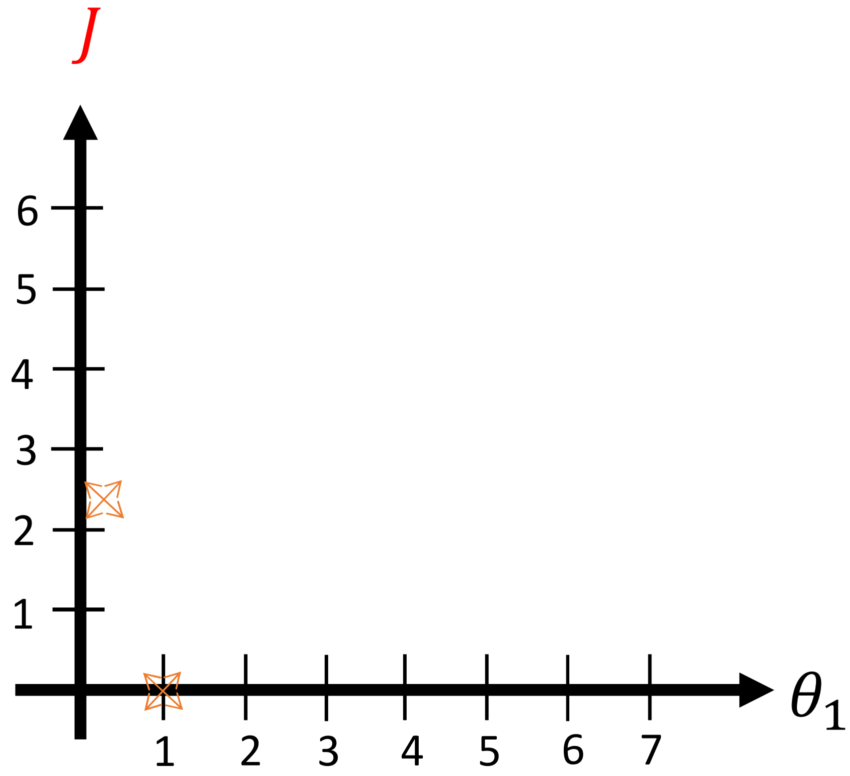
Exemple($\theta_1 = 0.4$)

$H(X) = \theta_1 * X$

$\theta_1 = 0.4$

$H(2) = 0.4 * 2$





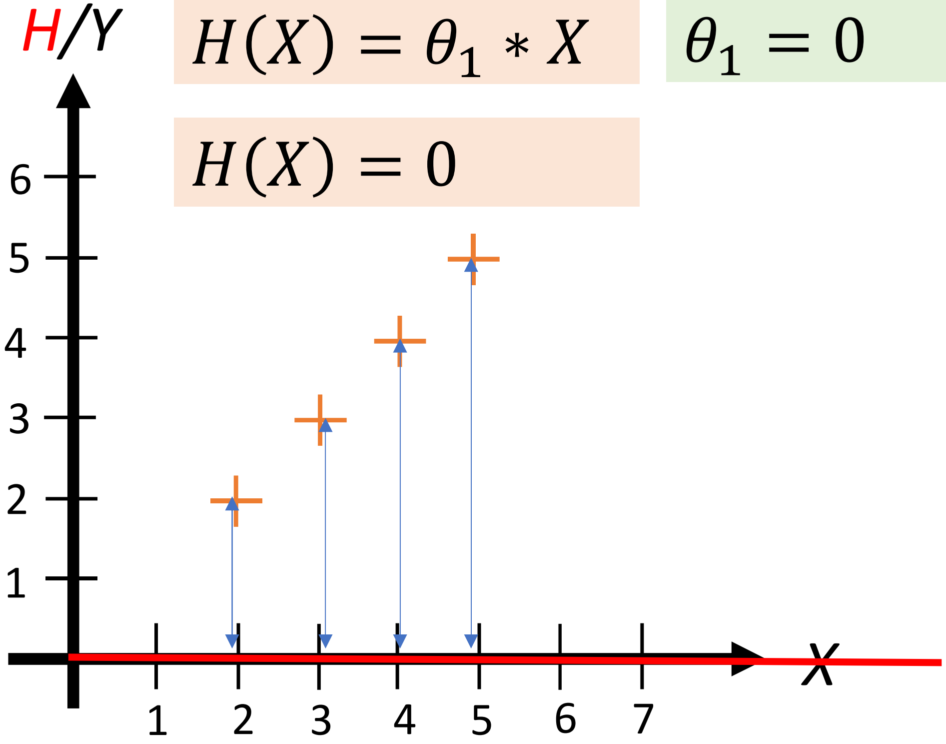
$$J(0.4) = \frac{1}{2 * 4}\left((0.4 * 2 - 2)^2 + (0.4 * 3 - 3)^2 + \cdots\right) = 2.43$$

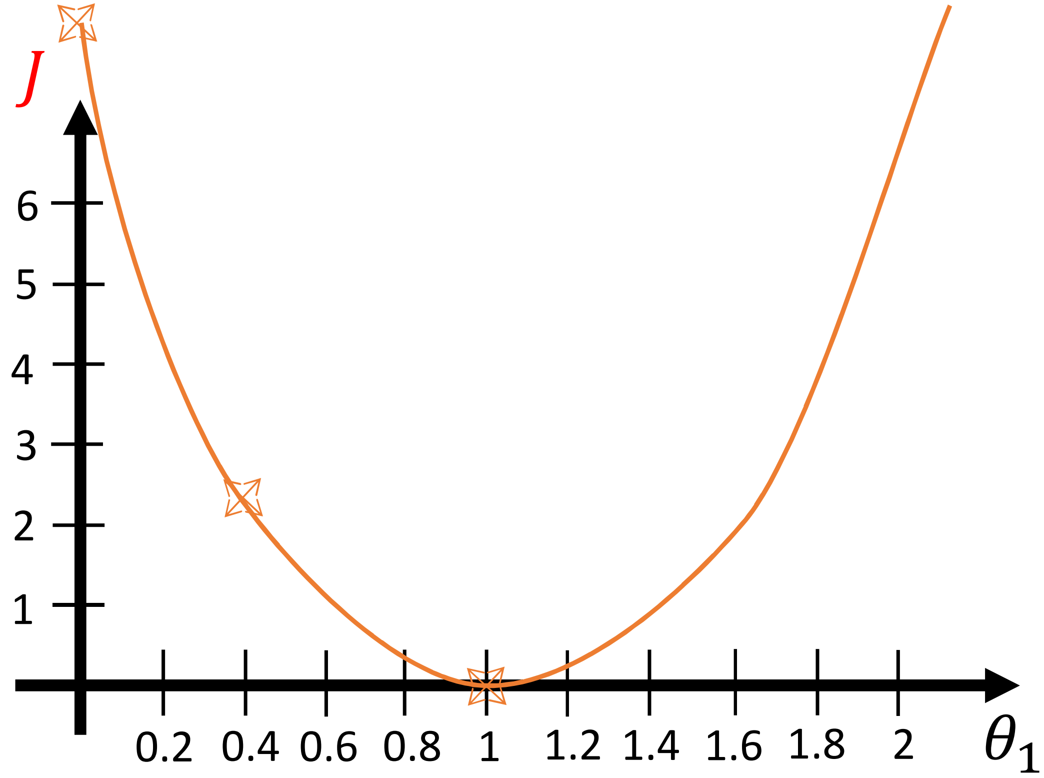
Exemple($\theta_1 = 0$)

$H(X) = \theta_1 * X$

$\theta_1 = 0$

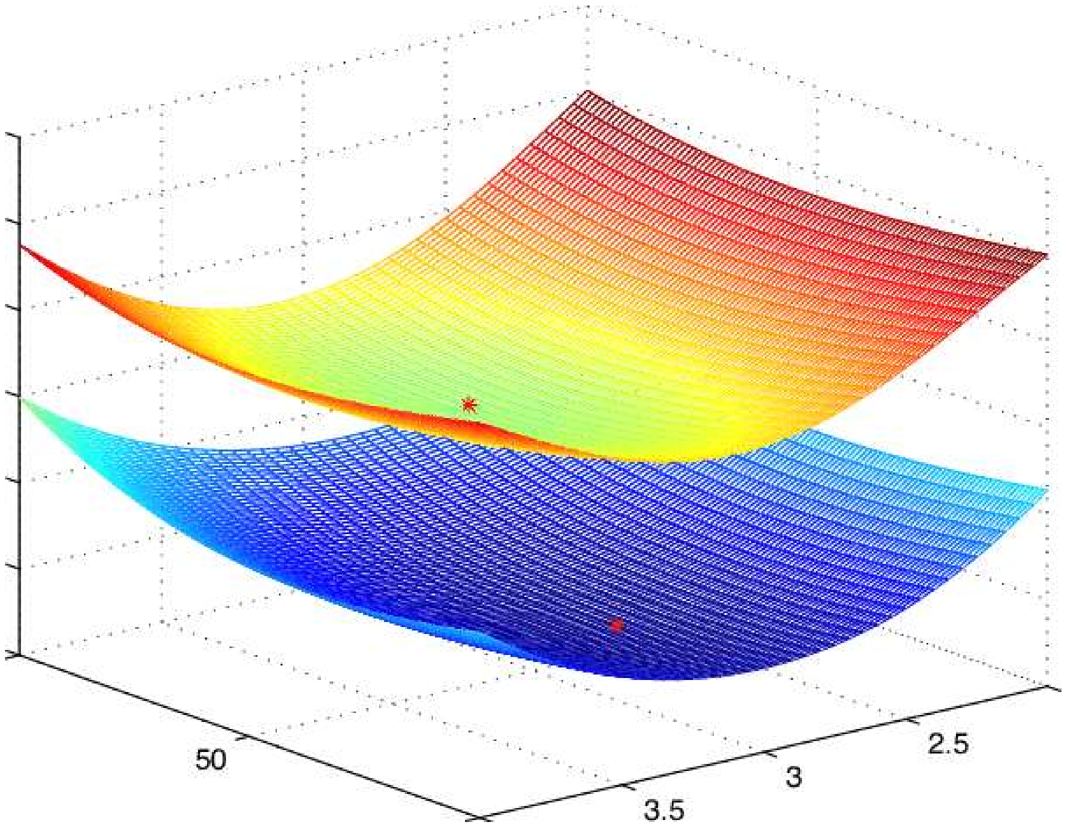
$H(X) = 0$





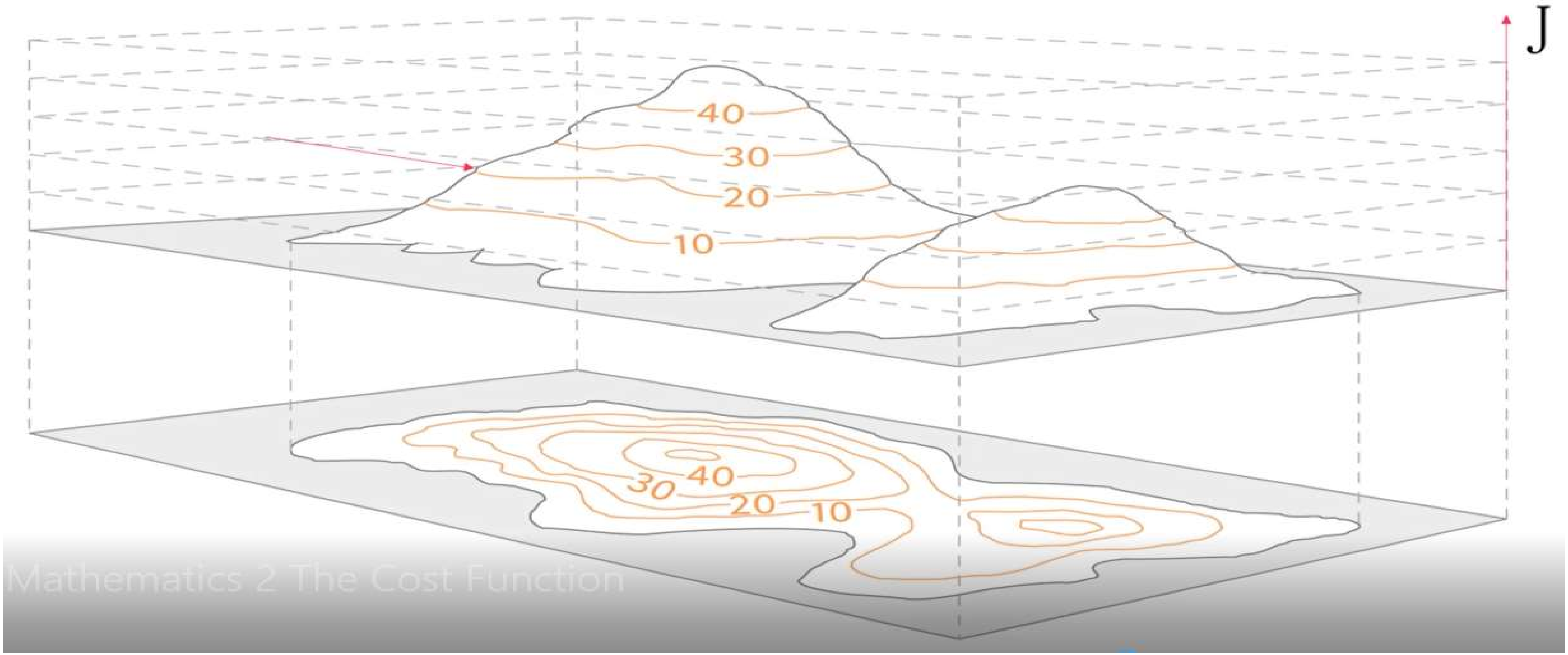
$$J(0) = \frac{1}{2 * 4}\left((2)^2 + (3)^2 + (4)^2 + (5)^2\right) = 9$$

Cost function

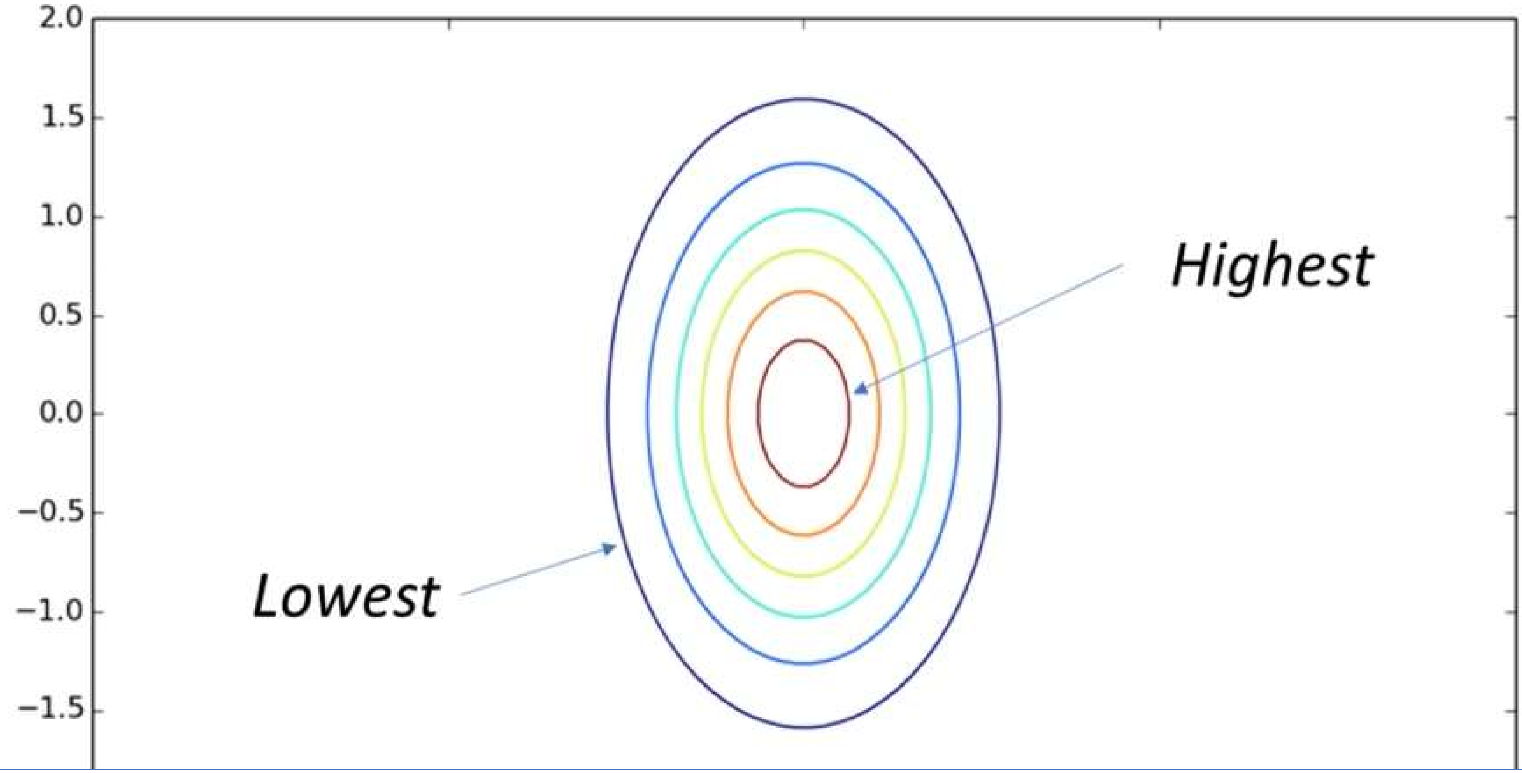


$$H(X) = \theta_0 + \theta_1 * X$$

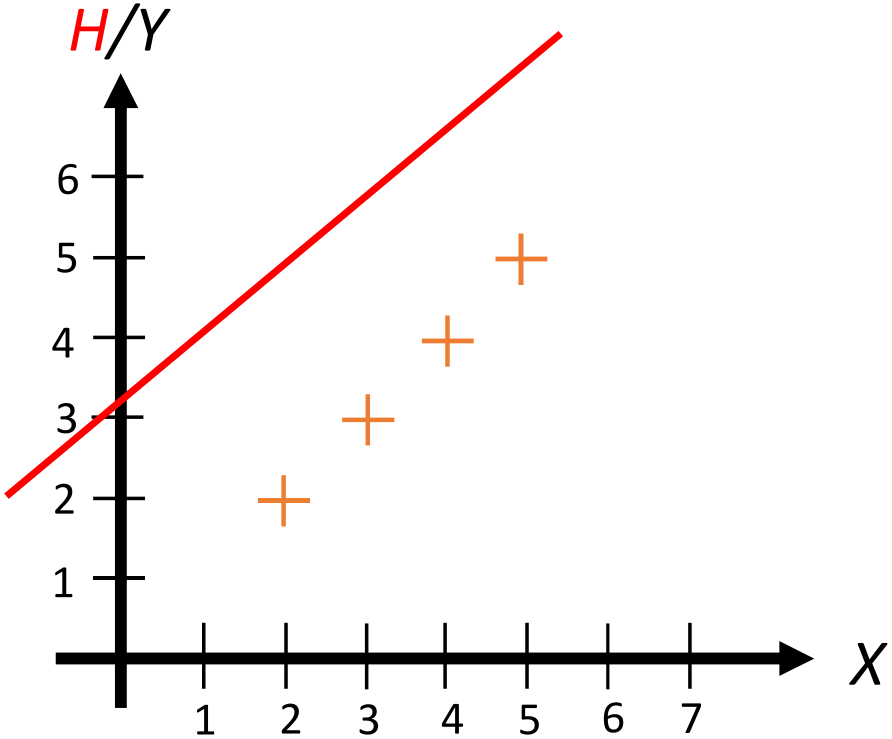
Cost function

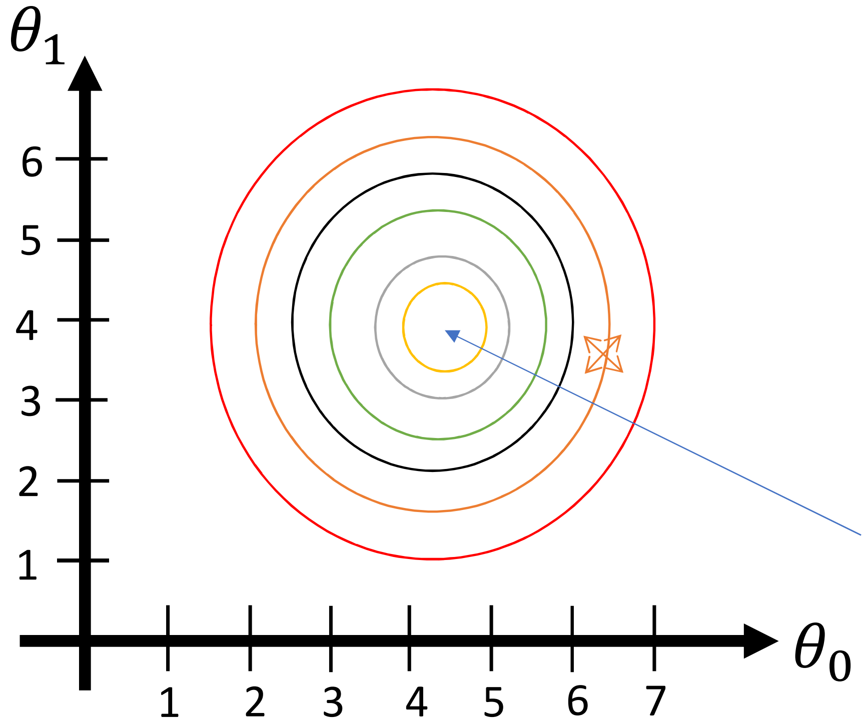


Cost function



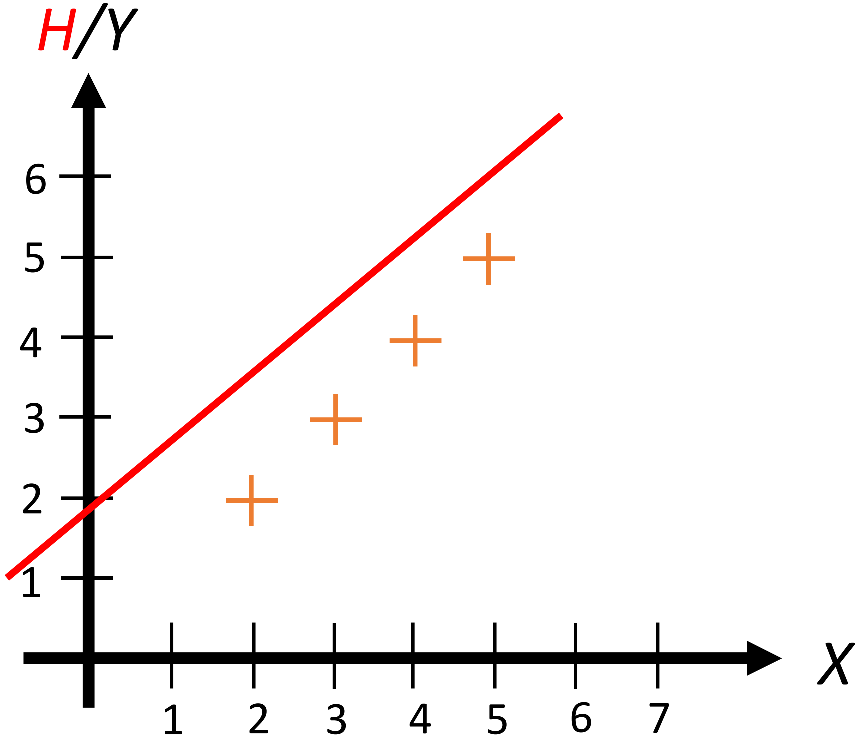
Cost function

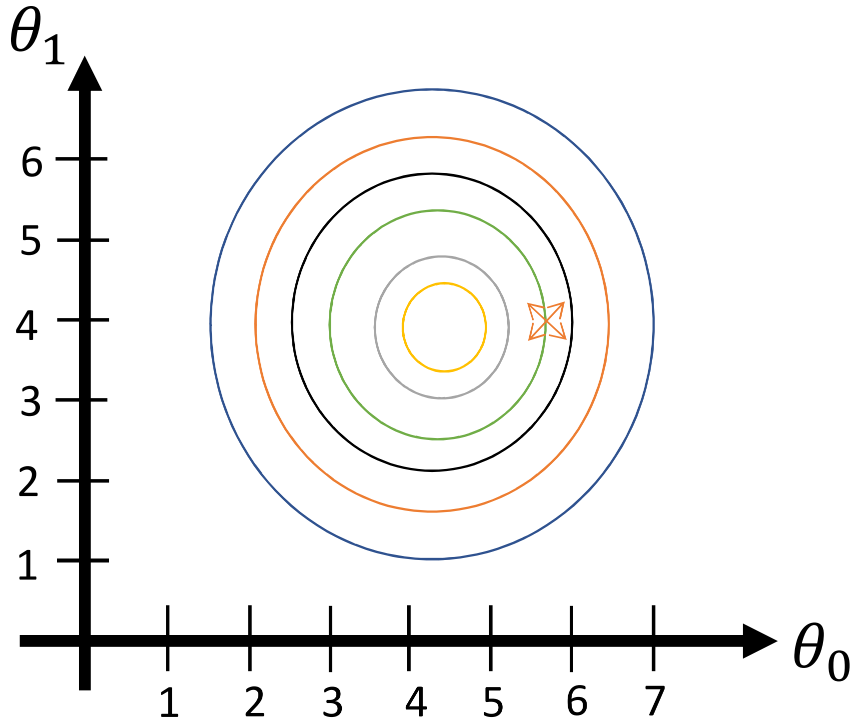




$$H(X) = \theta_0 + \theta_1 * X$$

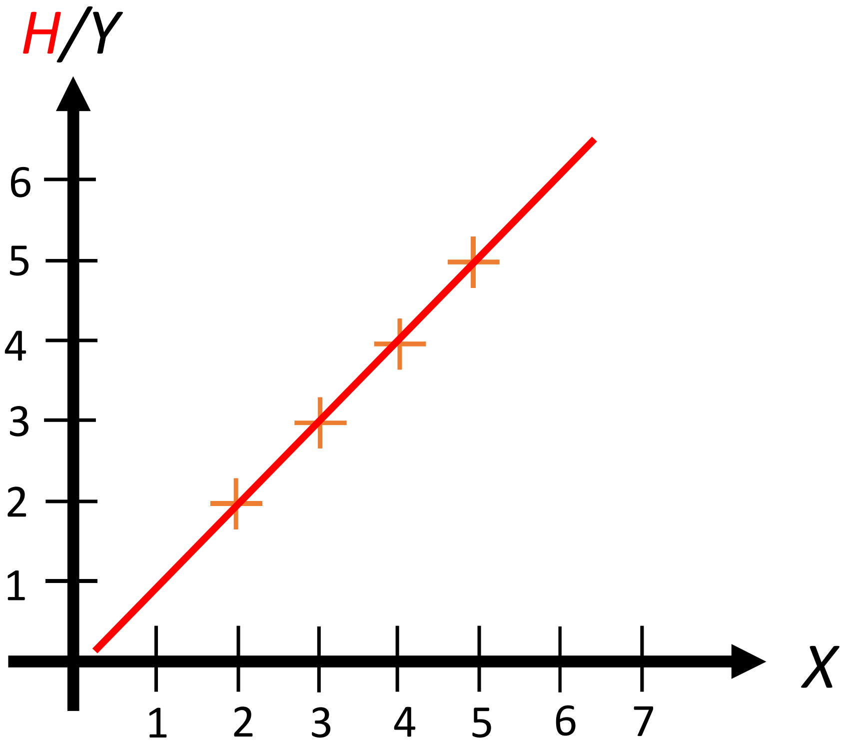
Cost function

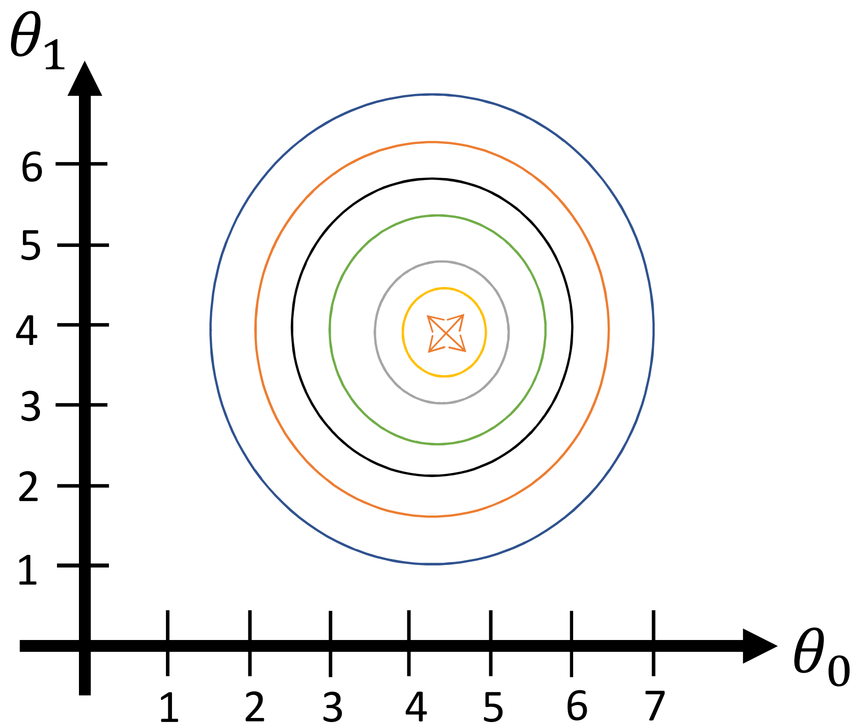




$$H(X) = \theta_0 + \theta_1 * X$$

Cost function





$$H(X) = \theta_0 + \theta_1 * X$$

Optimization
Gradient Descent

- The Gradient: Vector indicates the direction in witch the function is increasing by largest amount.

Multivariable Functions

$$F(X,Y)$$

$$\nabla\mathcal{F} = \frac{\partial\mathcal{F}}{\partial x}\boxed{i} + \frac{d\mathcal{F}}{dy}\boxed{j}$$

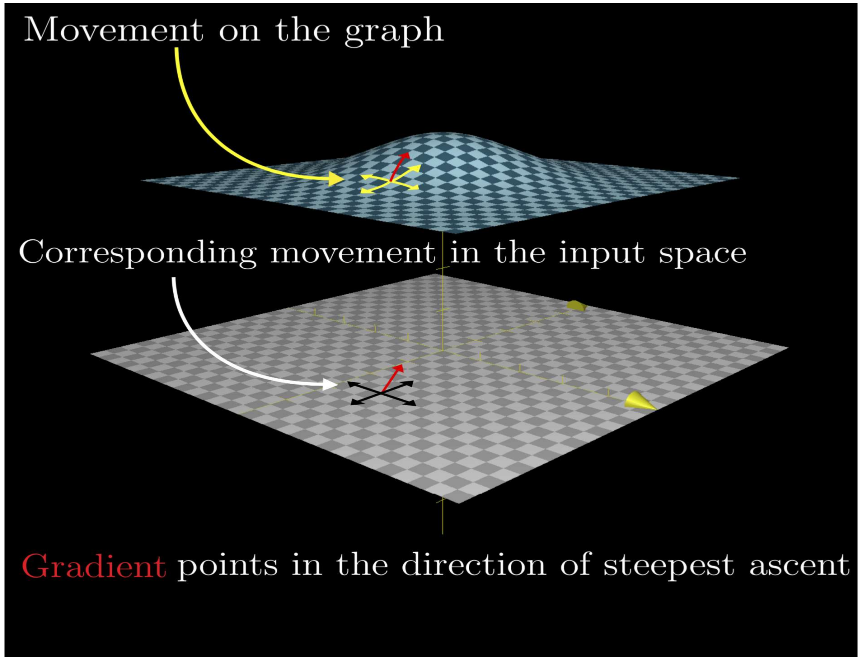
Gradient Descent

- The Gradient: Vector indicates the direction in witch the function is increasing by largest amount.

Multivariable Functions

$$F(X, Y)$$

$$\nabla \mathcal{F} = \frac{\partial \mathcal{F}}{\partial x} \boxed{i} + \frac{d\mathcal{F}}{dy} \boxed{j}$$



How GD Works

$=$

$$\theta \boxed{:=} \theta - \nabla J(\theta_0 + \theta_1)$$

$Assignment$ $\boxed{=}$

$Equality$ $\boxed{==}$

How GD Works

$$\theta \coloneqq \theta - \nabla J(\theta_0 + \theta_1)$$

$$\theta_0 = 0$$

$$\theta_1 \coloneqq \theta_1 - \nabla J(\theta_1)$$

How GD Works

$$\theta_1 := \theta_1 - \nabla J(\theta_1)$$

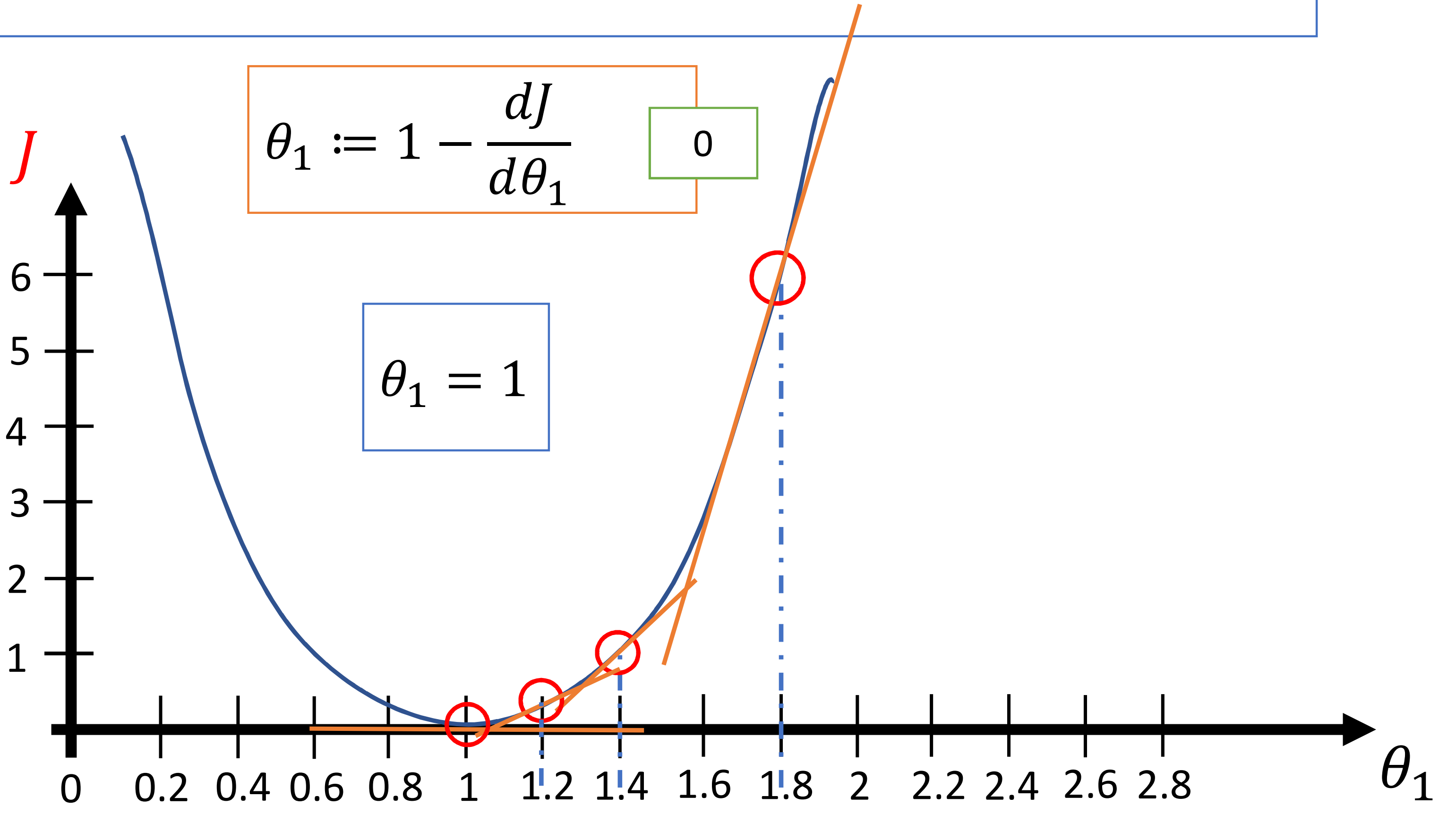
$$\theta_1 := 1.8$$

$$\theta_1 := 1.8 - \nabla J(\theta_1)$$

$$\theta_1 := 1.8 - \frac{dJ}{d\theta_1}$$ 0.4

$$\theta_1 := 1.4 - \frac{dJ}{d\theta_1}$$ 0.2

$$\theta_1 := 1.2 - \frac{dJ}{d\theta_1}$$ 0.1



Query 1 : What about the Initialization?

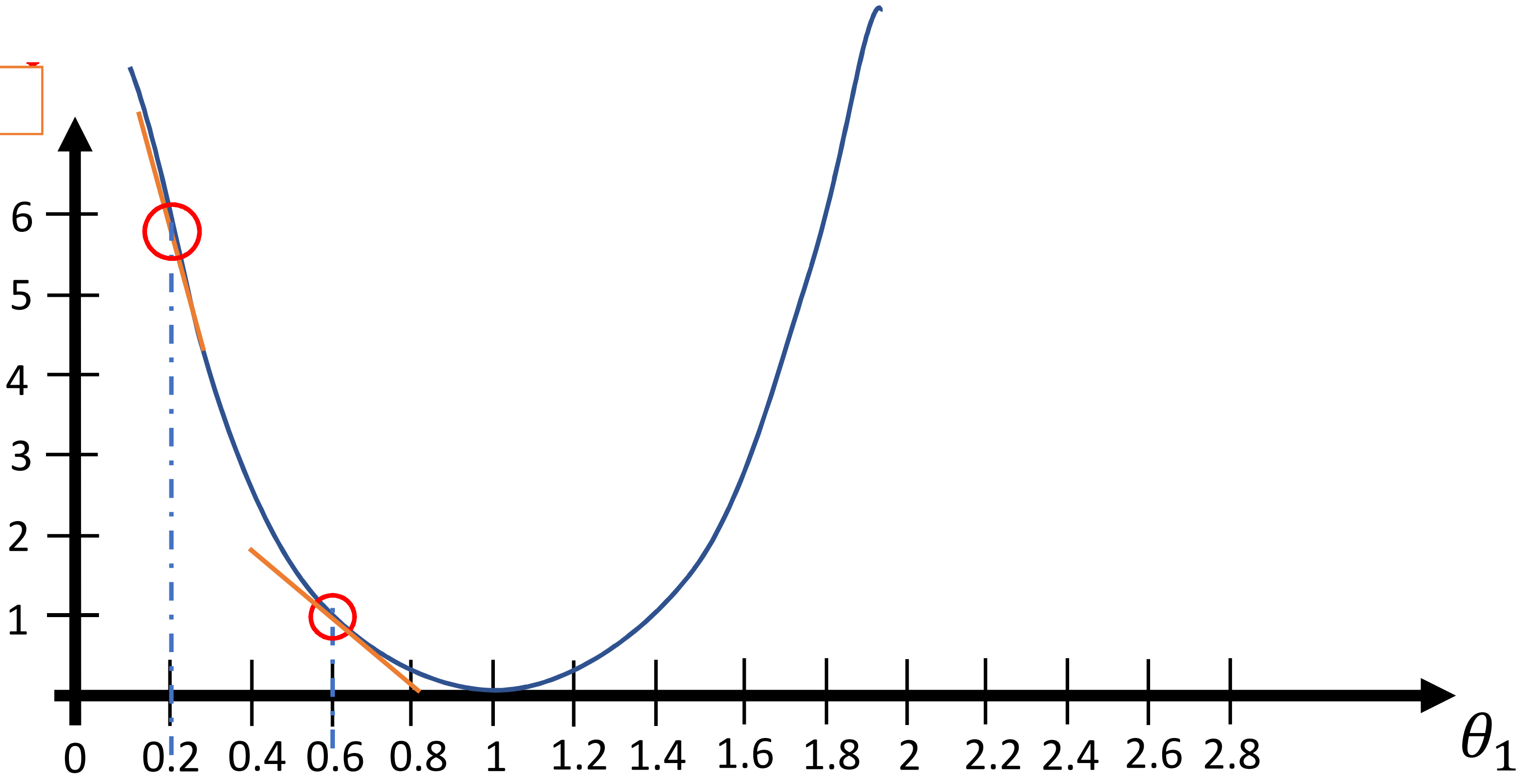
$\theta_1 := \theta_1 - \nabla J(\theta_1)$

$\theta_1 := 1.8$ $\theta_1 := 0.2$

$\theta_1 := 0.2 - \nabla J(\theta_1)$

$\theta_1 := 0.2 - \dfrac{dJ}{d\theta_1}$ -0.4

$\theta_1 := 0.2 + 0.4$ 0.6



Query 2 How to Adjust the Speed of Algorithm?

$\theta_1 := \theta_1 - \nabla J(\theta_1)$

$\theta_1 := 1.8$

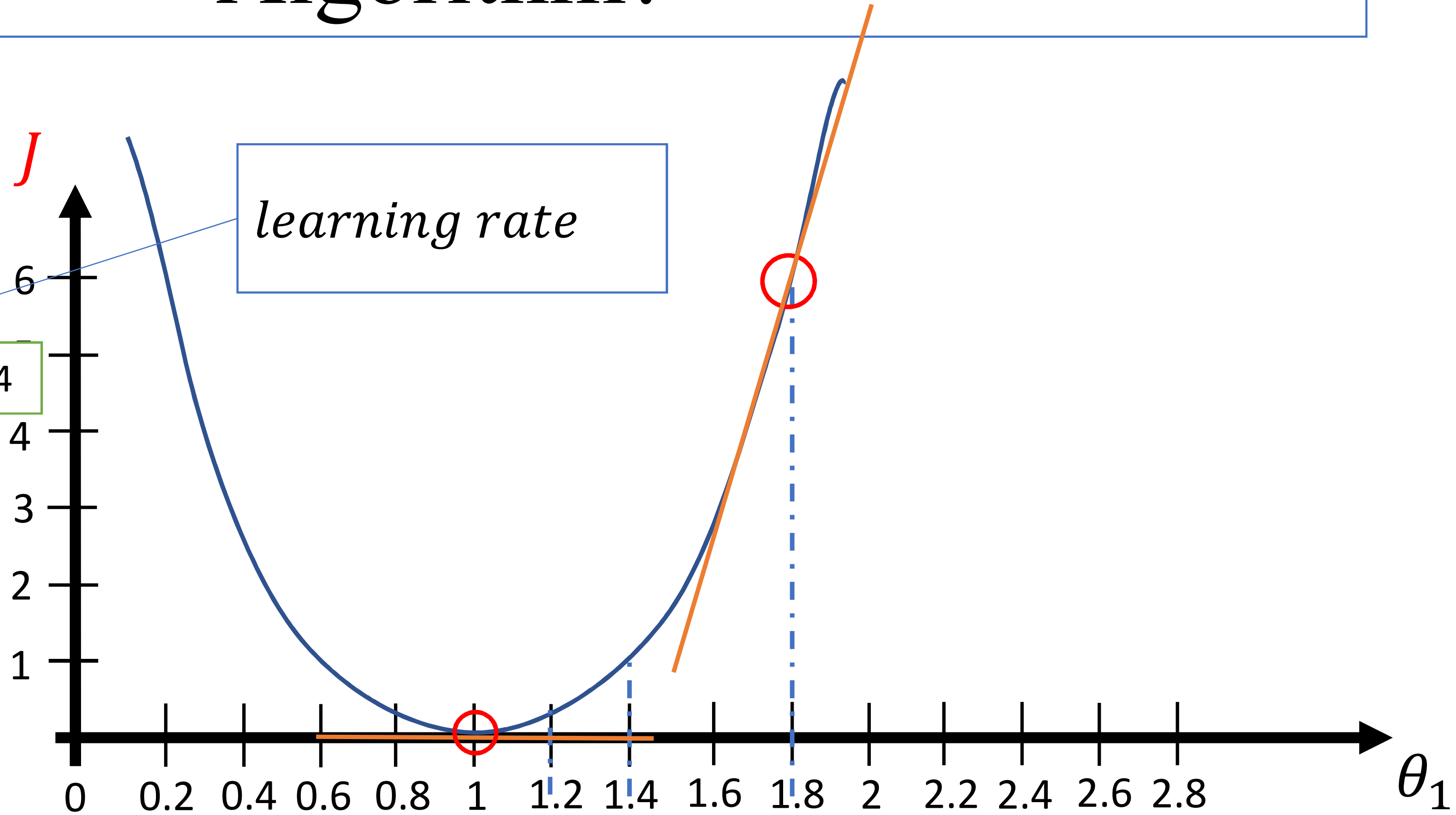
$\theta_1 := 1.8 - \nabla J(\theta_1)$

learning rate

$\theta_1 := 1.8 - 2 * \dfrac{dJ}{d\theta_1}$ 0.4

$\theta_1 := 1.8 - 0.8$ 1

$\theta_1 = 1$



Query 2 How to Adjust the Speed of Algorithm?

$$\theta_1 \coloneqq \theta_1 - \propto * \nabla J(\theta_1)$$

Query 2 How to Adjust the Speed of Algorithm?

$\theta_1 := \theta_1 - \propto \nabla J(\theta_1)$

$\theta_1 := 1.8$

$\theta_1 := 1.8 - \nabla J(\theta_1)$

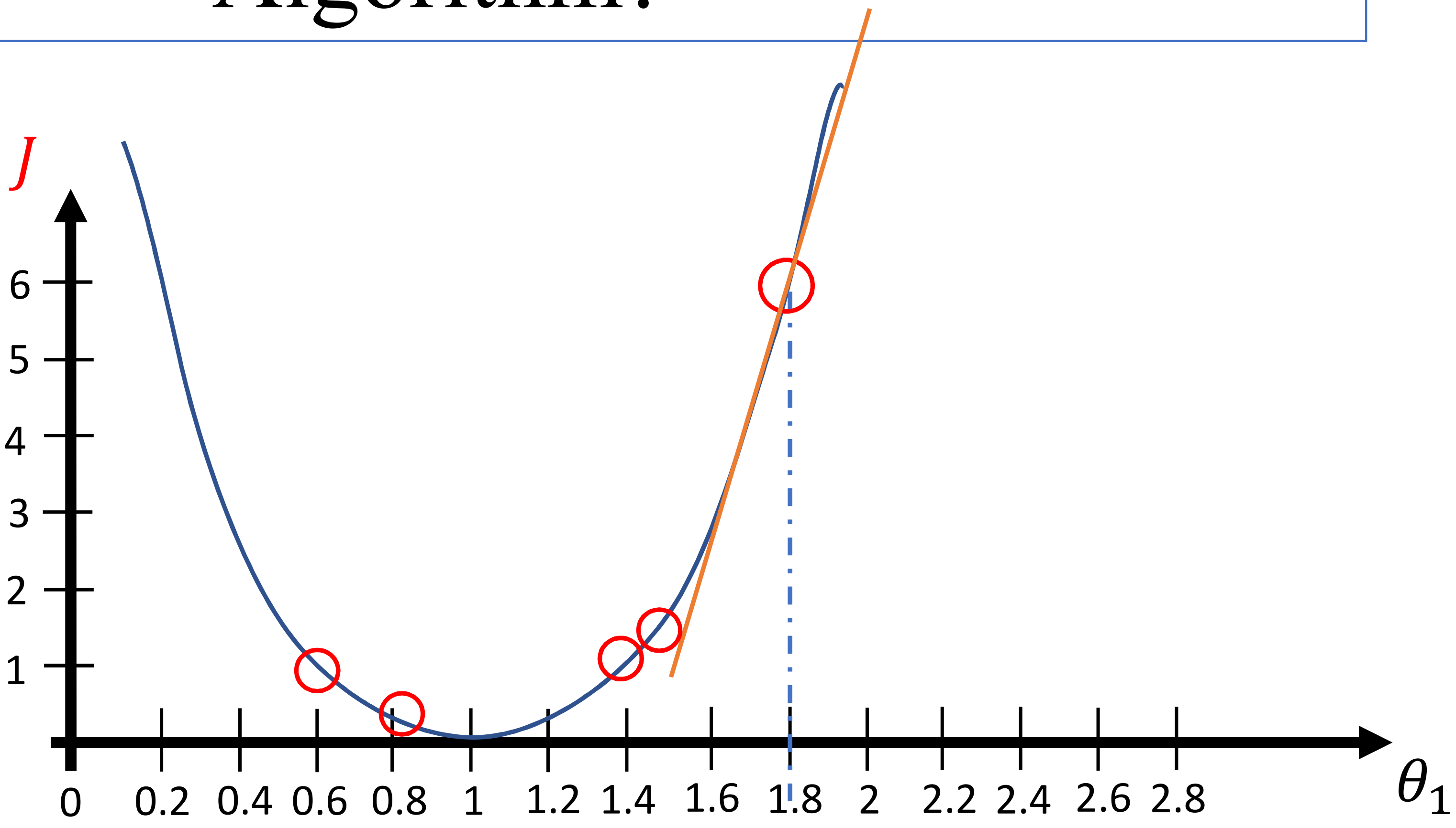
$$\theta_1 := 1.8 - 3 * \frac{dJ}{d\theta_1}$$

$\theta_1 = 0.6$

$\theta_1 = 1.4$

$\theta_1 = 0.88$

$\theta_1 = 1.45$



Query 2 How to Adjust the Speed of Algorithm?

$\theta_1 := \theta_1 - \propto \nabla J(\theta_1)$

$\theta_1 := 1.8$

$\theta_1 := 1.8 - \propto \nabla J(\theta_1)$

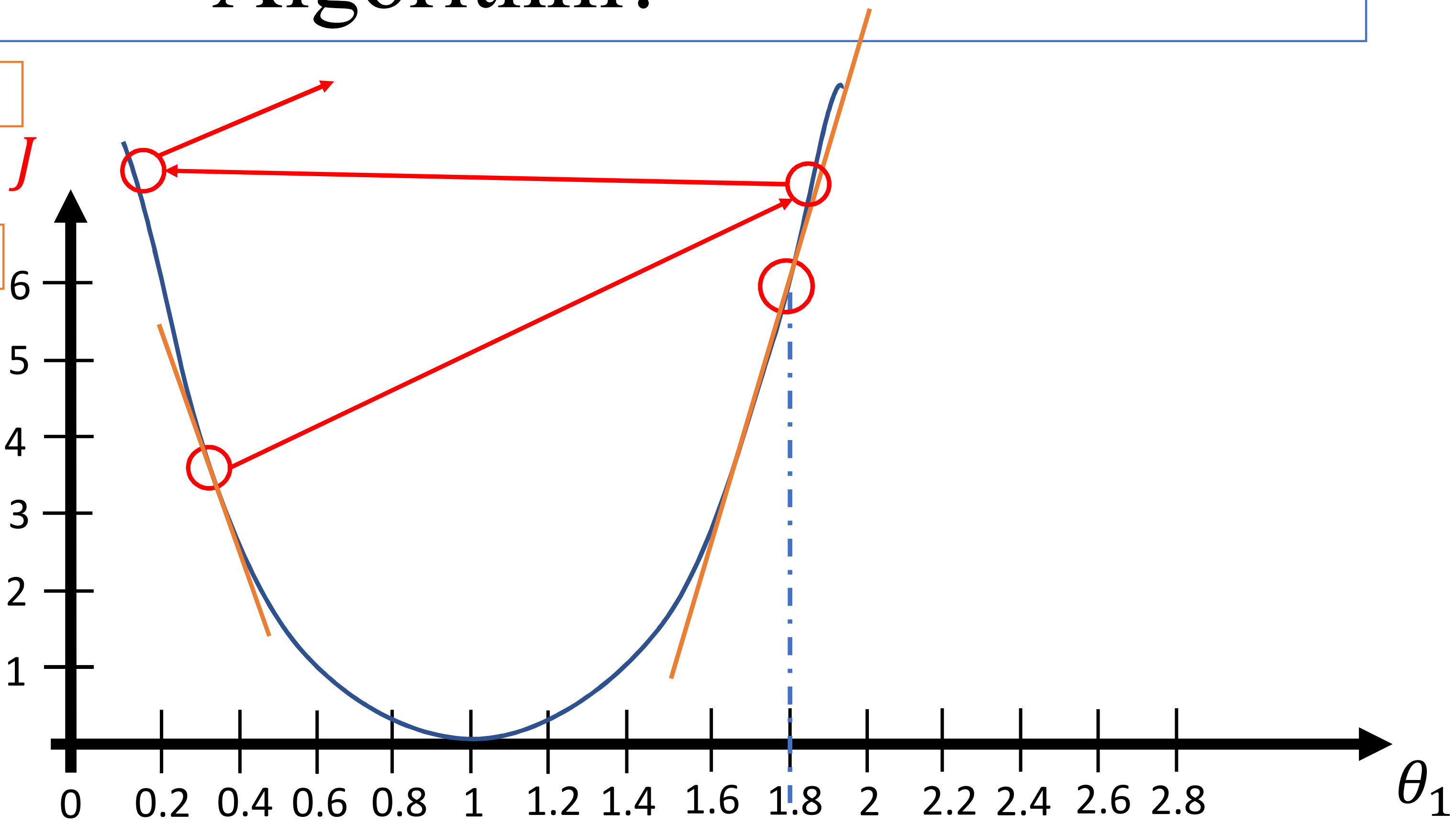
$$\theta_1 := 1.8 - 5 * \frac{dJ}{d\theta_1}$$

$\theta_1 = ??$

$\theta_1 = ??$

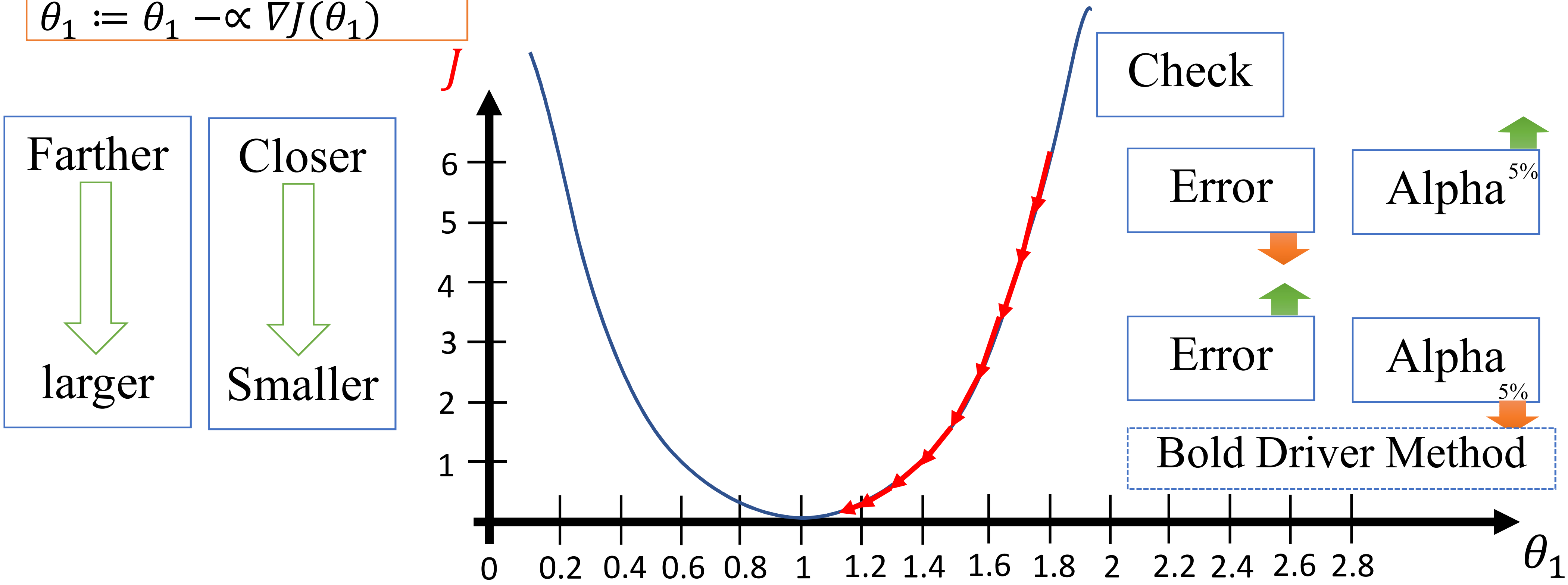
$\theta_1 = ??$

$\theta_1 = ??$

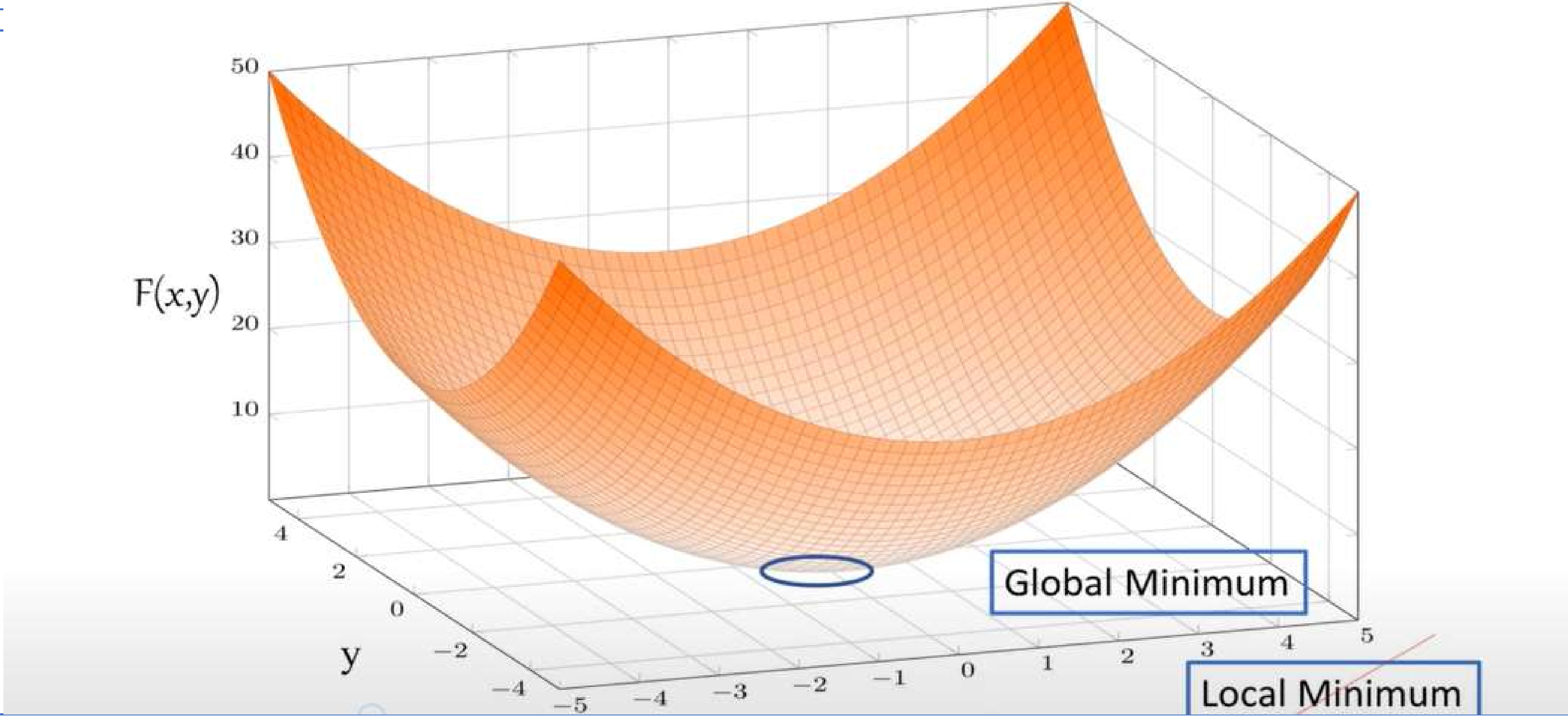


Query 2 How to Adjust the Speed of Algorithm

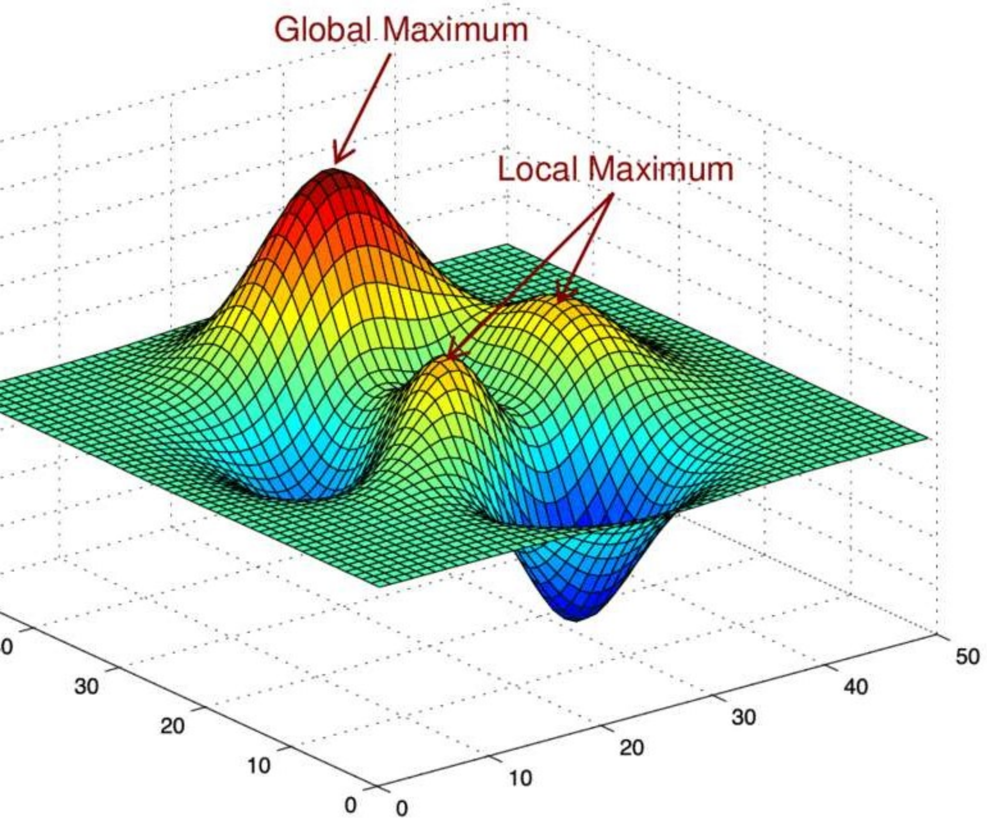
$$\theta_1 := \theta_1 - \propto \nabla J(\theta_1)$$

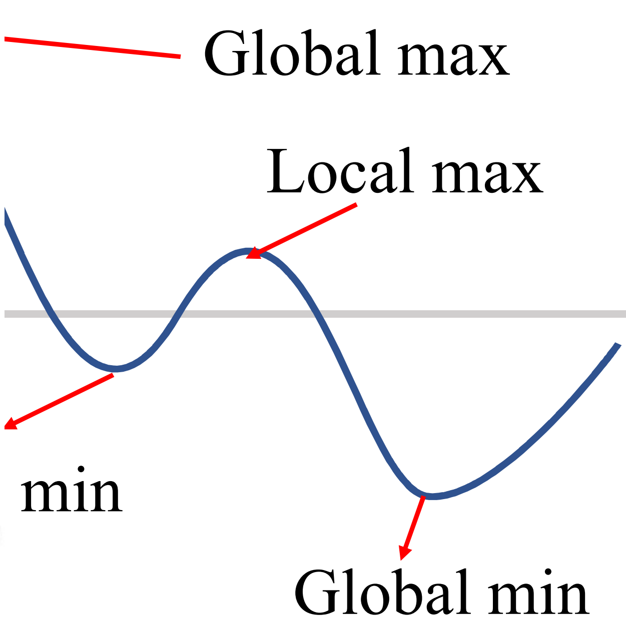


Query 3 What if it Was a Non-Convex Function?

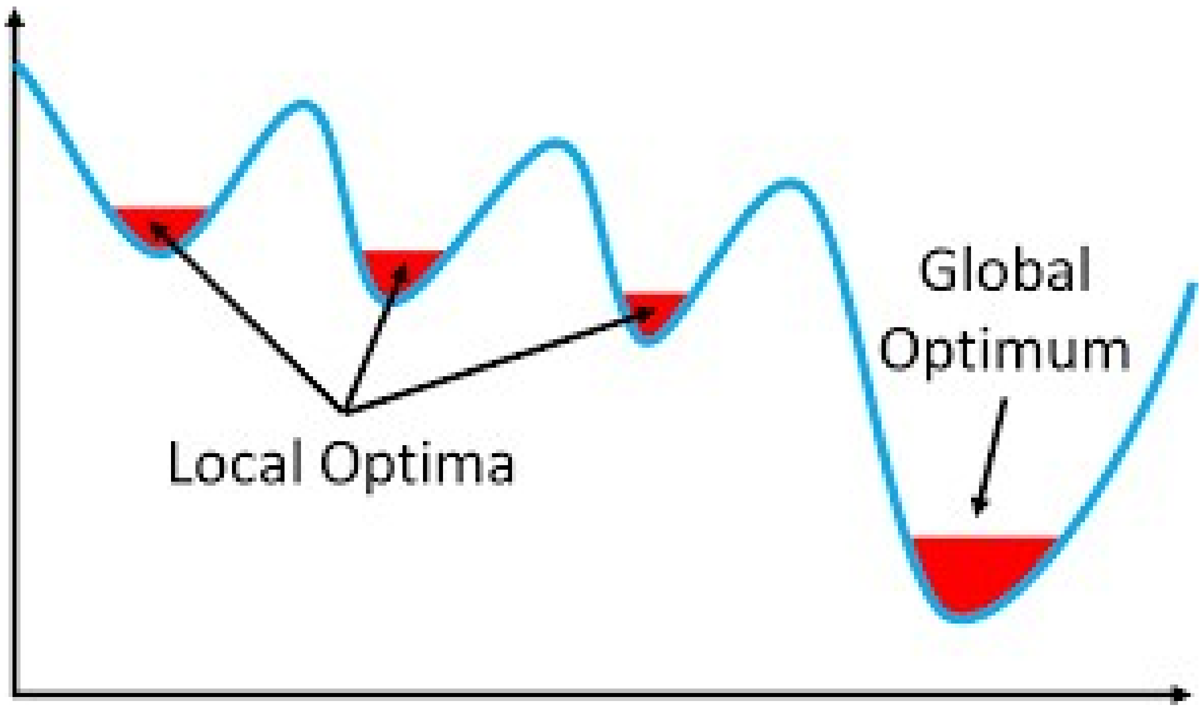


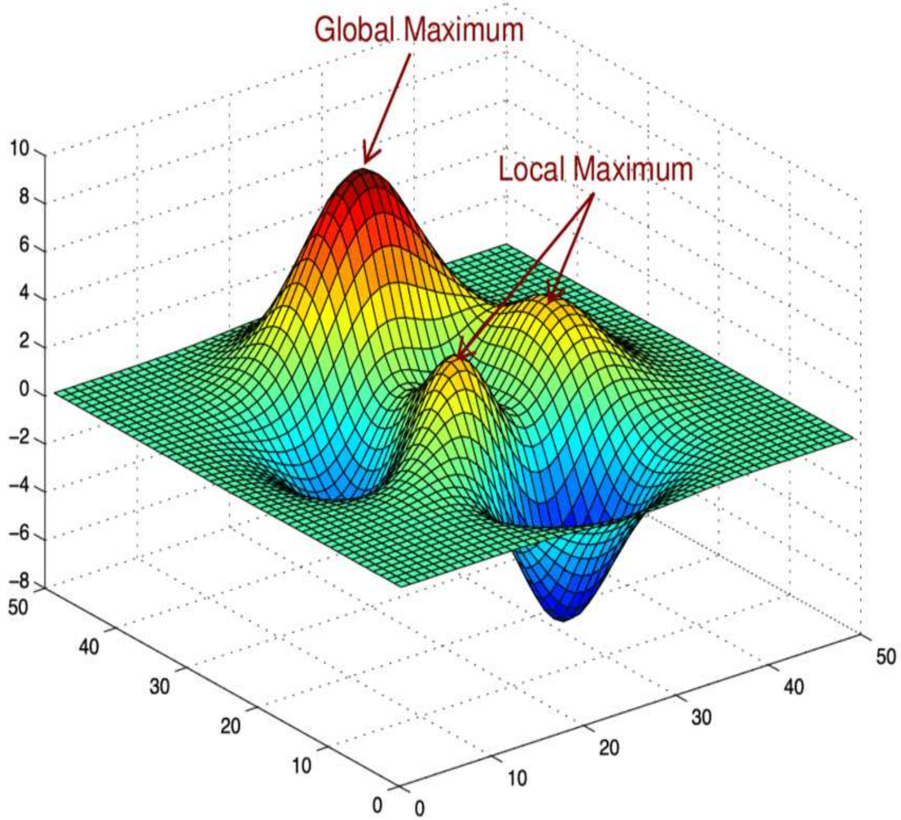
Global vs Local Optimum





Global vs Local Optimum





Polymerization Between Gradient and Hypothesis

Linear Hypothesis

$$H(X) = \theta_0 + \theta_1 * X$$

$$\frac{1}{2m}\sum_{i=1}^{m}\left[H\left(X^i\right) - Y^i\right]^2$$

Gradient Descent

$$\theta_1 \coloneqq \theta_1 - \propto \nabla J(\theta_1)$$

$$\begin{gathered} Repeate\ until\ convergence: \{ \\ \theta_i \coloneqq \theta_i - \propto \nabla J(\theta_0, \theta_1) \\ [i = 0, i = 1]\} \end{gathered}$$

$$\begin{gathered} Repeate\ until\ convergence: \{ \\ \theta_i \coloneqq \theta_i - \propto \frac{d}{d\theta_i} J(\theta_0, \theta_1) \\ [i = 0, i = 1]\} \end{gathered}$$

Polymerization Between Gradient and Hypothesis

$$\{ \quad \theta_i := \theta_i - \propto \frac{d}{d\theta_i} J(\theta_0, \theta_1) \quad [i = 0, i = 1] \}$$

$$\frac{d}{d\theta_0} J(\theta_0, \theta_1) = \frac{d}{d\theta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}\left[H(X^i) - Y^i\right]^2\right)$$

$$= \frac{d}{d\theta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}\left[\theta_0 + \theta_1 * X - Y^i\right]^2\right)$$

$$= \left(\frac{1}{m}\sum_{i=1}^{m}\left[\theta_0 + \theta_1 * X - Y^i\right].1\right) = \frac{1}{m}\sum_{i=1}^{m}\left[H(X^i) - Y^i\right]$$

$i = 0$

Polymerization Between Gradient and Hypothesis

$$\{ \quad \theta_i := \theta_i - \propto \frac{d}{d\theta_i} J(\theta_0, \theta_1) \quad [i = 0, i = 1] \}$$

$$\frac{d}{d\theta_1} J(\theta_0, \theta_1) = \frac{d}{d\theta_1}\left(\frac{1}{2m}\sum_{i=1}^{m}\left[H\left(X^i\right) - Y^i\right]^2\right)$$

$$= \frac{d}{d\theta_1}\left(\frac{1}{2m}\sum_{i=1}^{m}\left[\theta_0 + \theta_1 * X - Y^i\right]^2\right)$$

$$= \left(\frac{1}{m}\sum_{i=1}^{m}\left[\theta_0 + \theta_1 * X - Y^i\right].X\right) = \frac{1}{m}\sum_{i=1}^{m}\left[H\left(X^i\right) - Y^i\right].X$$

$i = 1$

Polymerization Between Gradient and Hypothesis

$Repeat\ until\ convergence$: {
{

$$\theta_0 := \theta_0 - \propto \frac{d}{d\theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 := \theta_1 - \propto \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

}

$Repeat\ until\ convergence$: {
{

$$\theta_0 := \theta_0 - \propto \frac{1}{m} \sum_{i=1}^{m} \left[H\left(X^i\right) - Y^i \right]$$

$$\theta_1 := \theta_1 - \propto \frac{1}{m} \sum_{i=1}^{m} \left[H\left(X^i\right) - Y^i \right] . X$$

}

Polymerization Between Gradient and Hypothesis

$Repeat\ until\ convergence:\{$

$\{$

$$\theta_0 \coloneqq \theta_0 - \propto \frac{d}{d\theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 \coloneqq \theta_1 - \propto \frac{d}{d\theta_1} J(\theta_0, \theta_1)$$

$\}$

$Together$	
θ_0	θ_1
50	0.5
60	0.5

$Repeat\ until\ convergence:\{$

$\{$

$$\theta_0 \coloneqq \theta_0 - \propto \frac{1}{m} \sum_{i=1}^{m} [H(X^i) - Y^i]$$

$$\theta_1 \coloneqq \theta_1 - \propto \frac{1}{m} \sum_{i=1}^{m} [H(X^i) - Y^i].X$$

$\}$

$Repeat\ until\ convergence:\{$

$\{$

$$T_0 \coloneqq \theta_0 - \propto \frac{1}{m} \sum_{i=1}^{m} [H(X^i) - Y^i]$$

$$T_1 \coloneqq \theta_1 - \propto \frac{1}{m} \sum_{i=1}^{m} [H(X^i) - Y^i].X$$

$\theta_0 \coloneqq T_0$

$\theta_1 \coloneqq T_1$

$\}$

TP: Required Libraries and Importing Data

1. Pandas
2. Numpy
3. Matplotlib

Error:

- Unicode Error(EBCDIC)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as pt
from sklearn.model_selection import train_test_split
```

iloc Function

```python
# Chargement des données depuis un fichier CSV et séparation en variables indépendantes (indepX) et dépendantes (depY)
dataset = pd.read_csv("Datasets/Housing_Data.csv")
indepX = dataset.iloc[:, 0]  # Variable indépendante (ex : surface de la maison)
depY = dataset.iloc[:, 1]   # Variable dépendante (ex : prix de la maison)
```

Handling Data
Splitting Data into Train and Test Sets

```
# Division des données en ensembles d'entraînement et de test (80% entraînement, 20% test)
indepX_train, indepX_test, depY_train, depY_test = train_test_split(indepX, depY, test_size=0.2, random_state=42)
```

GD Function

```python
# Fonction de descente de gradient pour optimiser les paramètres du modèle linéaire
def gradientDescent(indepX, depY, init_theta, learning_rate, num_iterations):
    # Mise à jour itérative des paramètres theta pour minimiser l'erreur
    theta = init_theta
    for i in range(num_iterations):
        theta = grad(indepX, depY, theta, learning_rate)
    return theta
```

Gradient Function

```python
# Fonction pour calculer le gradient des paramètres à chaque étape
def grad(indepX, depY, curr_theta, learning_rate):
    # Calcul du gradient basé sur la dérivée partielle de la fonction de coût
    grad = np.zeros(2)  # Gradient pour theta[0] (biais) et theta[1] (poids)
    new_theta = curr_theta
    m = len(indepX)     # Nombre de données
    for i in range(m):
        x = indepX.iloc[i]  # Correction : Utiliser `.iloc` pour éviter les erreurs avec Series
        y = depY.iloc[i]    # Correction : Idem
        grad[0] += (-1/m) * (y - (curr_theta[0] + curr_theta[1] * x))
        grad[1] += (-1/m) * x * (y - (curr_theta[0] + curr_theta[1] * x))
    # Mise à jour des paramètres en fonction du gradient
    new_theta = np.zeros(2)
    temp0 = curr_theta[0] - learning_rate * grad[0]
    temp1 = curr_theta[1] - learning_rate * grad[1]
    new_theta[0] = temp0
    new_theta[1] = temp1
    return new_theta
```

H Function

```
# Fonction hypothèse pour prédire les valeurs à partir des paramètres optimisés
def hyp(theta, indepX):
    return [theta[0] + theta[1] * x for x in indepX]
```

Defining Main Function

```
# Fonction principale pour entraîner le modèle et afficher les résultats
def main():
    # Initialisation des paramètres (theta) à zéro
    init_theta = np.zeros(2)
    learning_rate = 0.05  # Taux d'apprentissage pour contrôler la vitesse de convergence
    num_iterations = 56  # Nombre d'itérations pour optimiser les paramètres

    # Appel de la descente de gradient pour optimiser theta
    theta = gradientDescent(indepX_train, depY_train, init_theta, learning_rate, num_iterations)

    # Prédiction des valeurs avec les paramètres optimisés sur les données de test
    H = hyp(theta, indepX_test)

    #check
    for i in range(len(depY_test)):
        print(float(H[i]))
        #print(depY_test[i])
        print(depY_test.iloc[i])
        print('----------------')
```

Main Function

```python
# Point d'entrée du programme
if __name__ == "__main__":
    main()
```