AI & Applications

Lecturer: Hoang Duc Quy

Email: quyhd@ut.edu.vn

Department: Institute of Information Technology and Electrical- Electronic Engineering (IT3E)

MACHINE LEARNING ALGORITHMs

Linear Regression

What you'll learn?

- Linear regression model
- Cost function
  - Cost function formula
  - Cost function intuition
  - Visualize the cost function
  - Example
- Gradient descent algorithm*
  - Theory and implementation
  - Learning rate
  - GD for linear regression

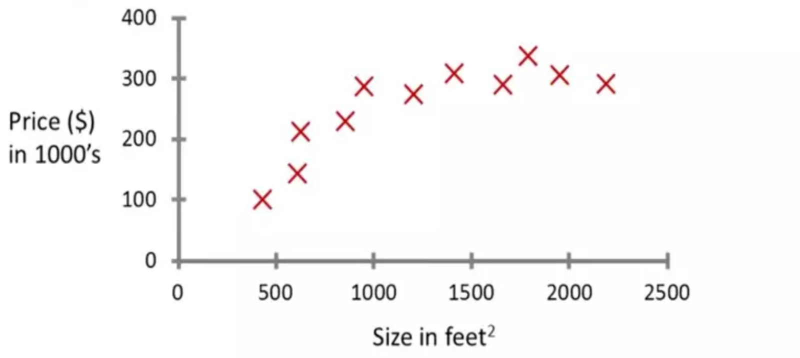
Quy H. D.
quyhd@ut.edu.vn

Terminology

Training set: **data used to train** the model

Size in feet^2	Price in $1000
600	150
615	210
800	250
…	…

Housing price prediction.



x = "input" variable

or feature

y = "output" variable

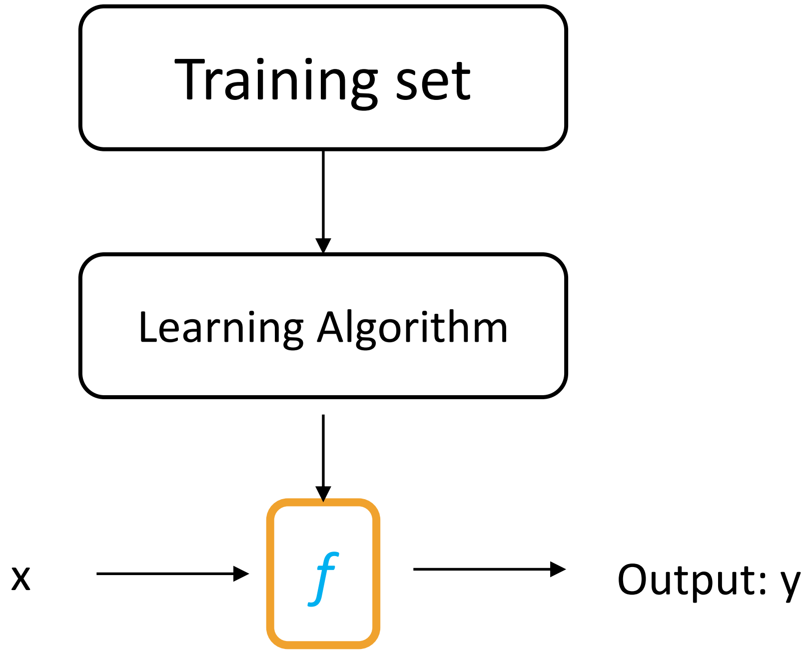
or target

m = number of training examples

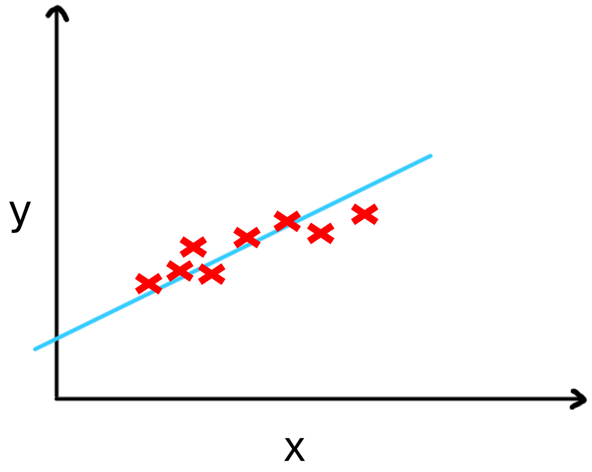
$\left(x^{(i)}, y^{(i)}\right)$ = i^{th} training example

Quy H. D.
quyhd@ut.edu.vn

Linear Regression model



How to represent ***f*?**



Model: $f_{w,b}(x) = wx + b$

- w *is the weight*
- b *is the bias*
- $f_{w,b}(x)$ *is the prediction*
- *Together w and b are called parameters*

Quy H. D.
quyhd@ut.edu.vn

Training set: **data used to train** the model

Size in feet^2	Price in $1000
600	150
615	210
800	250
…	…

x = "input" variable

or feature

y = "output" variable

or target

m = number of training examples

$(x^{(i)}, y^{(i)})$ = i^{th} training example

Model: $f_{w,b}(x) = wx + b$

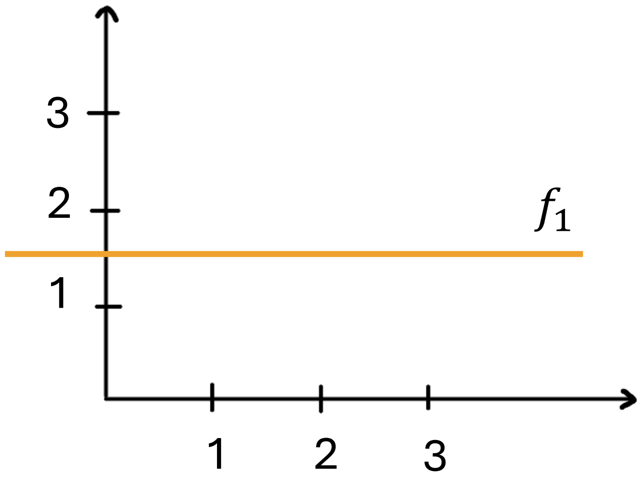
w, b: parameters

Quy H. D.
quyhd@ut.edu.vn

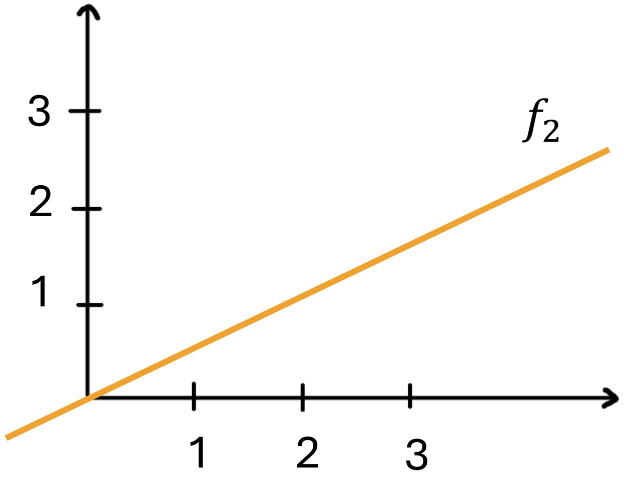
How w and b control the model

Model: $f_{w,b}(x) = wx + b$

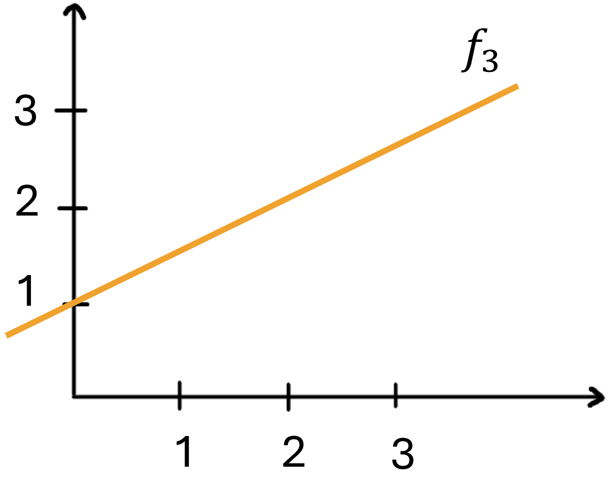
w, b: parameters



w = 0
b = 1.5



w = 0.5
b = 0



w = 0.5
b = 1

Quy H. D.
quyhd@ut.edu.vn

Cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{\mathrm{y}}^{(\mathrm{i})} - \mathrm{y}^{(\mathrm{i})})^2 = \frac{1}{\mathrm{m}}(\mathrm{L}^{(1)} + \mathrm{L}^{(2)} + \dots + \mathrm{L}^{(\mathrm{m})})$$

m: number of training examples
ŷ: model prediction
y: true label of data
J: cost function
L: loss function

Quy H. D.
quyhd@ut.edu.vn

Cost function

Model: $f_{w,b}(x) = wx + b$

w, b: parameters

Cost function:

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

Goal:

$Minimize_{w,b}\, J(w, b)$

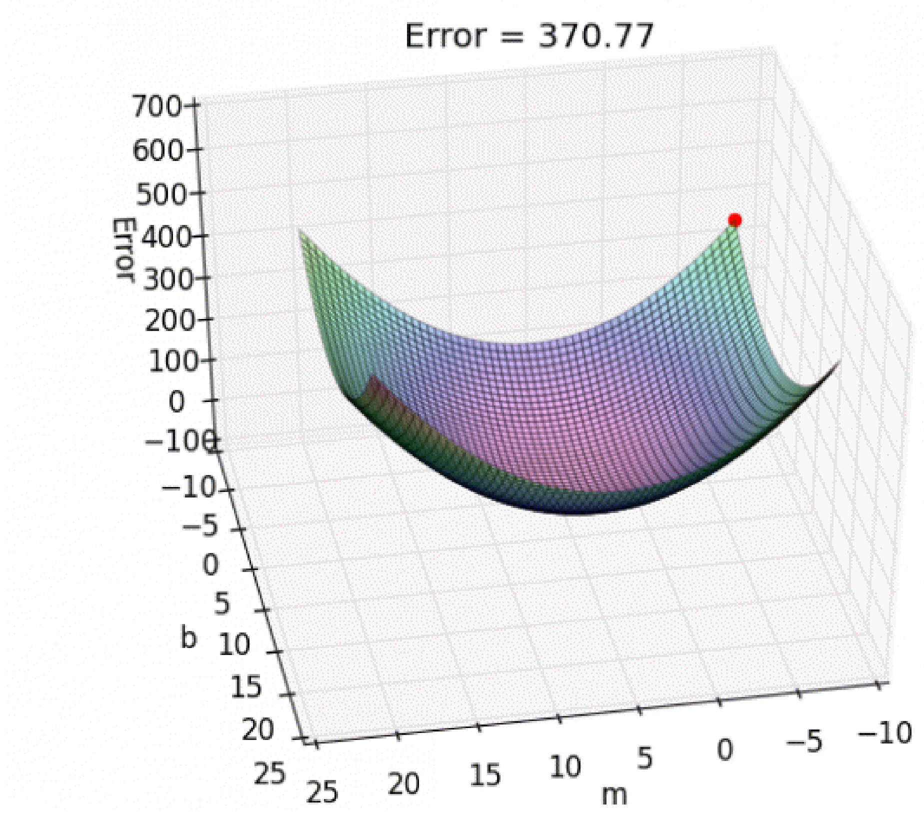
Quy H. D.
quyhd@ut.edu.vn

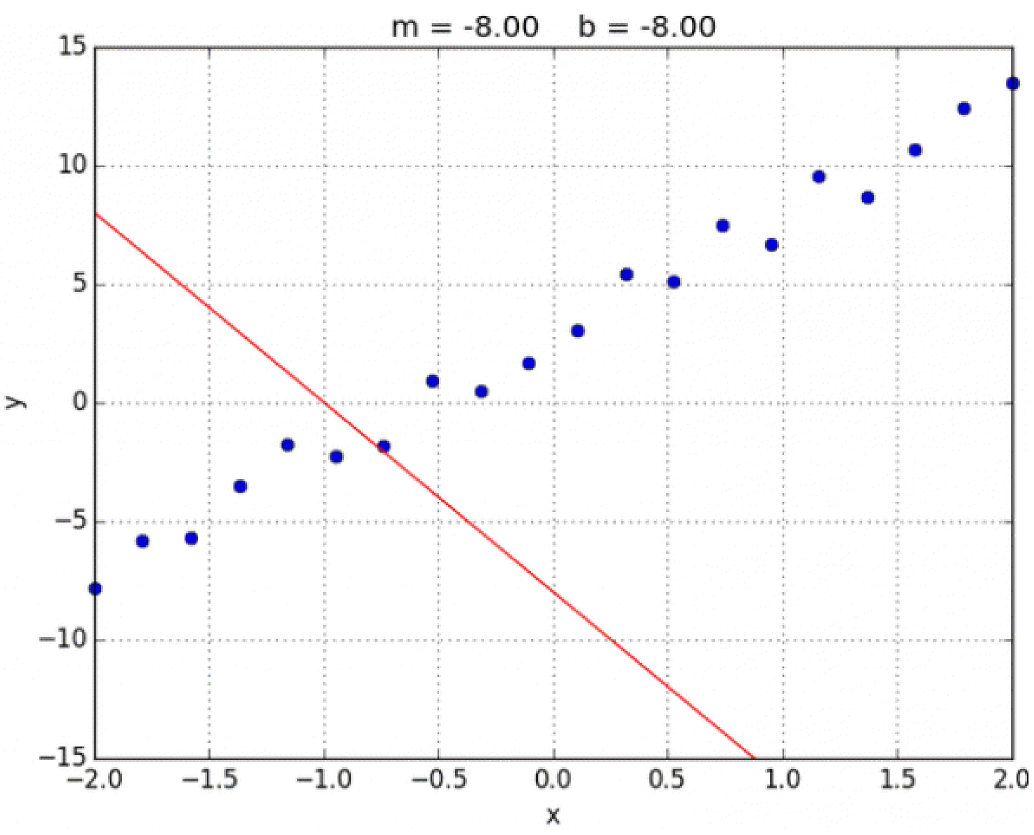
Quiz

Heigh (cm) x	Weight (kg) y	Predicted $\hat{y} = w * x^{(i)} + b$	Square Error $L = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$	Loss function $\frac{1}{m}\sum_{i=1}^{m} L^{(i)}$
147	49	- 2939.5	L1	$= (L^1 + L^2 + \cdots + L^8)/m$ $= J$
150	50	..	L2	
153	51	..	L3	
155	52	..	..	
158	54	..	..	
160	56	..	..	
163	58	..	..	
165	59	..	L8	

1. What is $x^{(5)}$ and its corresponding y
2. Define m?
3. Assume w = -20 , b = 0.5. Calculate $\hat{y}^{(6)}$
4. Assume w = -20 , b = 0.5. Calculate $J_{w,b}$

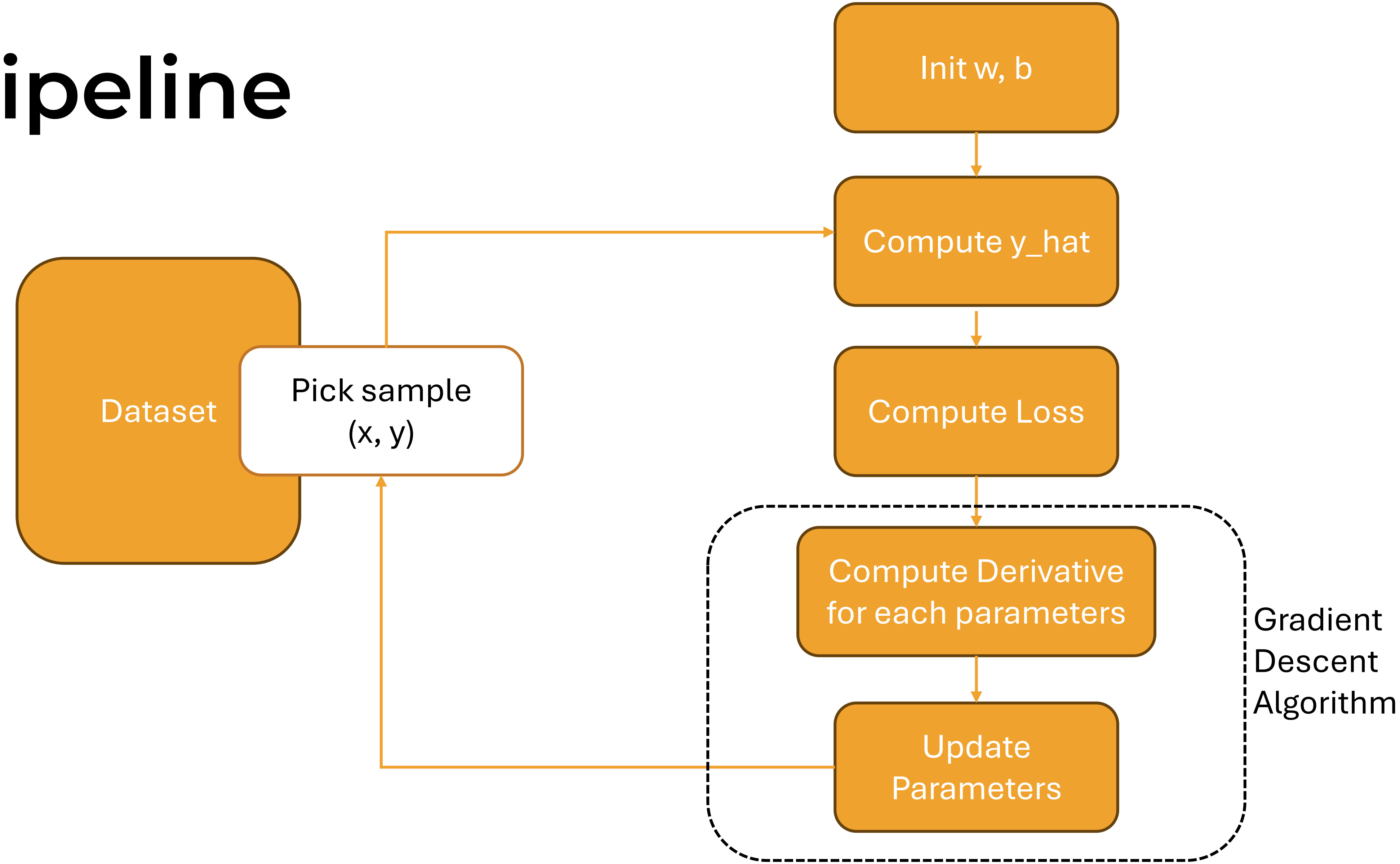
Visualize Cost Function





Quy H. D.
quyhd@ut.edu.vn

LR pipeline



Quy H. D.
quyhd@ut.edu.vn

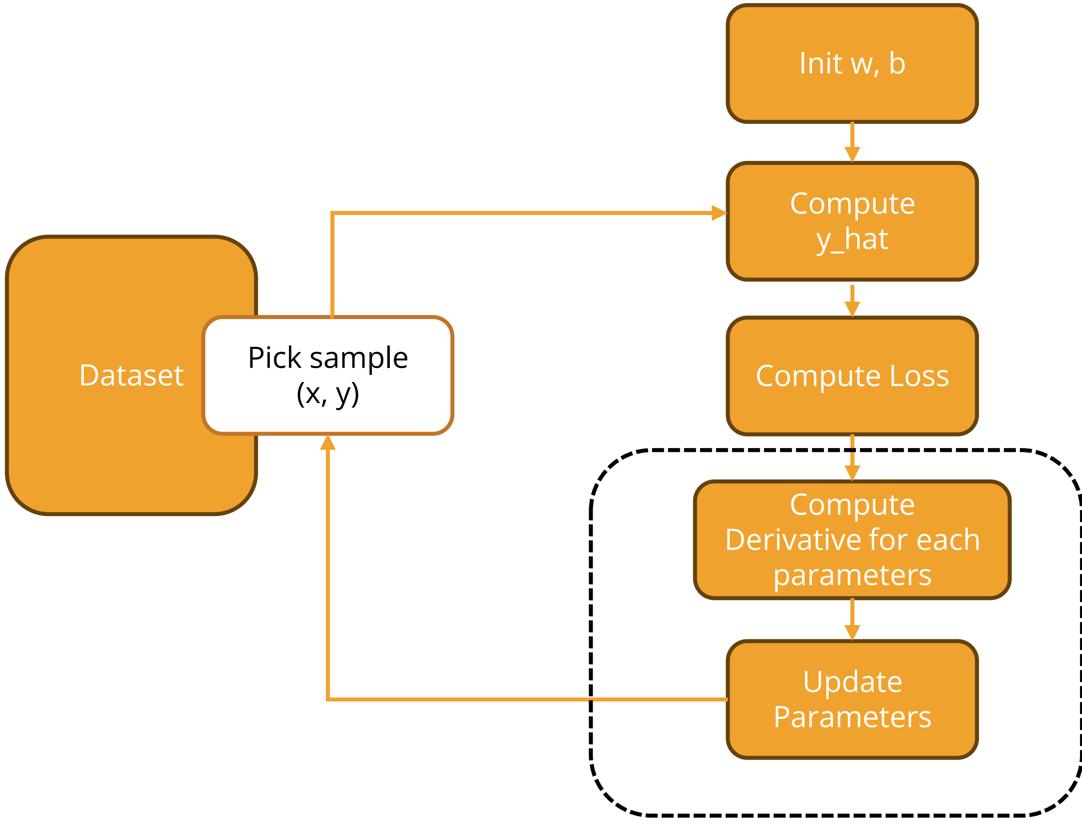
Gradient Descend

Model: $f_{w,b}(x) = wx + b$
w, b: parameters

Mean Square Error function:

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

Goal:
$Minimize_{w,b}\, J(w, b)$



Outline:

- **Start** with some w, b (e.g. w=0, b=0)
- Keep **changing** w, b to **reduce J(w,b) until** we settle or **near a minimum**

Quy H. D.
quyhd@ut.edu.vn

Gradient Descent Algorithm

Repeat these step until convergence:

$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$\frac{\partial J(w,b)}{\partial w} = \frac{1}{m}\left(\frac{\partial L^{(1)}}{\partial w} + \frac{\partial L^{(2)}}{\partial w} + \cdots \frac{\partial L^{(m)}}{\partial w}\right)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\left(\frac{\partial L^{(1)}}{\partial b} + \frac{\partial L^{(2)}}{\partial b} + \cdots \frac{\partial L^{(m)}}{\partial b}\right)$$

$\alpha: learning\ rate$

Quy H. D.
quyhd@ut.edu.vn

Gradient Descent Algorithm

Correct: Simultaneous update

$$temp\_w = w - \propto \frac{\partial}{\partial w} J(w, b)$$

$$temp\_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

$$w = temp\_w$$

$$b = temp\_b$$

Incorrect

$$temp\_w = w - \propto \frac{\partial}{\partial w} J(w, b)$$

$$w = temp\_w$$

$$temp\_b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

$$b = temp\_b$$

Quy H. D.
quyhd@ut.edu.vn

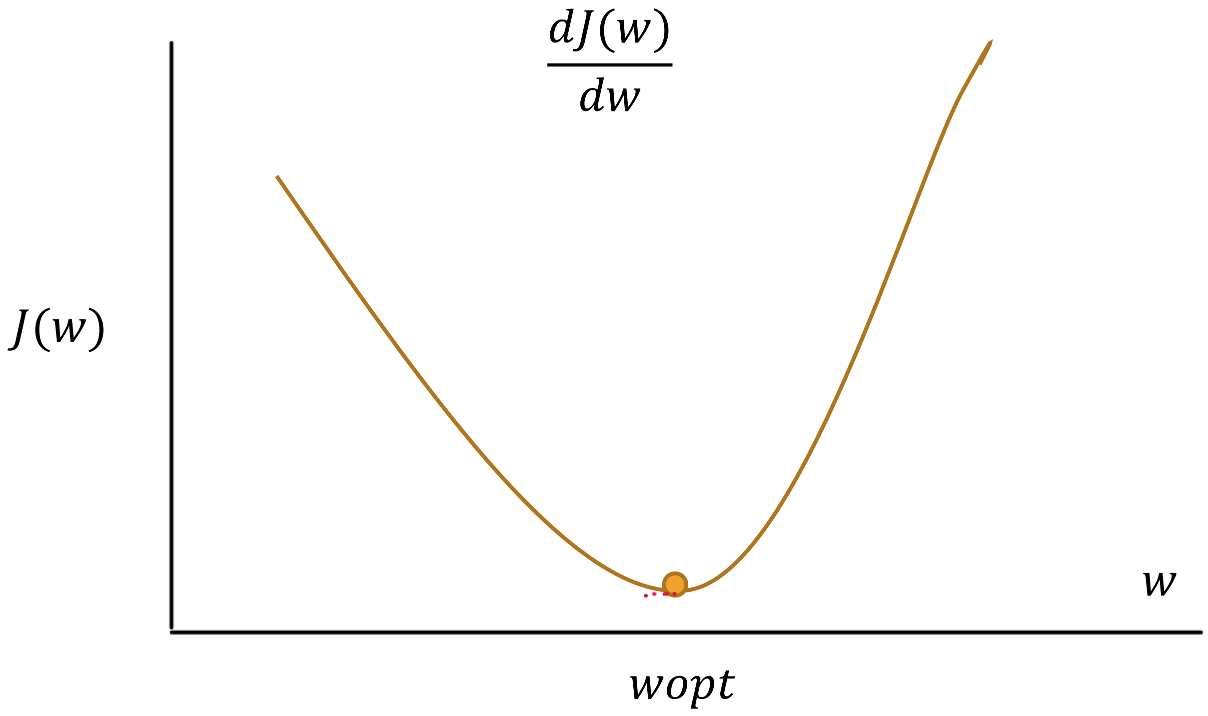
Visualize Cost Function

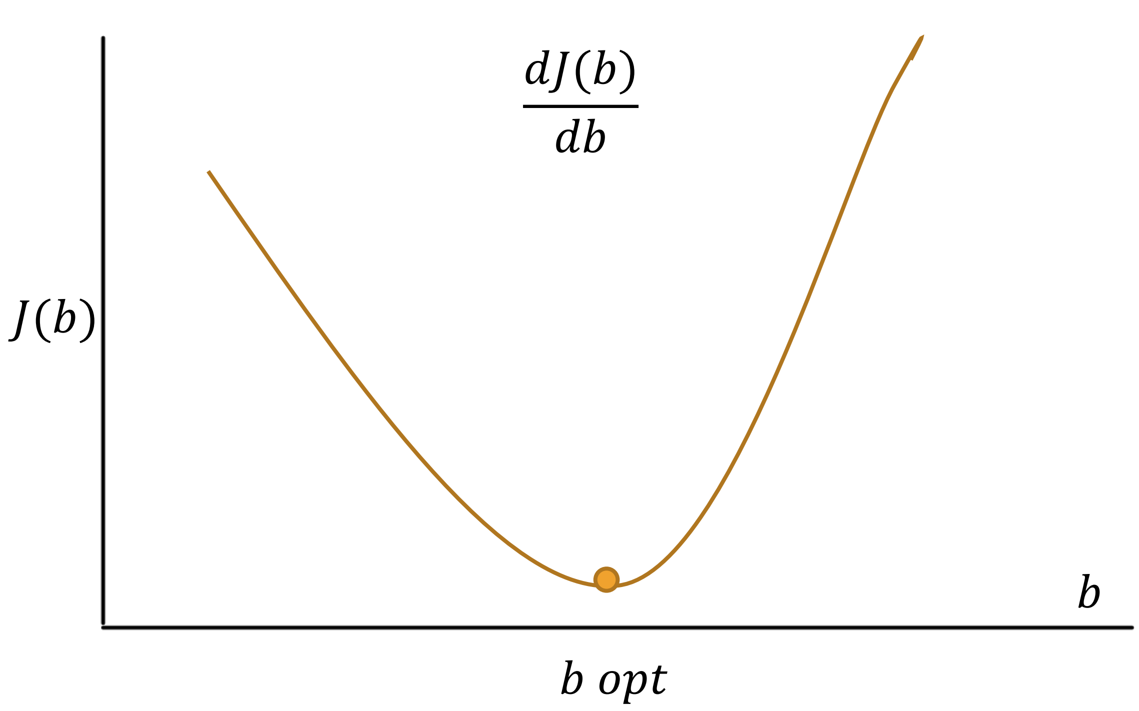
$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$

Assume b = 0, and different w

Assume w = 0, and different b





Quy H. D.
quyhd@ut.edu.vn

GD for linear regression

Repeat these step until convergence:

Model: $f_{w,b}(x) = wx + b$

w, b: parameters

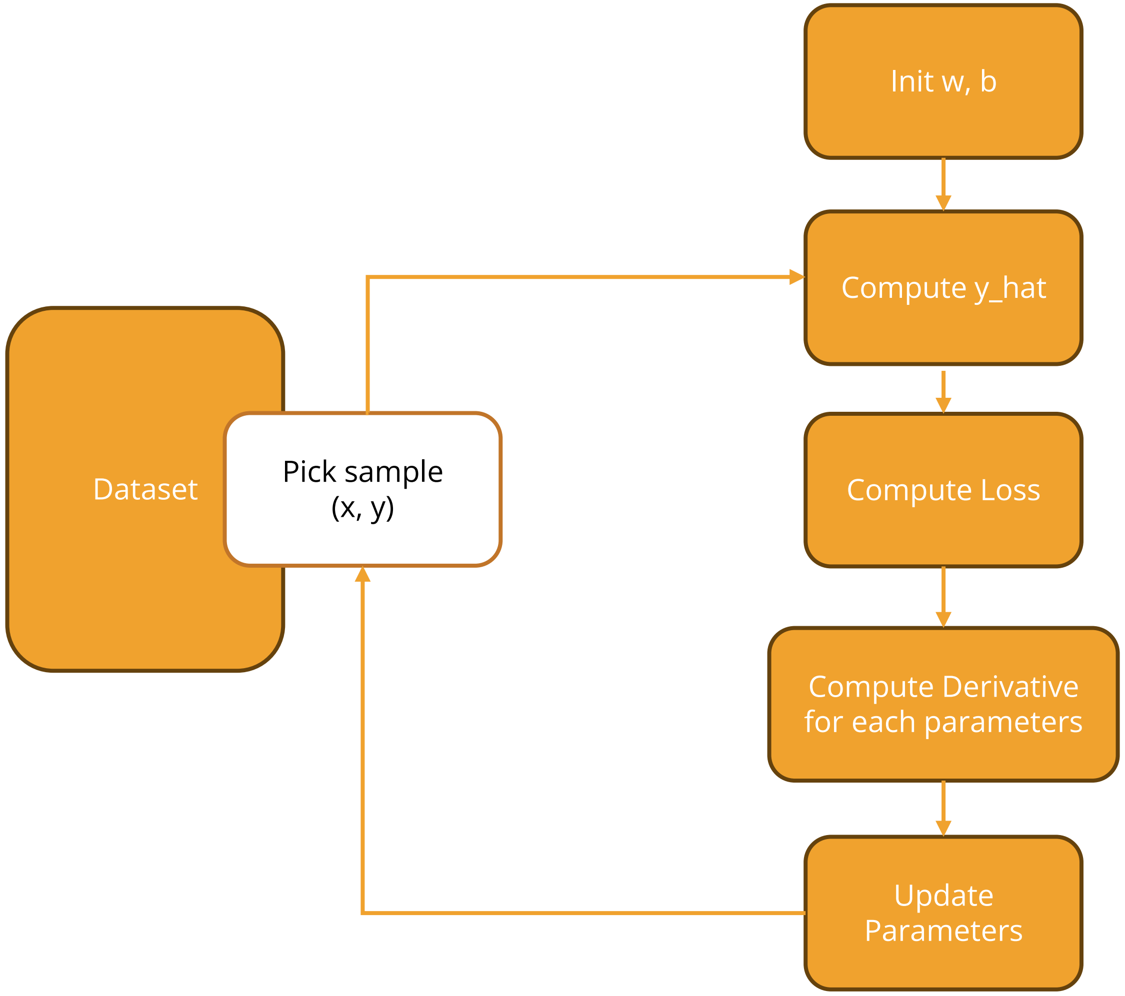
Loss function:

$$L(\mathrm{w}, \mathrm{b}) = \frac{1}{2}(\hat{y} - y)^2$$

Update parameters:

$$w\_new = w\_old - \alpha \frac{\partial}{\partial w} J(w\_old, b\_old)$$

$$b\_new = b\_old - \alpha \frac{\partial}{\partial b} J(w\_old, b\_old)$$



Quy H. D.
quyhd@ut.edu.vn

Batch GD

$$w = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$$

Stochastic GD

$$w = w - 2\alpha (\hat{y}^{(i)} - y^{(i)}) x^{(i)}$$

$$b = b - 2\alpha (\hat{y}^{(i)} - y^{(i)})$$

Quy H. D.
quyhd@ut.edu.vn

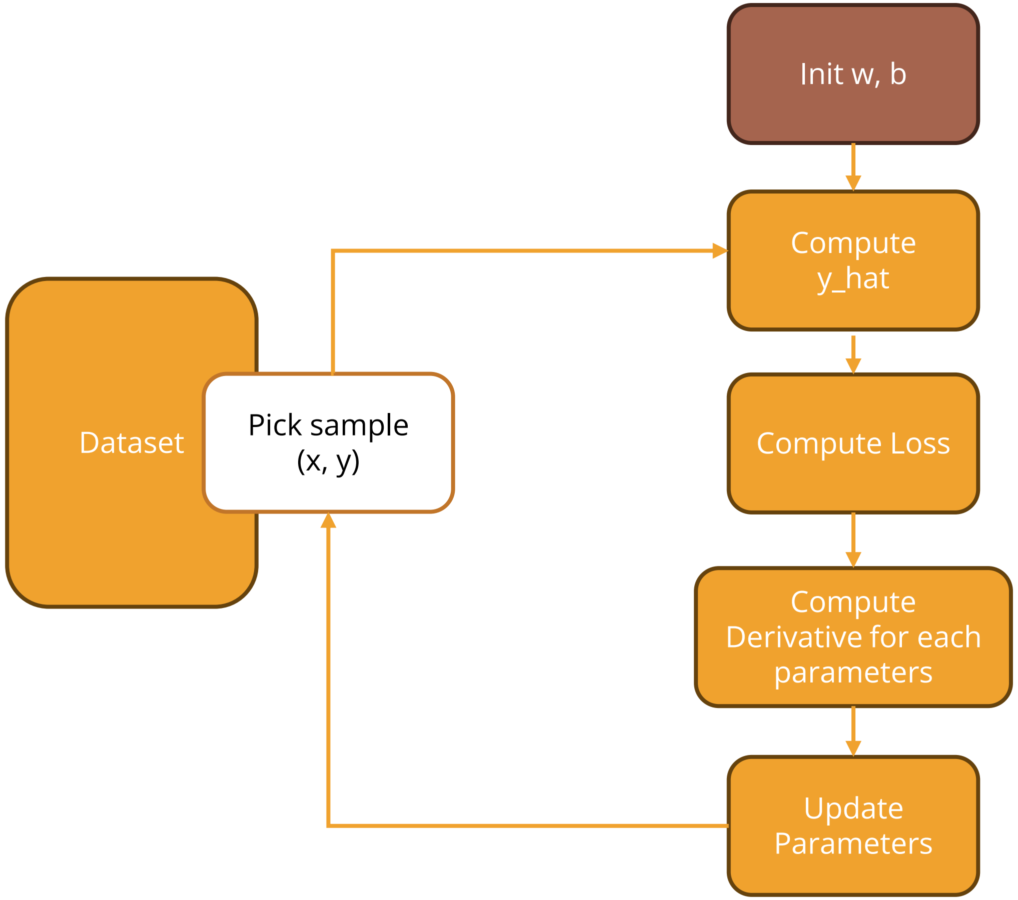
Quiz

Heigh (cm) x	Weight (kg) y	Predicted y $\hat{y}^{(i)} = w * x^{(i)} + b$	Square Error $L = \frac{1}{2}(\hat{y} - y)^2$	Loss function $\frac{1}{m}\sum_{i=1}^{m} L^{(i)}$
147	50	..	..	..
150	51			
151	51			
155	53			
158	54			
160	56			
163	58			
165	59			

1. Assume w = 0, b = 0, alpha = 0.00002
2. J(w,b) = ?
3. Perform SGD for 1 iteration. Calculate updated_w, updated_b, J(update_w, updated_b)
4. Perform BGD for 1 iteration. Calculate updated_w, updated_b, J(update_w, updated_b)

Coding exercise

Step 1: Initialize parameters



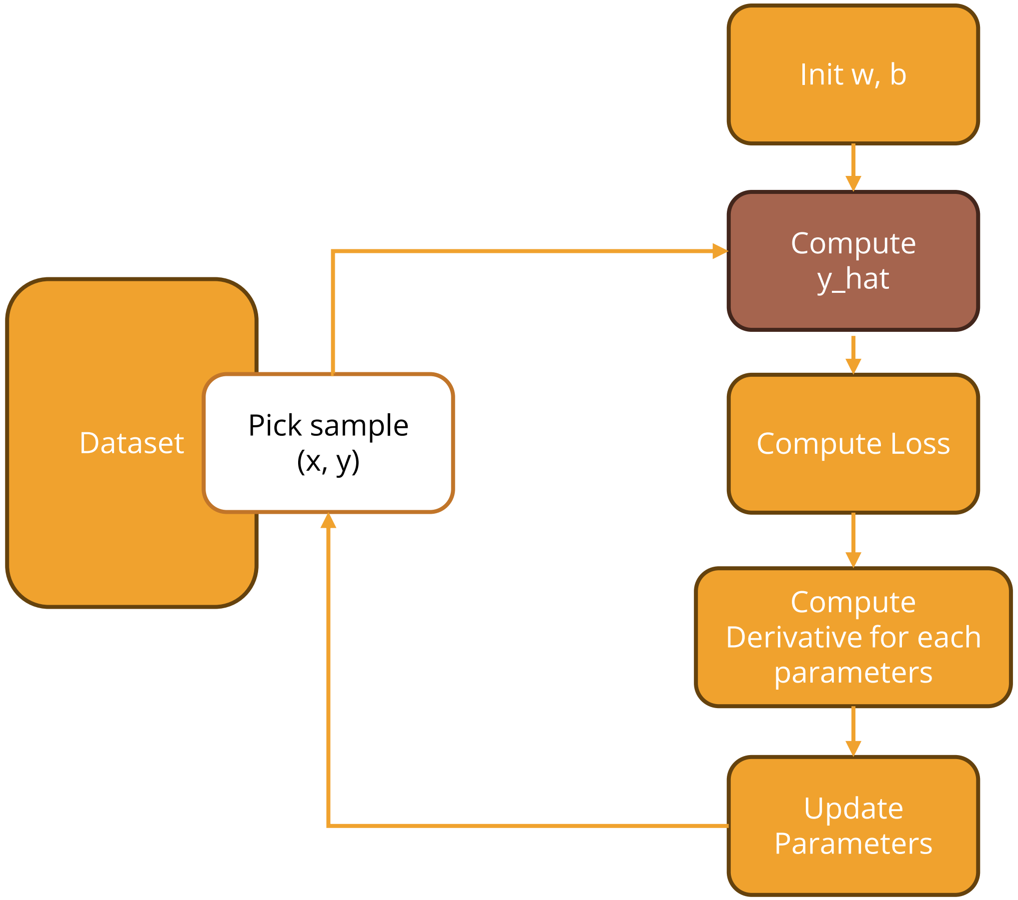

```
import pandas as pd
import numpy as np

x = pd.read_csv('data.csv')['x'].values
y = pd.read_csv('data.csv')['y'].values

b = 0
w = 0
lr = 0.00002
```

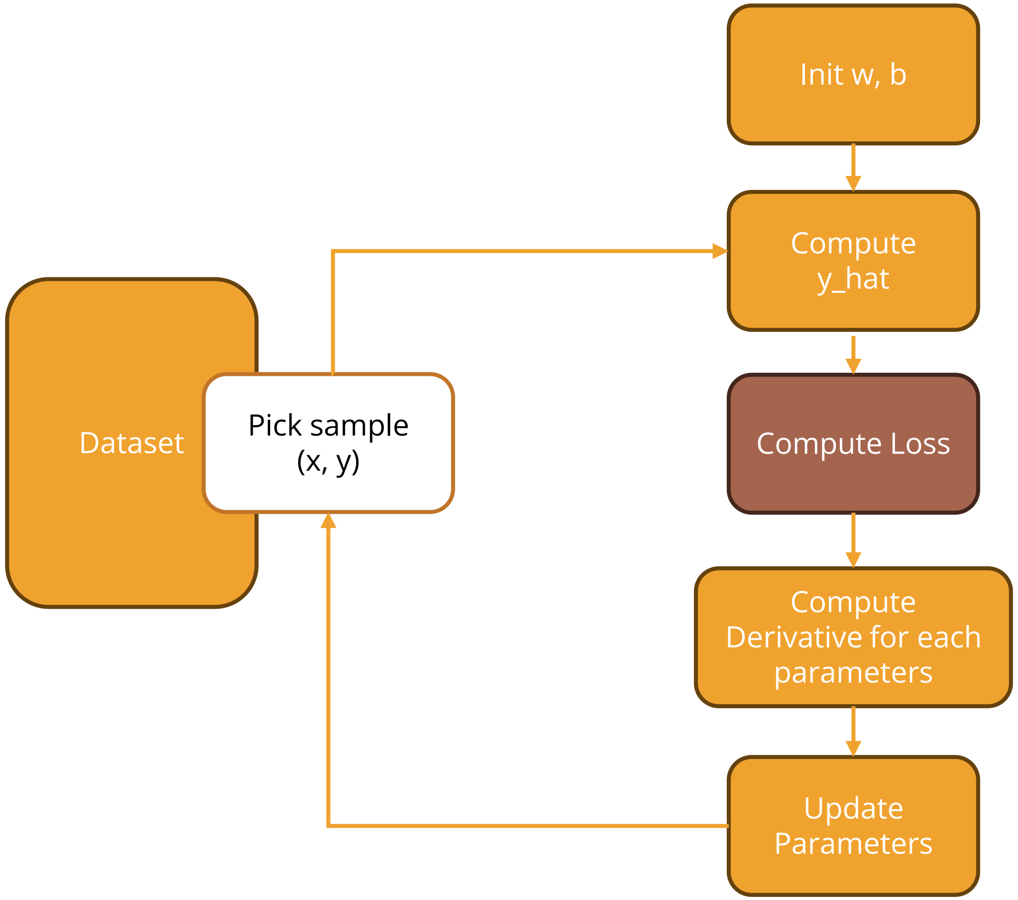
Quy H. D.
quyhd@ut.edu.vn

Step 2: Calculate predict



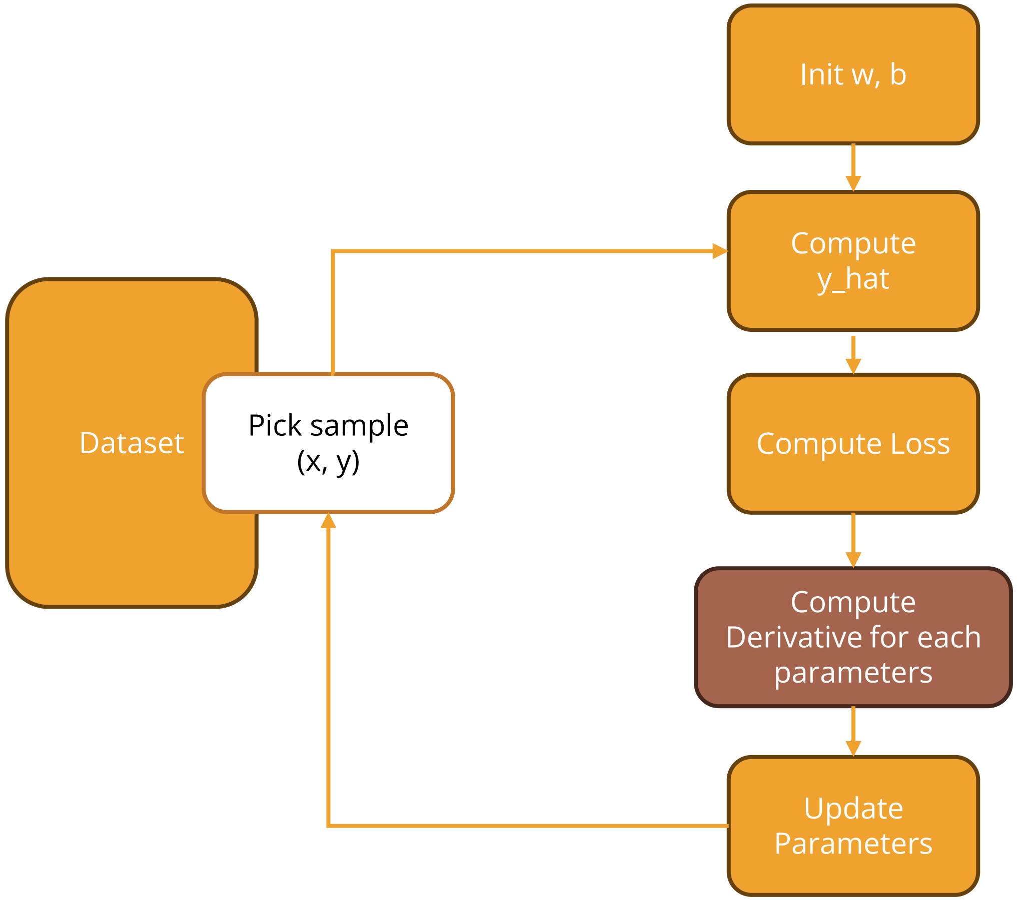

```python
def predict(x, w, b):
    return x * w + b
```

Step 3: Calculate loss




```python
def compute_loss(y_hat, y):
    return (y_hat - y) ** 2
```

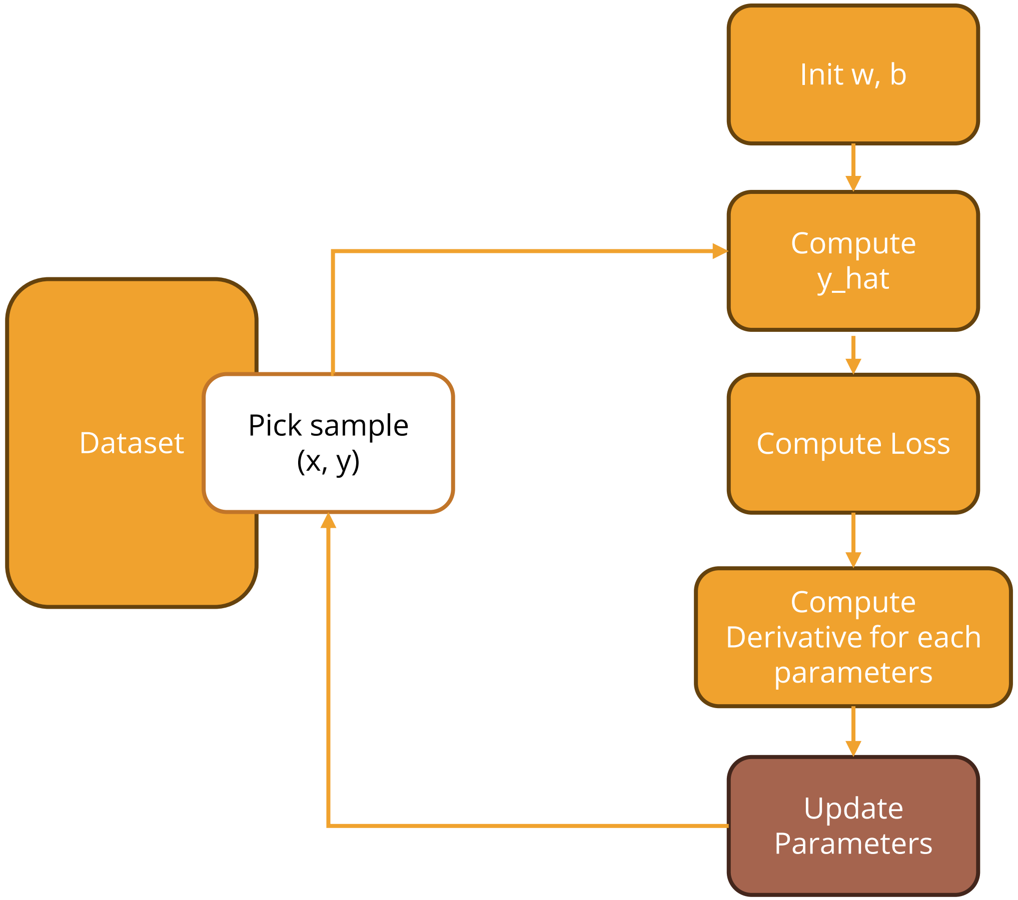
Step 4: Compute gradient




```python
def compute_gradient(y_hat, y, x):
    dw = 2 * x * (y_hat - y)
    db = 2 * (y_hat - y)

    return dw, db
```

Step 5: Update parameters




```python
def update_weight(w, b, dw, db, lr):
    new_w = w - lr * dw
    new_b = b - lr * db

    return new_w, new_b
```

THANK YOU

Quy H. D.
quyhd@ut.edu.vn