ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Linear Regression and Gradient Descent Algorithm

Outline

1. Supervised learning for classification and regression problems
2. Linear Regression (LR) model
3. Training LR by Gradient Descent Algorithm
4. Implementation LR

Types of Machine Learning

- Supervised Learning:
  - Input: X, y pre-classified training examples
  - Given a new observation x, what is the best label y?
- Unsupervised Learning:
  - Input: X
  - Given a set of observations, cluster or summarize them
- Semi-supervised Learning
- Re-inforcement Learning:
  - Determine what to do based on rewards and punishments.

Supervised Learning

- Supervised learning is where you have input variables (x) and an output variable (Y) and you use **an algorithm to learn** the mapping function from the input to the output.

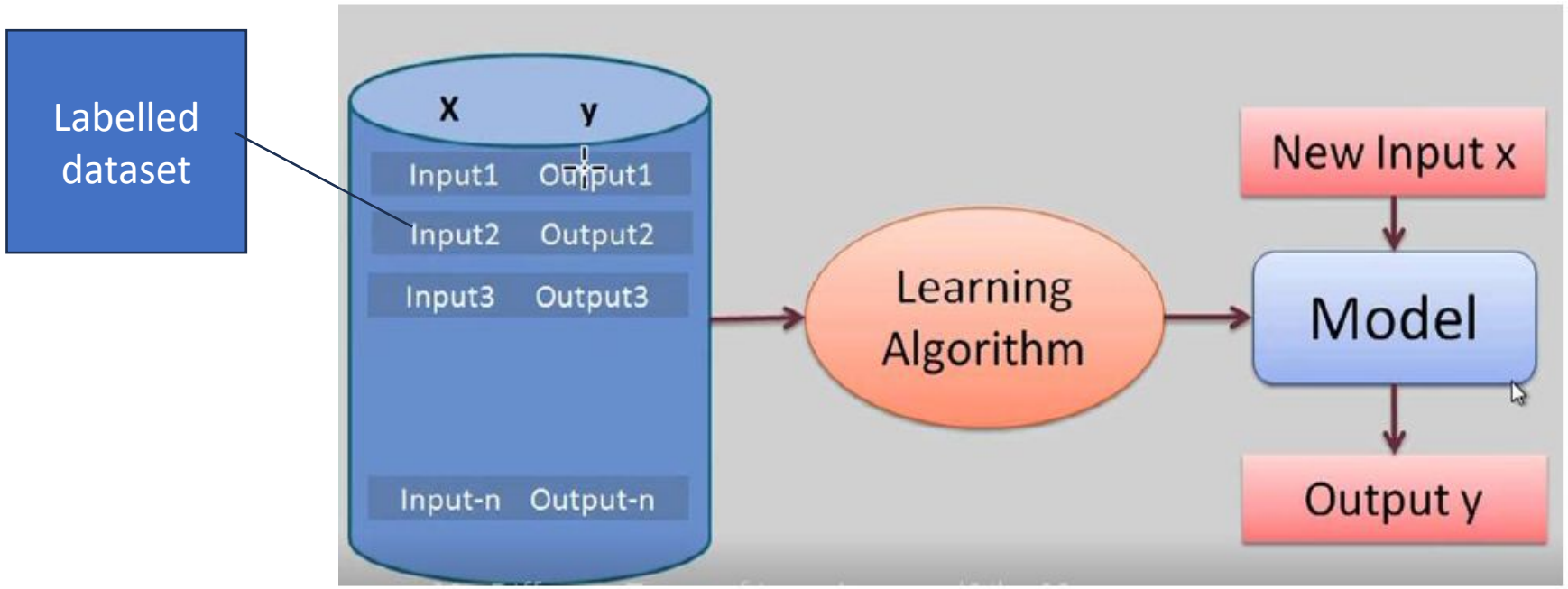
$$Y = f(x)$$

- The goal is to **approximate the mapping function** so well that when you have new input data (x) that you can predict the output variables (Y) for that data.
- It is called supervised learning because **the process of an algorithm learning from the training dataset** can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

* https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Supervised Learning

- What is labelled dataset
  - Data for which you already know the target answer
  - Example: dataset of animal pictures, each image would be labeled with the type of animal it is, like cat, dog, bird



Example of house price labelled dataset

Size (Square Meters)	Price (VND)
50	1,500,000,000
70	2,100,000,000
80	2,400,000,000
60	1,800,000,000
90	2,700,000,000
120	3,600,000,000
40	1,200,000,000
110	3,300,000,000
75	2,250,000,000

Example of spam email labelled dataset

Email Subject	Email Body Excerpt	Label
Mua ngay! Giảm giá lớn cho các sản phẩm công nghệ!	Chỉ hôm nay, giảm giá 50% cho tất cả...	Thư rác
Cập nhật thông tin tài khoản ngân hàng của bạn	Kính gửi Quý khách, chúng tôi cần bạn cập nhật lại...	Thư hợp lệ
Bạn đã trúng thưởng!	Xin chúc mừng! Bạn đã trúng một chuyến du lịch...	Thư rác
Họp mặt cuối năm	Kính mời anh/chị tham dự buổi họp mặt cuối năm...	Thư hợp lệ
Đặt hàng ngay để nhận quà tặng	Mua 1 tặng 1! Đừng bỏ lỡ cơ hội để sở hữu...	Thư rác

Supervised Learning

- Given:
  - a set of input features X1, ..., Xn
  - A target feature Y
  - a set of training examples where the values for the input features and the target features are given for each example
  - a new example, where only the values for the input features are given
- Predict the values for the target features for the new example.

Types of Supervised Learning

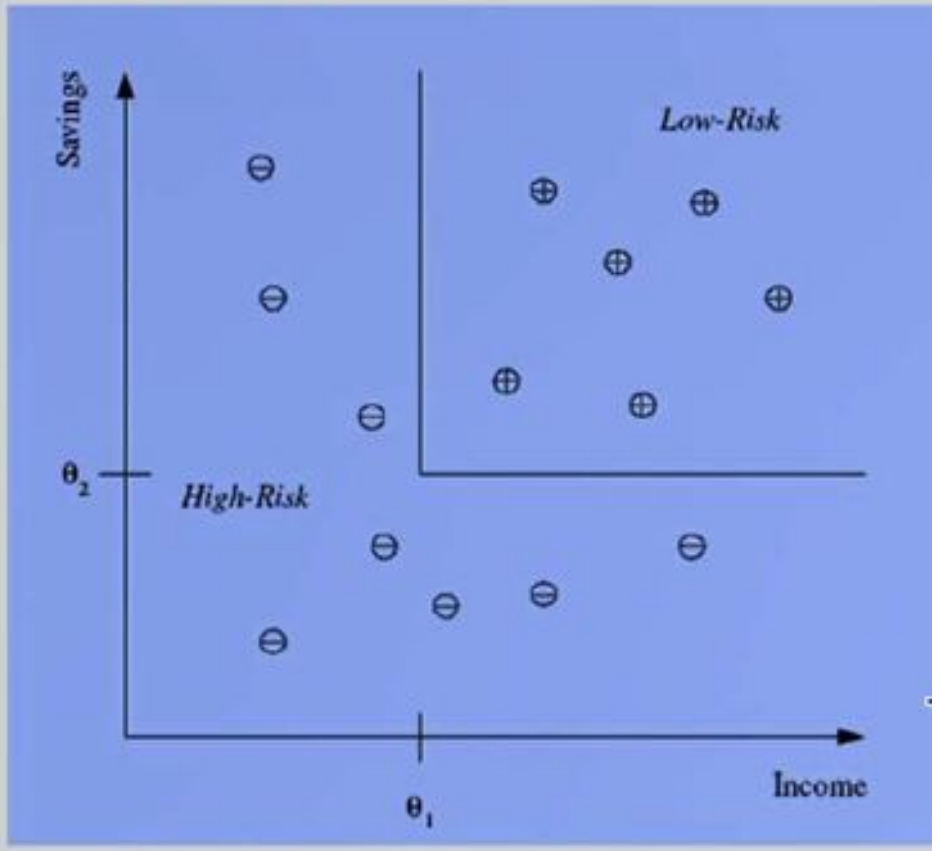
- Supervised learning problems can be further grouped into regression and classification problems.
- **Classification**:
  - The output variable is a category (discrete value), such as “red” or “blue” or “disease” and “no disease”.
  - Machine is trained to classify something into some class
    - Ex. Patient has disease or not
    - Ex. Email is spam or not
- **Regression**:
  - The output variable is a continuous value, such as “dollars” or “weight”.
  - Machine is trained to predict some value like price, weight, or height.

- Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.
- Other examples of supervised machine learning algorithms:
  - Linear regression for regression problems.
  - Random forest for classification and regression problems.
  - Support vector machines for classification problems.

Supervised Learning: Classification

Example: Credit scoring

Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* > θ_1 AND *savings* > θ_2
THEN low-risk ELSE high-risk

Supervised Learning: Regression

Example: Price of a used car

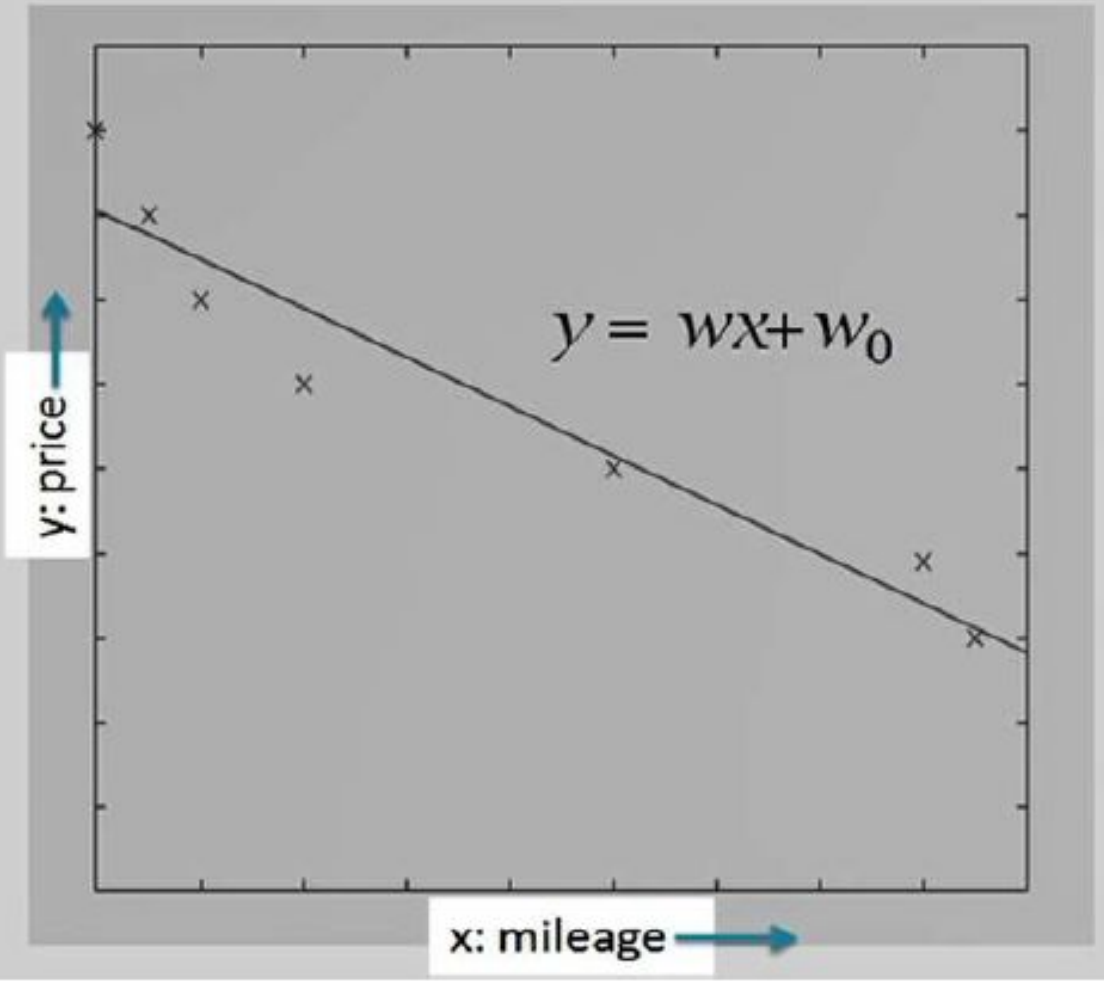
x : car attributes

y : price

$$y = g(x, \theta)$$

$g(\ )$ model,

θ parameters



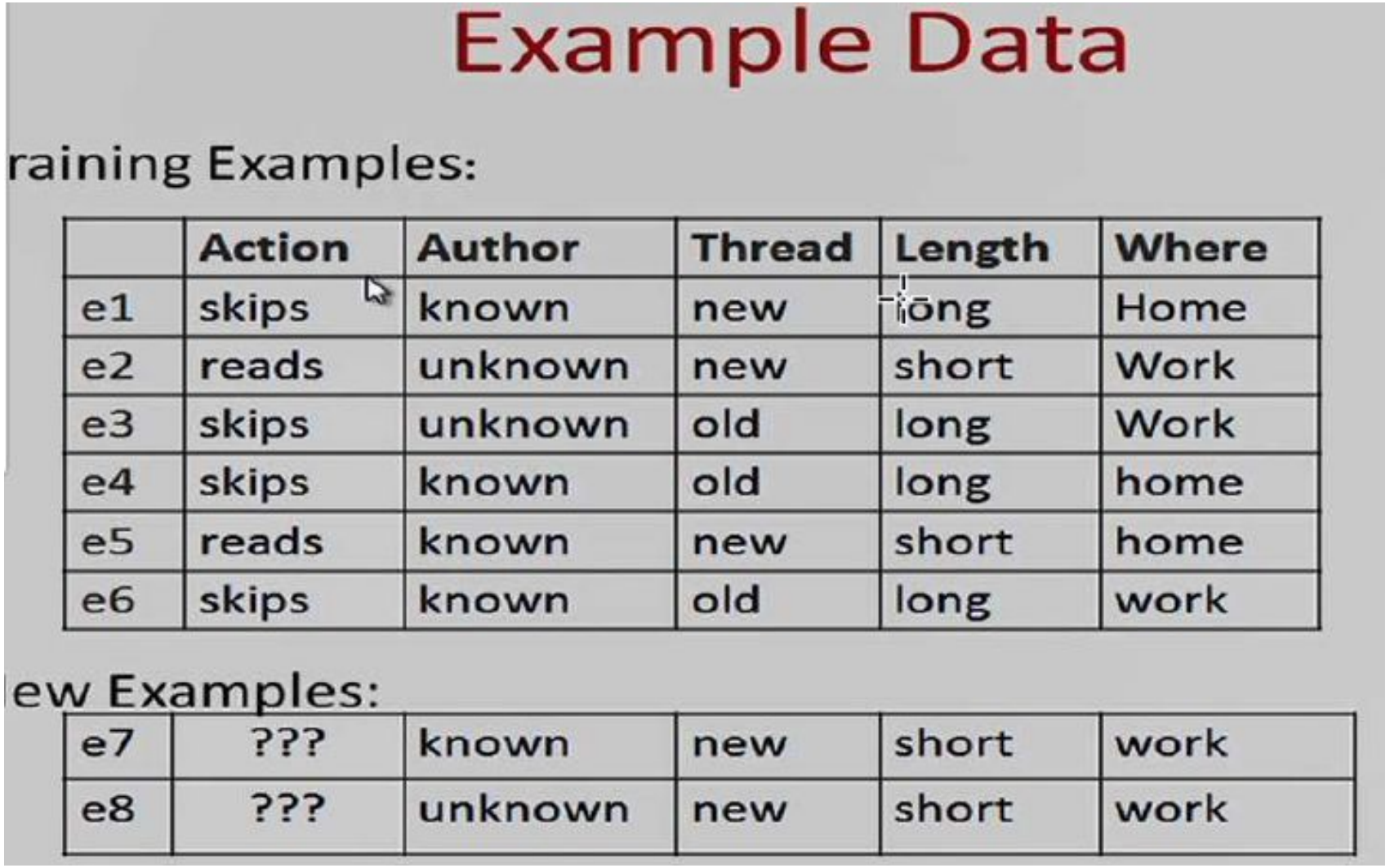
Regression vs Classification

- Classification is about predicting a label and Regression is about predicting a quantity.
- Classification is the problem of predicting a discrete class label output for an example. The predicted label is from a set of predefined labels.
  - For example: predict an email is spam or ham
- Regression is the problem of predicting a continuous quantity output for an example.
  - For example: predict house price from its features (i.e. information).

Features

- Observations are represented by a set of quantifiable properties, called features
- They can be
  - Categorical:
    - Animal species (e.g., cat, dog, bird)
    - Color (e.g., red, blue, green)
    - Genre of music (e.g., rock, jazz, pop)
  - ordinal
    - Size (e.g., large, medium, small)
    - Rating scales (e.g., low, medium, high)
    - Educational levels (e.g., elementary, high school, college)
  - integer-value
    - The number of words in a text
    - Number of likes on a post
    - Number of products sold in a day
  - real-valued
    - Height of a person
    - Temperature
    - Stock prices

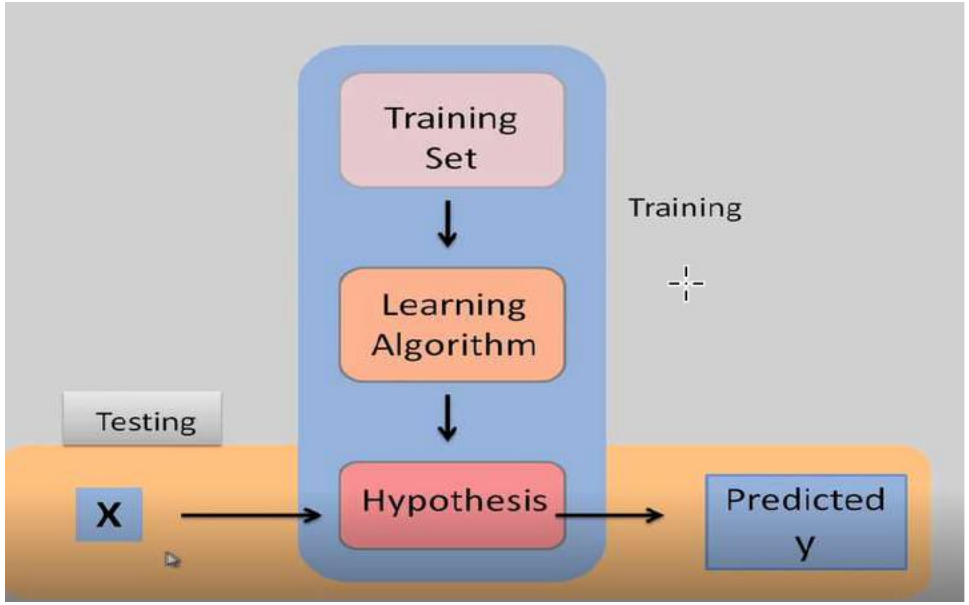
Example Data



	Action	Author	Thread	Length	Where
e1	skips	known	new	long	Home
e2	reads	unknown	new	short	Work
e3	skips	unknown	old	long	Work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

e7	???	known	new	short	work
e8	???	unknown	new	short	work

Supervised Learning: Training and Testing Phases



- **Two phases:**
  - Training: the model is developed,
  - Testing: the model is evaluated. The input data (X) is fed into the hypothesis during the testing phase to generate the predicted outputs.
- **Training Set:**
  - collection of data, including both the input features (often referred to as X) and the output labels (often referred to as Y).
  - The training set is used to teach the machine learning model how to predict the output from the input.
- **Learning Algorithm:**
  - represents the machine learning model that is being used.
  - The learning algorithm takes the training set and learns from it by adjusting its parameters to minimize errors in prediction.
- **Hypothesis:**
  - is the function or model created by the learning algorithm.
  - It represents the learned relationship between the input features and the output labels.
- **Predicted Y:** predicted output for new, unseen inputs.
- **Testing:** This is the phase where the trained model (hypothesis) is tested on a new set of inputs (testing set) to evaluate how well it has learned to generalize from the training data to unseen data. The actual output from testing is compared against the predicted Y to measure the performance of the model.

Regression (Hồi qui)

- From Height, predict Weight?

Height(cm)	Weight(kg)
147	49
150	50
153	51
155	52
158	54
160	56
163	58
165	59

Training data

(x1, y1)

(x2, y2)

.

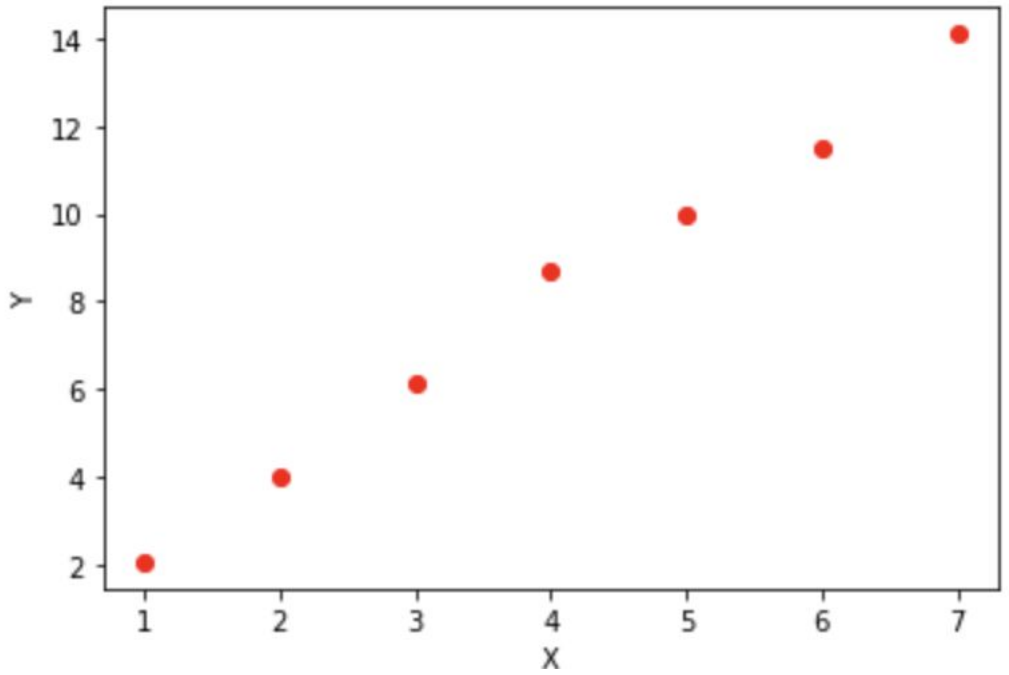
.

.

(xn, yn)

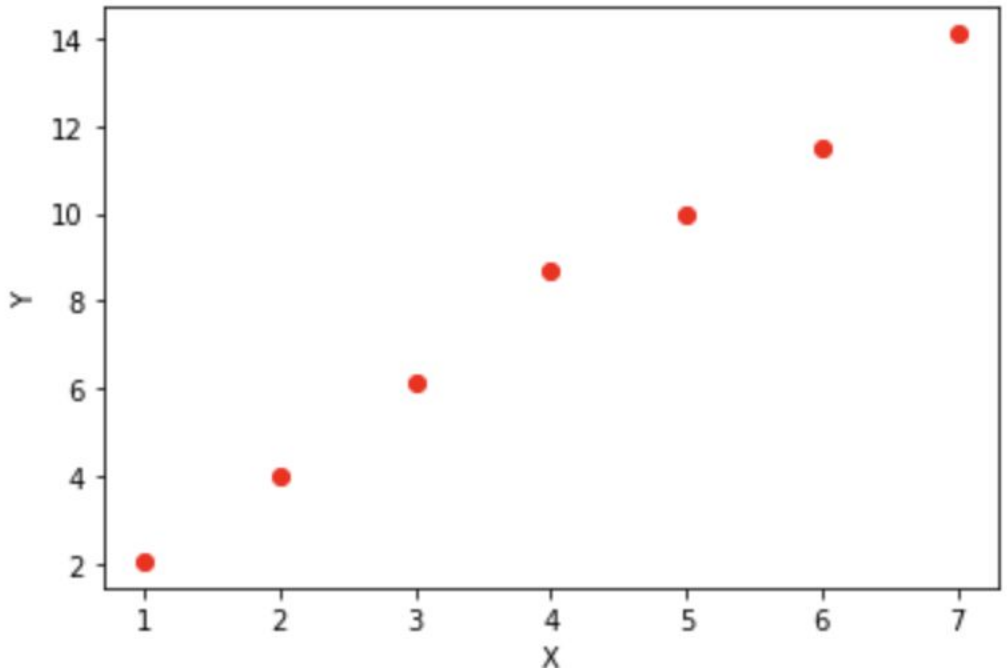
Linear Regression

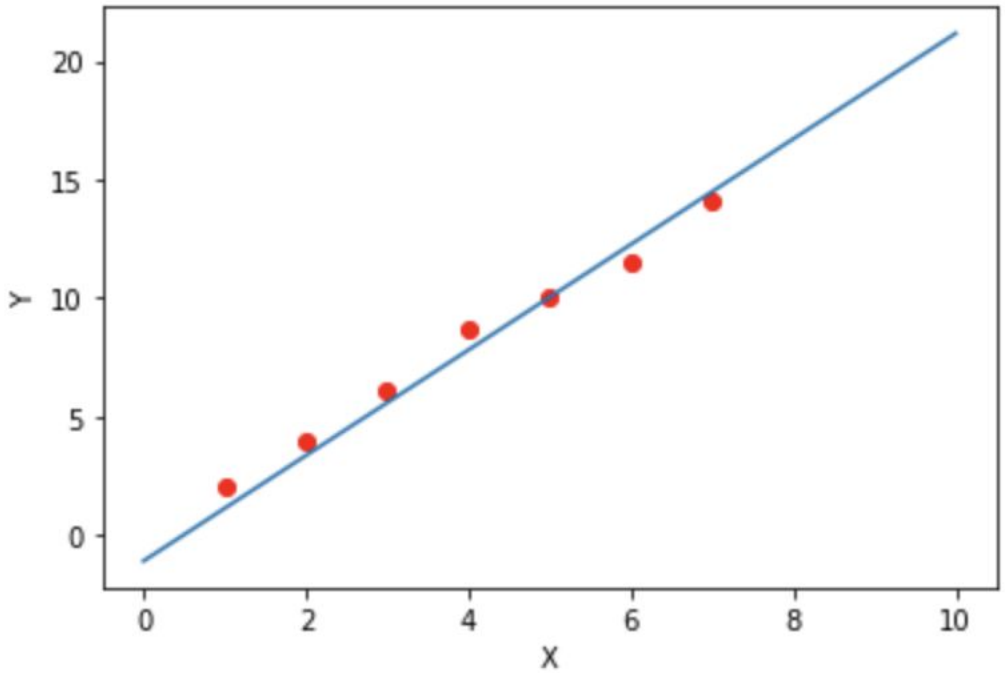
- How is the relationship between Height and Weight?



Linear Regression

Suppose that Weight linearly depends on Height



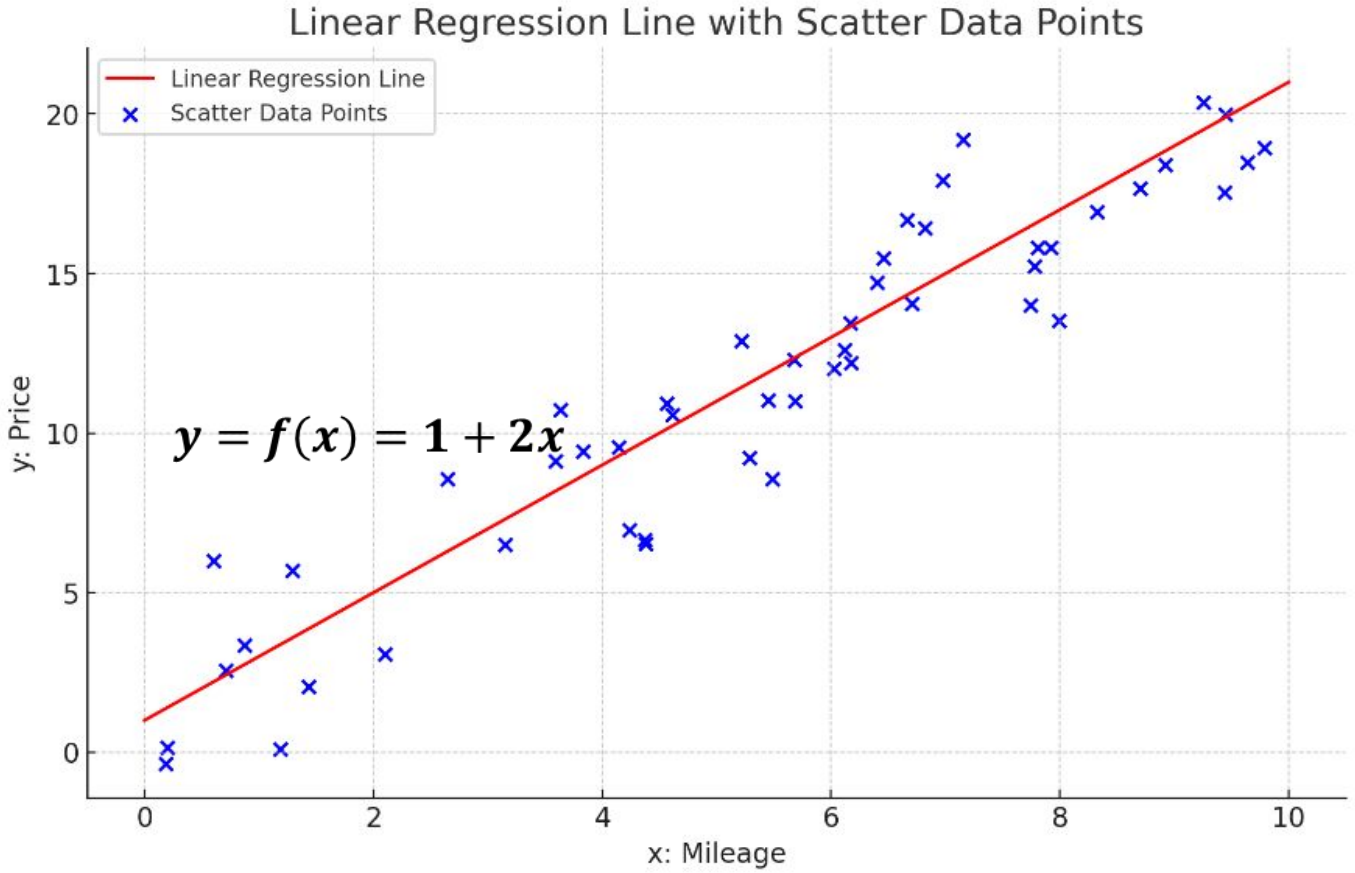


Simple Linear Regression: one variable

Training data: $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$

- $y = f(x) = w_0 + w_1 x$

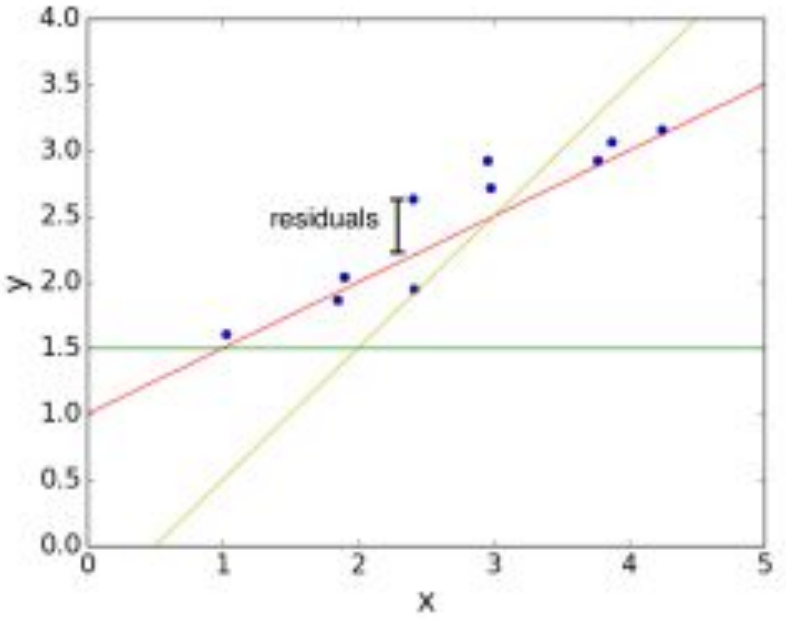
Learn w_0 and w_1



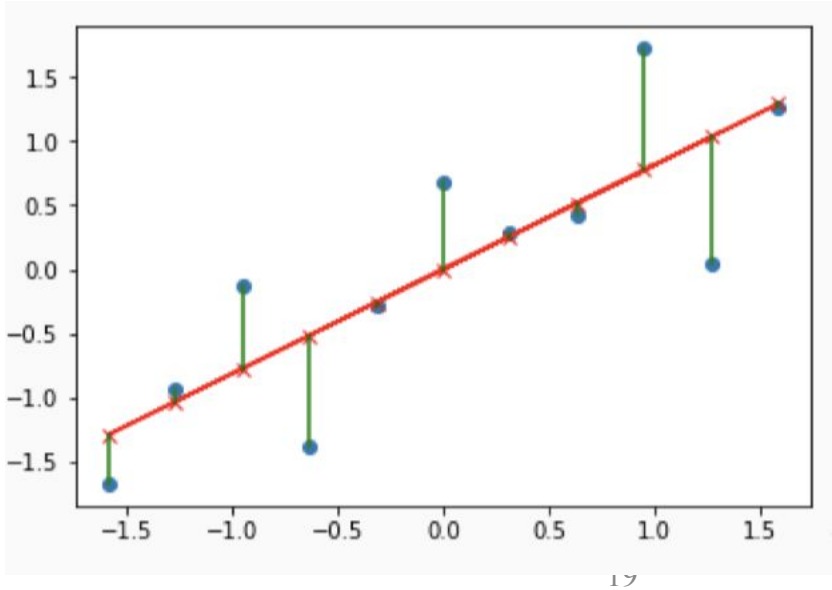
Loss Function

- Training data: $(x_1, y_1), (x_2, y_2). . . (x_n, y_n)$
- Simple linear regression model

$$y = f(x) = w_0 + w_1 x$$



- Prediction by the model: x -> f(x)
- For every training example (x_i, y_i) we compare $f(x_i)$ with y_i
- Loss Function
  - a mathematical function that measures how well the model's predictions match the actual data.
  - it quantifies the error between the predicted values and the observed values in the training data.

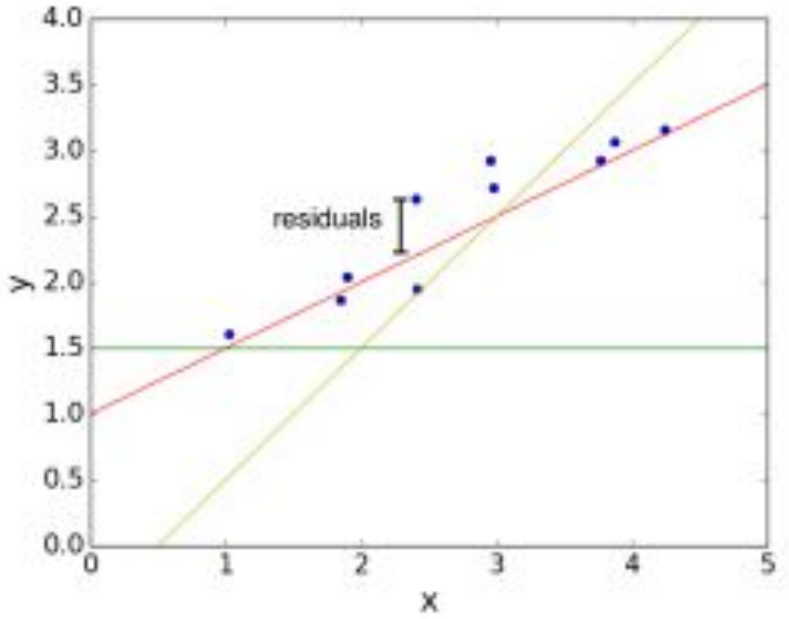


- Loss Function
  - a mathematical function that measures how well the model's predictions match the actual data.
  - it quantifies the error between the predicted values and the observed values in the training data.

Loss Function

- Training data: $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$
- Simple linear regression model

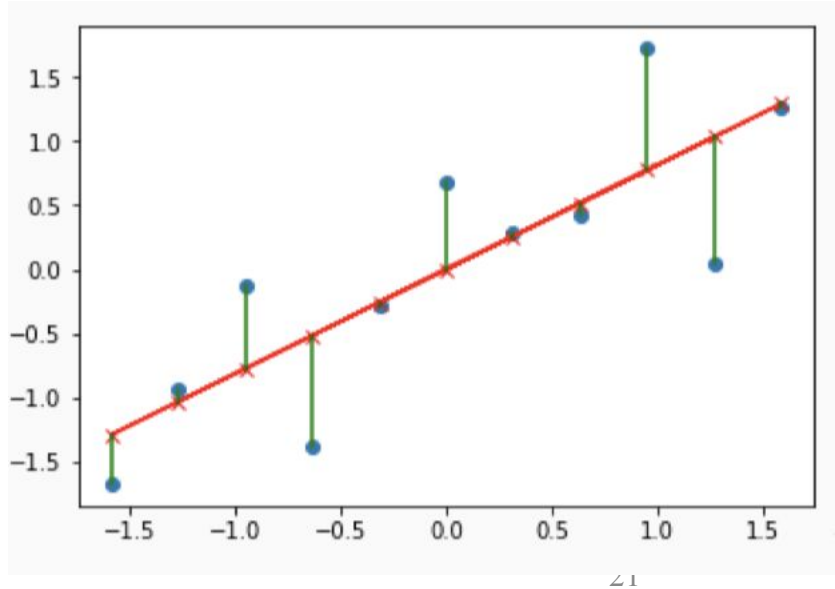
$$y = f(x) = w_0 + w_1 x$$



- Prediction by the model: $x_i \rightarrow \hat{y}_i = f(x_i)$
- For every training example (x_i, y_i) we compare $\hat{y}_i = f(x_i)$ with y_i

squared error $\frac{1}{2}(\hat{y}_i - y_i)^2$

Residual to be minimized

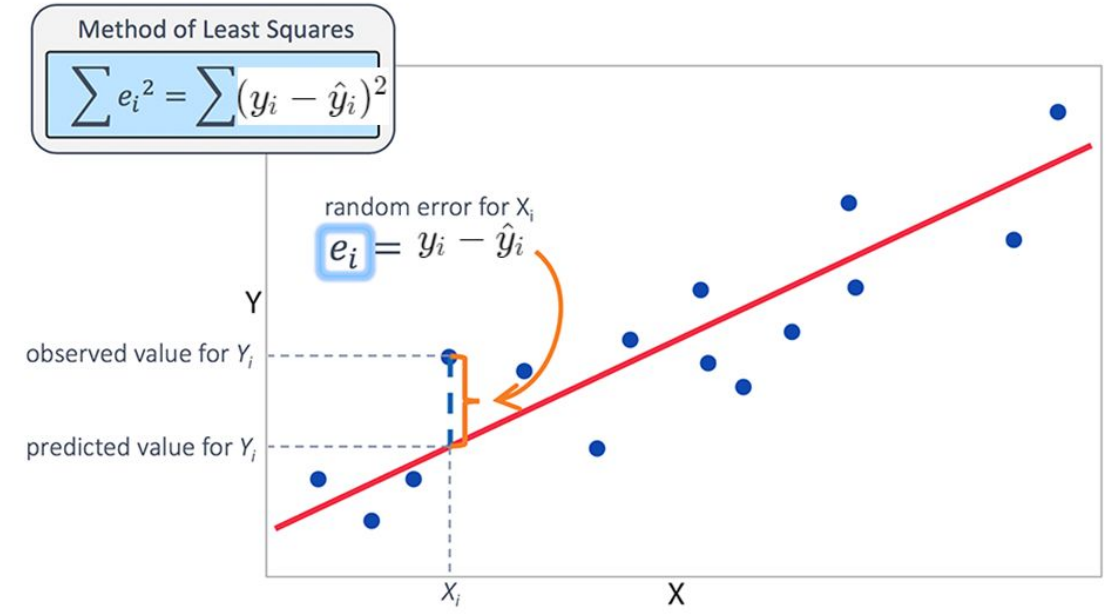


Loss Function

Training data:

$(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$

$y = f(x) = w_0 + w_1 x$



Error of 1 data point:

$$\frac{1}{2}(\hat{y}_i - y_i)^2$$

Residual to be minimized

MSE: Mean Square Error

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

Loss function

- **Loss Function**
  - **a mathematical function measuring how well the model's predictions match the actual data.**
  - **quantifies the error between the predicted values and the observed values in the training data.**

Cost Function

Training data:

$(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$

Linear regression model:

$$y = f(x) = w_0 + w_1 x$$

$$\text{Loss: MSE} = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Cost function

$$J = \frac{1}{2n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 = \frac{1}{2n}\sum_{i=1}^{n}((w_0 + w_1 x_i) - y_i)^2$$

Parameters: w_0, w_1

Goal: minimize cost function: $\min J$

$$\frac{\partial J}{\partial w_0} = 0, \quad \frac{\partial J}{\partial w_1} = 0$$

Linear Regression: Learning

- Goal: minimize cost function

$$\frac{\partial J}{\partial w_0} = 0, \qquad \frac{\partial J}{\partial w_1} = 0$$

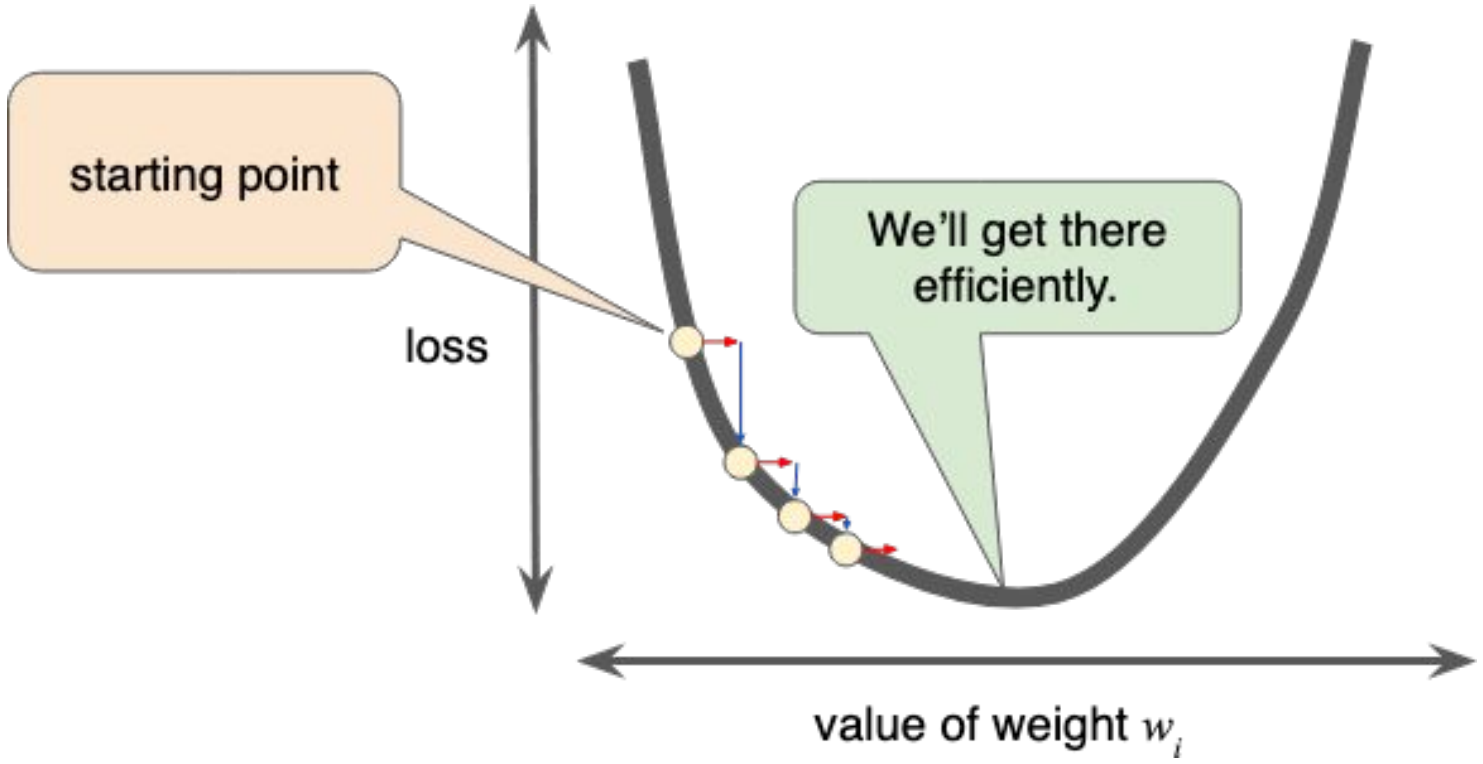
$$J = \frac{1}{2\mathrm{n}} \sum_{i=1}^{\mathrm{n}} (f(x_{i1}) - y_i)^2 =$$

$$= \frac{1}{2\mathrm{n}} \sum_{i=1}^{\mathrm{n}} ((w_0 + w_1 x_{i1}) - y_i)$$

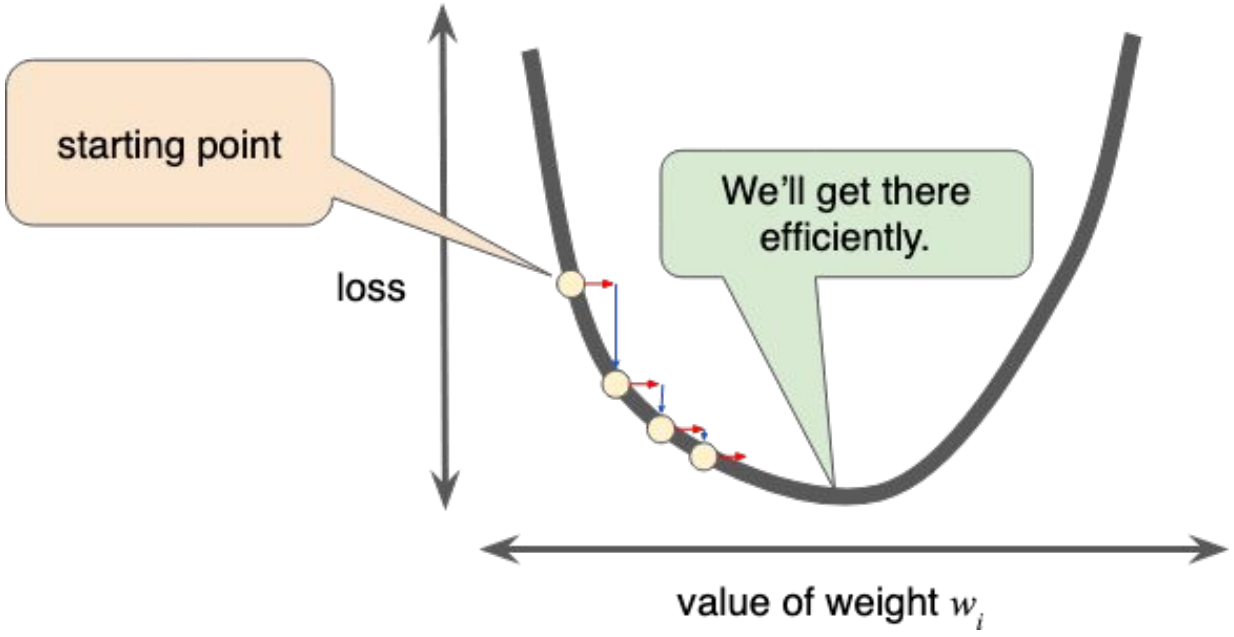
$$\frac{\partial J}{\partial w_0} = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} (f(x_{i1}) - y_i)$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} (f(x_{i1}) - y_i)\, x_{i1}$$

Linear Regression: Learning by Gradient Descent



Linear Regression: Learning by Gradient Descent



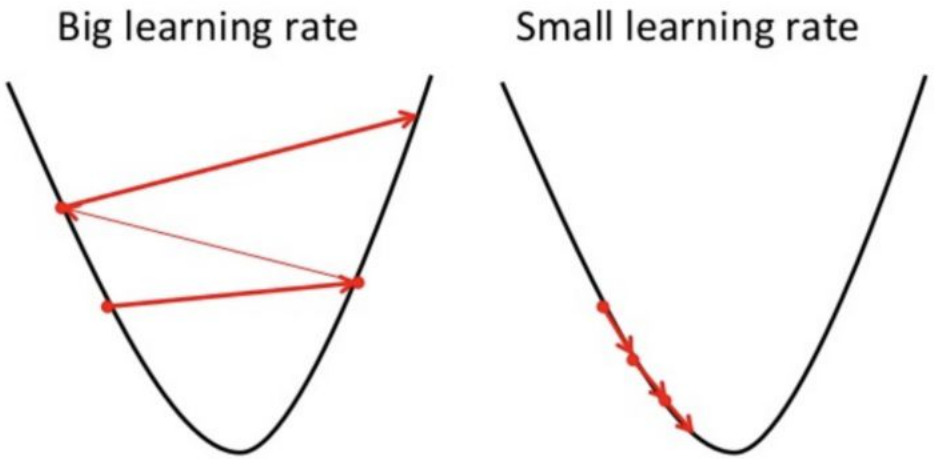
Update parameters:

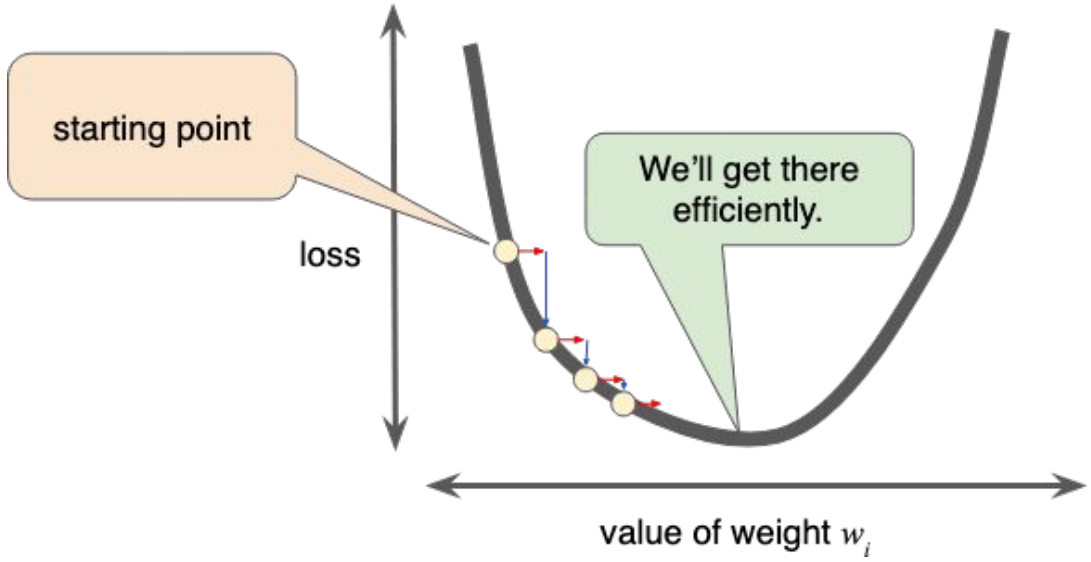
$$w_i = w_i - \mu \frac{\partial L}{\partial w_i}$$

Learning rate

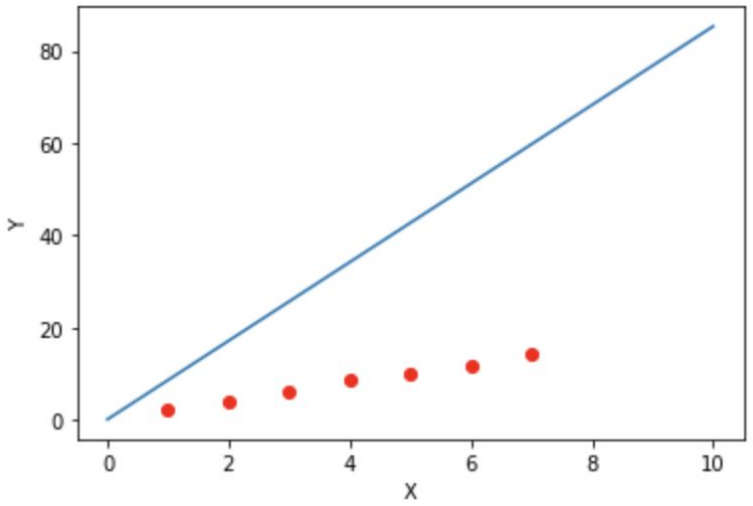
Gradient

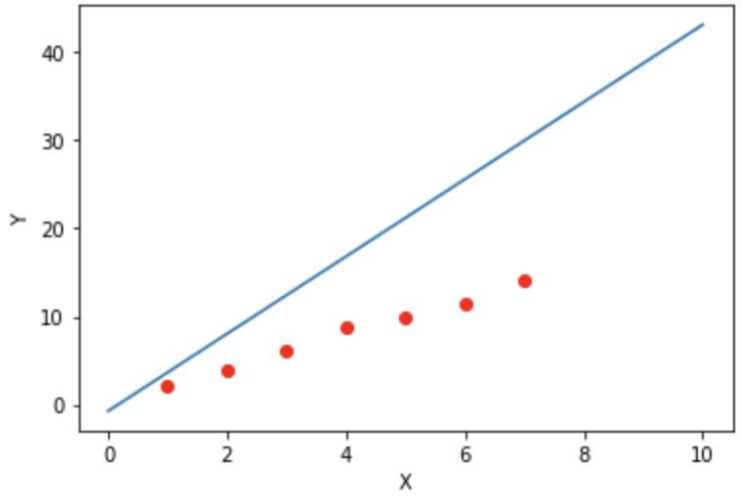
Linear Regression: Learning by Gradient Descent

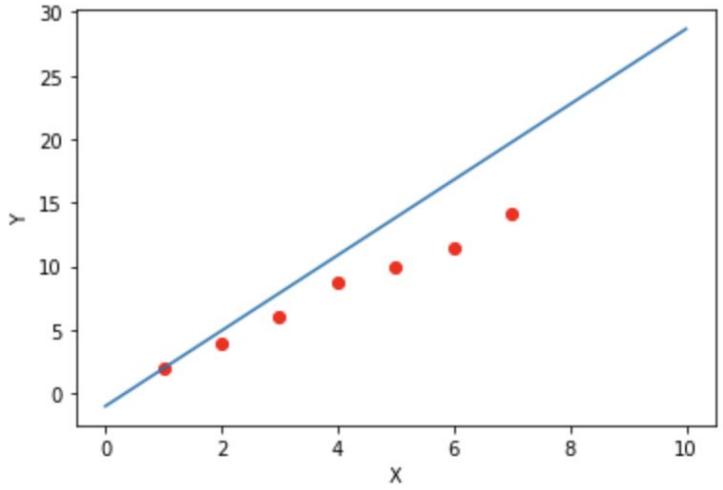


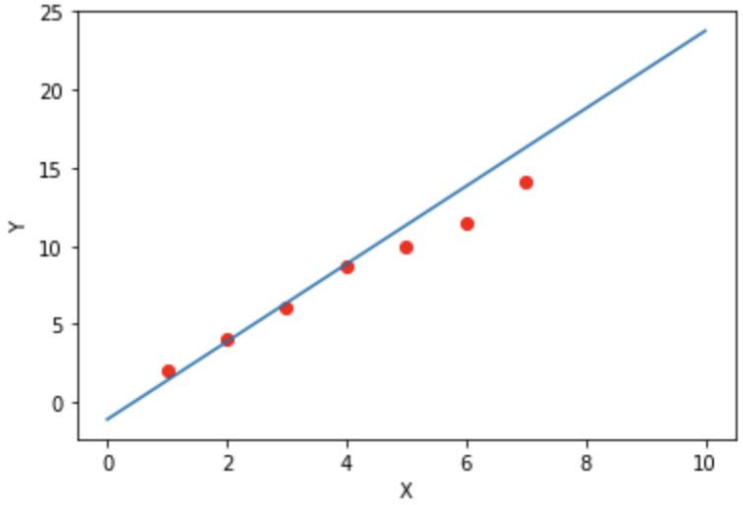


Learning progress









Linear Regression: Implementation

Steps Required in Gradient Descent Algorithm

- Step 1 we first initialize the parameters of the model randomly
- Step 2 Compute the gradient of the cost function with respect to each parameter. It involves making partial differentiation of cost function with respect to the parameters.
- Step 3 Update the parameters of the model by taking steps in the opposite direction of the model. Here we choose a hyperparameter learning rate. It helps in deciding the step size of the gradient.
- Step 4 Repeat steps 2 and 3 iteratively to get the best parameter for the defined model

Pseudocode for Gradient Descent

- $t \leftarrow 0$
- max_iterations $\leftarrow$ 1000
- w_1, $w_0 \leftarrow$ initialize randomly
- while $t <$ max_iterations do
  - $t \leftarrow t + 1$
  - $w_1^{(t+1)} \leftarrow w_1^{(t)} - \mu \nabla w_1^{(t)}$
  - $w_0^{(t+1)} \leftarrow w_0^{(t)} - \mu \nabla w_0^{(t)}$
- end

Note: max_iterations is the number of iteration we want to do to update our parameter

w_0 and w_1 are the bias and weight parameter

μ is the learning rate parameter

```
# Define function f to calculate y as a linear function of x
Function f(x, w):
    Return w[0] + w[1] * x
```

```
# Define the loss function to measure the mean squared error
Function loss(x, y, w):
    Initialize d to 0
    For each index i in x:
        Increment d by (y[i] - (w[0] + w[1] * x[i]))^2
    Return d / (2 * length of x)

# Define the derivative function to calculate the gradients with respect to w[0] and w[1]
Function derivative(x, y, w):
    Initialize d0 to 0
    Initialize d1 to 0
    For each index i in x:
        Increment d1 by x[i] * (f(x[i], w) - y[i])
        Increment d0 by f(x[i], w) - y[i]
    Return (d0 / length of x, d1 / length of x)
```

d0: the gradient with respect to w_0

d1: the gradient with respect to w_1

$$d_0 = \frac{\partial J}{\partial w_0} = \frac{1}{n}\sum_{i=1}^{n}\left(f(x^{(i)}, \mathbf{w}) - y^{(i)}\right)$$

$$d_1 = \frac{\partial J}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n}\left(f(x^{(i)}, \mathbf{w}) - y^{(i)}\right)x^{(i)}$$

```
# Training process with a specified number of epochs and learning rate
Set epoch to 10
Set learning_rate to 0.01
Initialize w to [1, 1] # Start with y = x + 1
Initialize los_old to 0

For each iteration i from 0 to epoch:
    # Display the graph
    Plot points (x, y) on the graph with red 'o' markers
    Label x-axis as 'X'
    Label y-axis as 'Y'
    Generate 50 points for x0 evenly spaced from -1 to 10
    Calculate y0 as w[0] + w[1] * x0
    Plot the line (x0, y0)
    Show the plot

    # Update the parameters
    Calculate the current loss
    Print the current epoch and loss
    If the current loss is greater than the previous loss minus a small threshold and i is greater than 0:
        Break the loop
    Update los_old to the current loss

    # Calculate the gradient
    Calculate the derivatives a and b with respect to w[0] and w[1]
    Update w[0] by subtracting a * learning_rate
    Update w[1] by subtracting b * learning_rate

Return the final parameters w
```

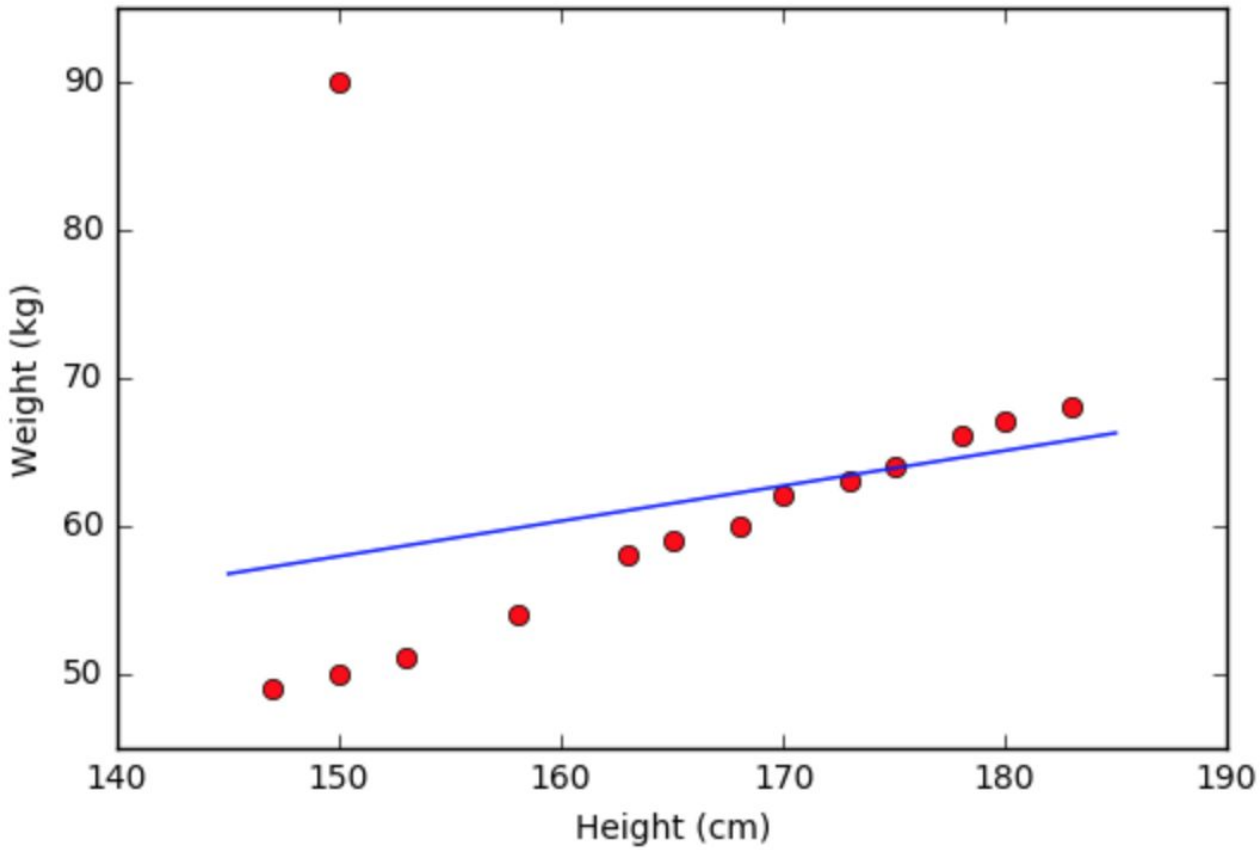
d0 and d1

- Update w_0: $w_0 := w_0 - \mu \cdot d_0$
- Update w_1: $w_1 := w_1 - \mu \cdot d_1$

```python
# hàm y = f(x)
# w: tham số
def f(x,w):
    return w[0]+w[1]*x

# tính hàm loss
def loss(x,y,w):
    d = 0
    for i in range(len(x)):
        d += (y[i] - (w[0] + w[1]*x[i]))
    return d/(2*len(x))

# tính đạo hàm tại điểm w[0] và w[1]
def derivative(x,y,w):
    d0=0
    d1=0
    for i in range(len(x)):
        d1 += x[i]*(f(x[i],w)-y[i])
        d0 += f(x[i],w)-y[i]
    return d0/len(x),d1/len(x)
```

```python
# Assuming x and y are defined as numpy arrays of the same length
x = np.linspace(start=1, stop=10, num=50)
y = 2 * x + np.random.normal(0, 1, 50)  # Example linear data with some noise

epoch = 10
learning_rate = 0.01
w = [1, 1]  # Initial guess for the model parameters
los_old = float('inf')  # Initialize as infinity

for i in range(epoch):
    # Plot the data points
    plt.plot(x, y, 'ro')
    plt.xlabel('X')
    plt.ylabel('Y')

    # Plot the current model
    x0 = np.linspace(start=1, stop=10, num=50)
    y0 = w[0] + w[1] * x0
    plt.plot(x0, y0)
    plt.show()

    # Update parameters
    los = loss(x, y, w)
    print("epoch_", i, ':')
    print(los, " : ", los_old)

    # If loss has not decreased significantly, stop the training
    if los > (los_old - 0.0001) and i > 0:
        break
    los_old = los

    # Calculate the gradient and update the weights
    a, b = derivative(x, y, w)
    w[0] = w[0] - a * learning_rate
    w[1] = w[1] - b * learning_rate
```

Issues:
Linear Regression is sensitive with outliers



Multiple Linear Regression

(X_i, y_i)

$X_i = (x_{i1}, \ldots, x_k)$

$$y_i = f(X_i) = w_0 x_{i0} + w_1 x_{i1} + \ldots + w_k x_{ik}$$

$$\text{Loss} = L = \frac{1}{2N}\sum_{i=1}^{N}(f(X) - y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}((w_0 + w_1 x_{i1} + \cdots + w_k x_{ik}) - y_i)^2 =$$

$$\frac{\partial L}{\partial w_t} = \frac{1}{N}\sum_{i=1}^{N}(f(X_i) - y_i)\, x_{it}$$

Summary

- Supervised learning vs unsupervised learning.
- Classification vs regression
- Linear regression form.
- Learning Linear Regression model by Gradient Descent algorithm?
- One variable and Multiple variable of Linear Regression

Answer the questions:

1. What is the objective of a supervised learning model?
2. What is linear regression model?
3. Formulating the linear regression model with single variable?
   - Function y = f(x)
   - Loss function
4. Learning parameters by Gradient Descent
   - Update parameters?
   - Learning rate?
5. Implementation

Exercise

- Implement Linear Regression for the problem of house pricing with multiple variables:
  - Firstly derive steps and mathematical formulas in the algorithm.
  - Secondly, implementation using python.

Mathematical Formulas

1. **Hypothesis Function**:
   - The hypothesis function for multiple linear regression is represented as:
     $f_w(x) = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$
   - Where $f_w(x)$ is the predicted house price, $w_0, w_1, ..., w_n$ are the parameters (coefficients), and $x_1, x_2, ..., x_n$ are the input features.
2. **Cost Function (Mean Squared Error)**:
   - The cost function measures the difference between the predicted and actual values. For multiple linear regression, it is represented as:
     $J(w) = \frac{1}{2m} \sum_{i=1}^{m} (f_w(x^{(i)}) - y^{(i)})^2$
   - Where m is the number of training examples, $f_w(x^{(i)})$ is the predicted value for the i th example, and $y^{(i)}$ is the actual value.
3. **Gradient Descent**:
   - Update rule for gradient descent:
     $w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$
   - Where α is the learning rate and $\frac{\partial}{\partial w_j} J(w)$ is the partial derivative of the cost function with respect to w_j.

$$f(X) = w_0.x_0 + w_1.x_1 + ... + w_d.x_d$$

$$W = [w_0, w_1, ..., w_d]^T$$

$$X = [x_0, x_1, ..., x_d]$$

$$y = X.W$$

$$Loss = \frac{1}{2}\sum_{i=1}^{n}(f(X_i) - y_i)^2 = \frac{1}{2}\|X.W - y\|^2$$

$$\frac{\partial Loss(W)}{\partial W} = X^T(X.W - y) = 0$$

$$=> X^T.X.W = X^T.y$$

$$X^T.W \triangleq b$$

$$X^T.X.W = b$$

$$A.W = X^T.b$$

$$A^{-1}A.W = A^{-1}.b \; W = A^{-1}.b$$