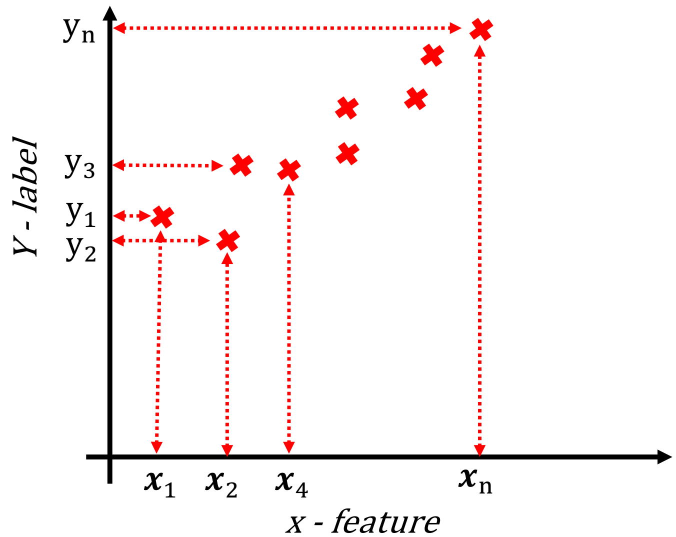
Simple Linear Regression

In simple linear regression we will be given a dataset of pair of values

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_{i}	x_{n}
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_{i}	y_{n}

x: feature (independent variable)
y: label (dependent variable)

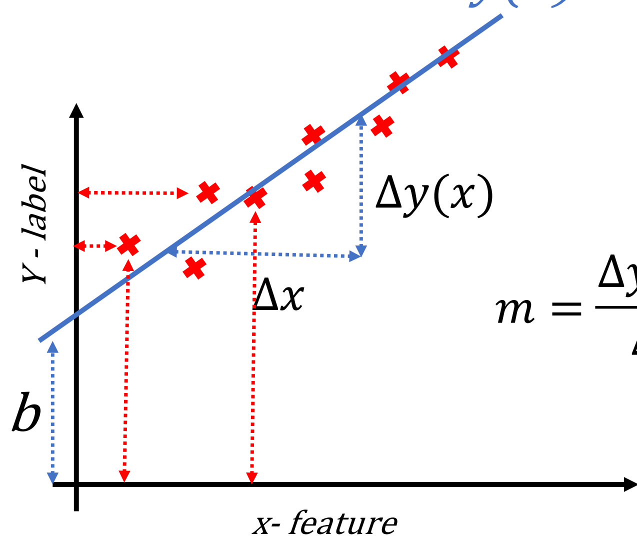


Simple Linear Regression

In simple linear regression we aim to derive a linear equation that best fit the given dataset

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_{i}	x_{n}
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_{i}	y_{n}

$$y(x) = mx + b$$



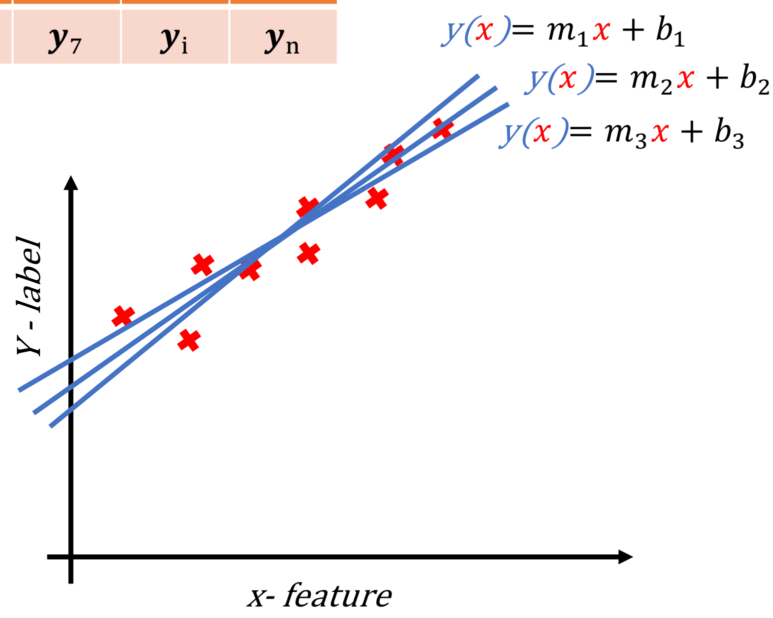
$$m = \frac{\Delta y(x)}{\Delta x}$$

x: feature (independent variable)
y: label (dependent variable)
m: slope of coefficient
b: y intercept or bias

Simple Linear Regression

Question: what are the best values for **m** and **b**?

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_i	x_n
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_i	y_n



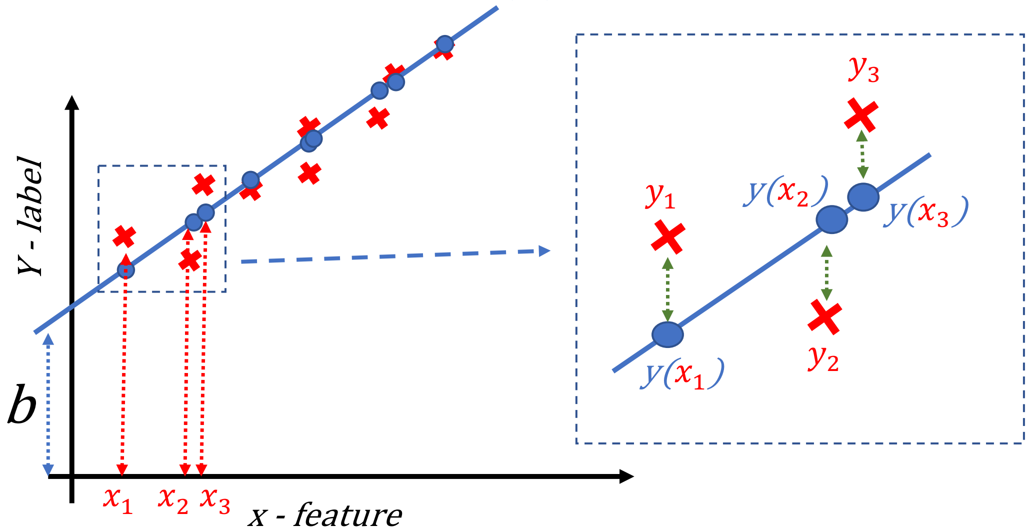
Simple Linear Regression

Question: what are the best values for **m** and **b**?

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_i	x_n
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_i	y_n

$$y(x) = mx + b$$

Answer: The values that reduce the error between measured and calculated labels (dependent variable)



Mean Square Error (MSE)

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

$$J = \frac{1}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^2$$

Simple Linear Regression

Mean Square Error (MSE)

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i) - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^2$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i) = 0$$

$$\frac{\partial J}{\partial m} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i)(x_i) = 0$$

$$m\sum_{i=1}^{n}(x_i) + nb = \sum_{i=1}^{n}(y_i)$$

$$m\sum_{i=1}^{n}(x_i)^2 + b\sum_{i=1}^{n}(x_i) = \sum_{i=1}^{n}(x_i y_i)$$

$$\bar{x} = \sum_{i=1}^{n}(x_i) \qquad \bar{y} = \sum_{i=1}^{n}(y_i)$$

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

Simple Linear Regression

In simple linear regression we aim to derive a linear equation that best fit the given dataset

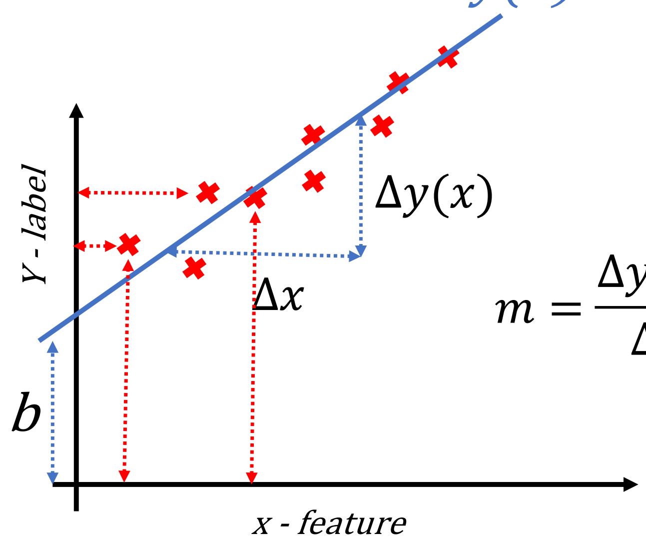
x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_{i}	x_{n}
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_{i}	y_{n}

$$y(x) = mx + b$$

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

$$\bar{x} = \sum_{i=1}^{n}(x_i) \qquad \bar{y} = \sum_{i=1}^{n}(y_i)$$

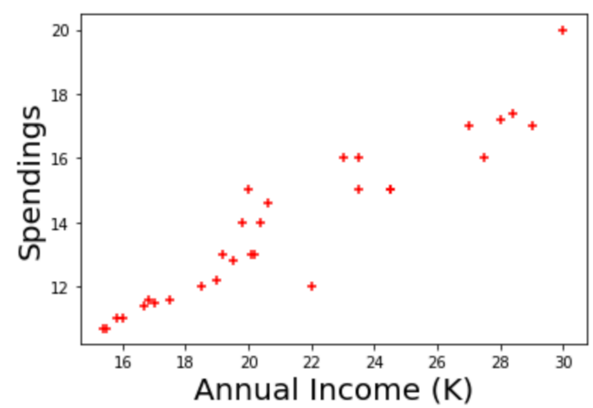


$$m = \frac{\Delta y(x)}{\Delta x}$$

Simple Linear Regression

Example: let us have a dataset of a group of mall customers which contains information about customers' annual income and their spending. It is required to design a linear regression model (inference model) that can predict amount of spending for a given the customer income.

	Annual Income (K)	Spendings
0	15.5	10.7
1	15.4	10.7
2	15.8	11.0
3	16.0	11.0
4	16.7	11.4



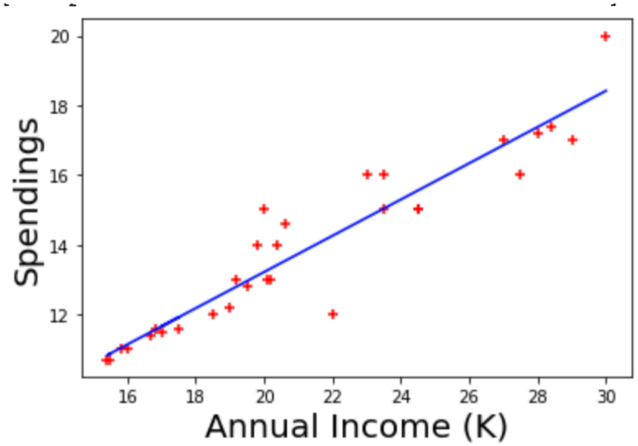
Simple Linear Regression

Example: let us have a dataset of a group of mall customers which contains information about customers annual income and their spending. It is required to design an inference model (linear regression model (inference model) that can predict amount of spending if given the customer income.

$$y(x) = mx + b$$

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(df[['Annual Income (K)']],df[['Spendings']])
print(reg.coef_) ## print the coefficient
print(reg.intercept_) ## print the intercept
```

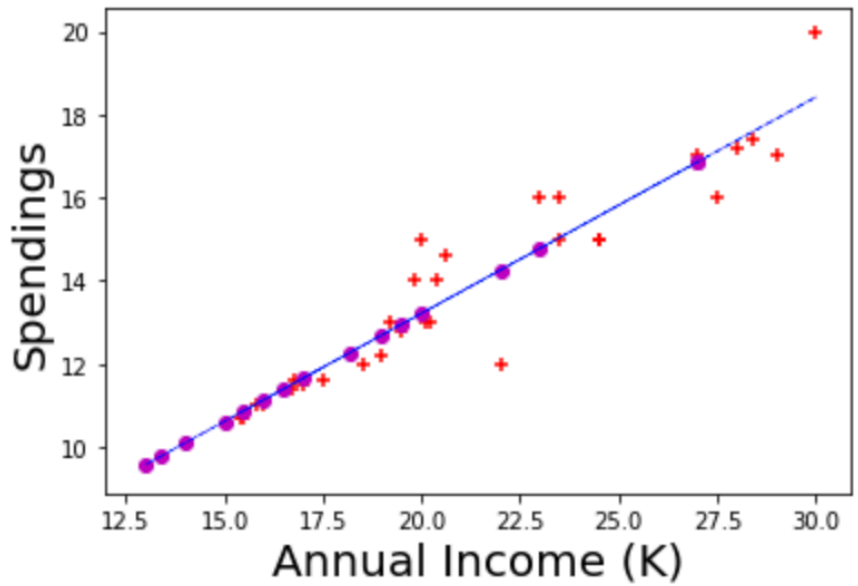
```
[[0.52004105]] [2.81485916]
```



Simple Linear Regression

Example: let us have a dataset of a group of mall customers which contains information about customers annual income and their spending. It is required to design an inference model (linear regression model (inference model) that can predict amount of spending if given the customer income.

$$y(x) = mx + b$$

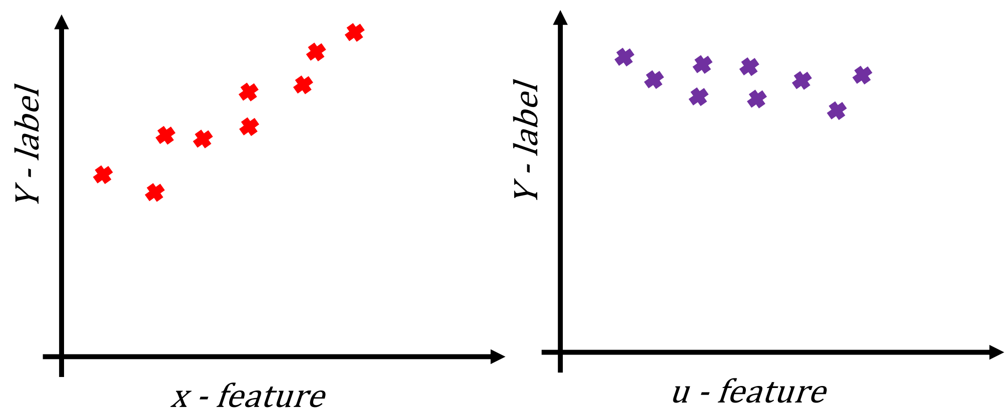


Multiple Regression

In Multiple Regression, the features vector includes two or more features

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_i	x_n
u_i	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_i	u_n
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_i	y_n

x,u: features (independent variables)
y: label (dependent variable)
m_x, m_u: slope of coefficient
b: y intercept or bias



$$y(x,u) = m_x x + m_u u + b$$

Multiple Regression

Mean Square Error (MSE)

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i, u_i) - y_i)^2 \ = \frac{1}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i)^2$$

$$\downarrow$$

$$\frac{\partial J}{\partial b} = \ \frac{2}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i) = 0$$

$$\downarrow$$

$$m_x \sum_{i=1}^{n}(x_i) + m_u \sum_{i=1}^{n}(u_i) + nb = \sum_{i=1}^{n}(y_i)$$

Multiple Regression

Mean Square Error (MSE)

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i, u_i) - y_i)^2 \; = \frac{1}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i)^2$$

$$\downarrow$$

$$\frac{\partial J}{\partial m_x} = \frac{2}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i)(x_i) = 0$$

$$\downarrow$$

$$m_x \sum_{i=1}^{n}(x_i)^2 + m_u \sum_{i=1}^{n}(x_i u_i) + b \sum_{i=1}^{n}(x_i) = \sum_{i=1}^{n}(x_i y_i)$$

Multiple Regression

Mean Square Error (MSE)

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i, u_i) - y_i)^2 \;\; = \frac{1}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i)^2$$

$$\frac{\partial J}{\partial m_u} = \frac{2}{n}\sum_{i=1}^{n}(m_x x_i + m_u u_i + b - y_i)(u_i)=0$$

$$m_x \sum_{i=1}^{n}(x_i u_i) + m_u \sum_{i=1}^{n}(u_i)^2 + b \sum_{i=1}^{n}(u_i)= \sum_{i=1}^{n}(u_i y_i)$$

Multiple Regression

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i, u_i) - y_i)^2$$

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_i	x_n
u_i	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_i	u_n
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_i	y_n

$$m_x \sum_{i=1}^{n}(x_i) + m_u \sum_{i=1}^{n}(u_i) + nb = \sum_{i=1}^{n}(y_i)$$

$$m_x \sum_{i=1}^{n}(x_i)^2 + m_u \sum_{i=1}^{n}(x_i u_i) + b \sum_{i=1}^{n}(x_i) = \sum_{i=1}^{n}(x_i y_i)$$

$$m_x \sum_{i=1}^{n}(x_i u_i) + m_u \sum_{i=1}^{n}(u_i)^2 + b \sum_{i=1}^{n}(u_i) = \sum_{i=1}^{n}(u_i y_i)$$

Need to solve three equations for an interception (b) and two coefficients (m_x and m_u)

Multiple Regression

Example: let us have a dataset of a group of mall customers which contains information about families customers annual income, number of workers in each family, number of kids in each family, and the family spending. It is required to design linear regression model (inference model) that can predict amount of spending based on family information (incoming, members working, kids).

	Annual Income	Working	Kids	Spendings
0	15000	1.0	0	11300
1	28000	NaN	0	25000
2	16000	1.0	0	11800
3	17000	1.0	0	12400
4	28000	2.0	0	21400

$$y(x, u, v) = m_x x + m_u u + m_v v + b$$

Multiple Regression

Example: let us have a dataset of a group of mall customers which contains information about families customers annual income, number of workers in each family, number of kids in each family, and the family spending. It is required to design linear regression model (inference model) that can predict amount of spending based on family information (incoming, members working, kids).

```
from sklearn import linear_model
regm = linear_model.LinearRegression()
regm.fit(df[['Annual Income','Working','Kids']],df.Spendings)
print(regm.coef_) ## print the coefficients
print(regm.intercept_) ## print the intercept
```

$$y(x,u,v) = m_x x + m_u u + m_v v + b$$

```
[6.16157272e-01 2.26840512e+03 1.71714466e+03]
64.65754063569693
```

Gradient Descent

As the number of features increases, we will have more equations of many independent random variables to solve. Thus, we need to use gradient descent to determine the interception and coefficients

Let us consider the simple linear regression case

$$MSE = J = \frac{1}{n}\sum_{i=1}^{n}(y(x_i) - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(mx_i + b - y_i)^2$$

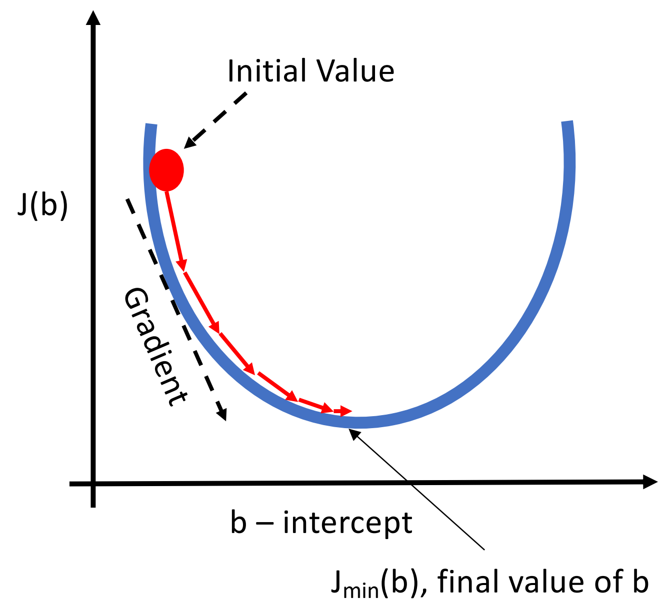
$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i)$$

$$\frac{\partial J}{\partial m} = \frac{2}{n}\sum_{i=1}^{n}(mx_i + b - y_i)(x_i)$$

$$b_j = b_{j-1} + \lambda\frac{\partial J}{\partial b}$$

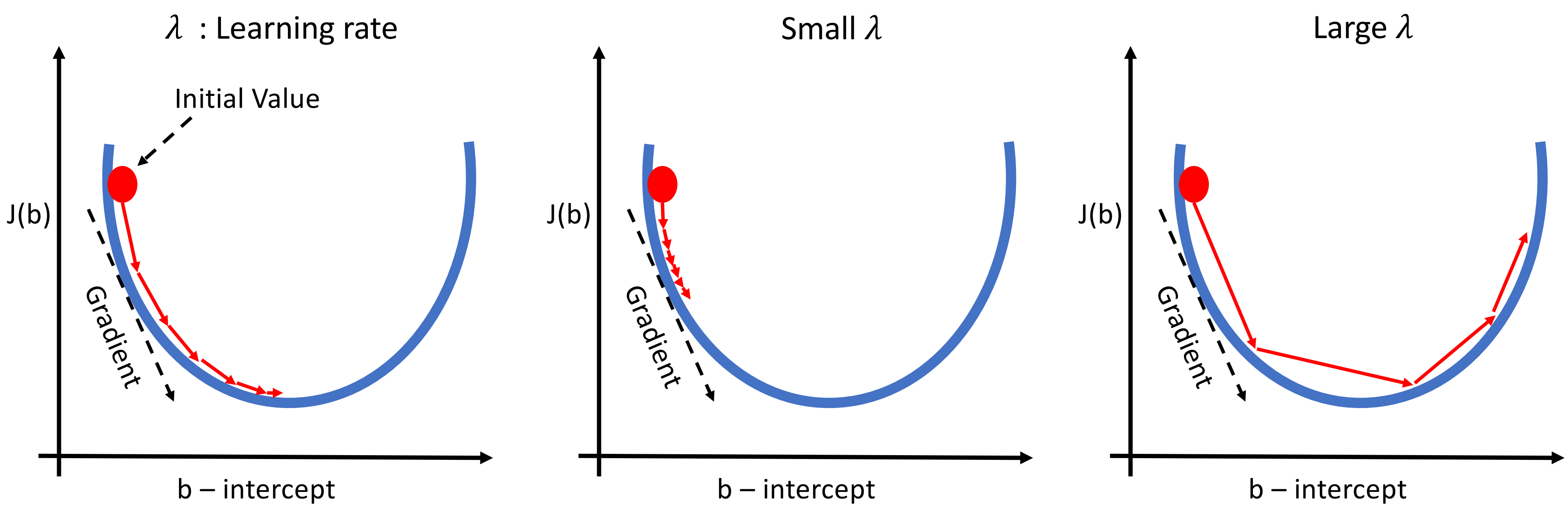
$$m_j = m_{j-1} + \lambda\frac{\partial J}{\partial m}$$

λ : Learning rate



Gradient Descent

Selecting an appropriate learning rate (λ) is important to determine successfully the best intercept and coefficient values.



Gradient Descent

Sample Gradient Decent Code

```
import numpy as np
import copy
def gradient_descent(x,y,m_curr,b_curr,learning_rate,epochs):
    n = len(x)
    i = 0
    j_curr = 100000
    while True:
        i= i + 1
        j_before = j_curr
        y_pred = m_curr * x + b_curr

        md = - ( 2 / n ) * sum( x * ( y - y_pred ))
        bd = - ( 2 / n ) * sum( y - y_pred )

        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd

        j_curr = ( 1 / n ) * sum(( y - y_pred )**2)

        if ((abs(j_curr - j_before) < 1e-5) or (i >= epochs)):
          return m_curr,b_curr,i,j_curr
```

Saving and Loading Training Models

There are different ways methods to save and load training models. In this tutorial we will presend two methods:

- Pickle library (https://docs.python.org/3/library/pickle.html).
- Joblib from sklearn library (https://scikit-learn.org/stable/modules/model_persistence.html)

To save model "reg" using pickle:

```
import pickle
with open('./SpendingsLinearModel.pickle','wb') as f:
  pickle.dump(reg,f)
```

To load model "reg" using pickle:

```
import pickle
with open('./SpendingsLinearModel.pickle','rb') as f:
  reg_pickle = pickle.load(f)
```

Saving and Loading Training Models

There are different ways methods to save and load training models. In this tutorial we will be using two methods:

- Pickle library (https://docs.python.org/3/library/pickle.html).
- Joblib from sklearn library (https://scikit-learn.org/stable/modules/model_persistence.html)

To save model “reg” using joblib :

```
import joblib as jb
jb.dump(reg, './SpendingsLinearModel.joblib')
```

To load model “reg” using joblib :

```
import joblib as jb
reg_joblib = jb.load('./SpendingsLinearModel.joblib')
```