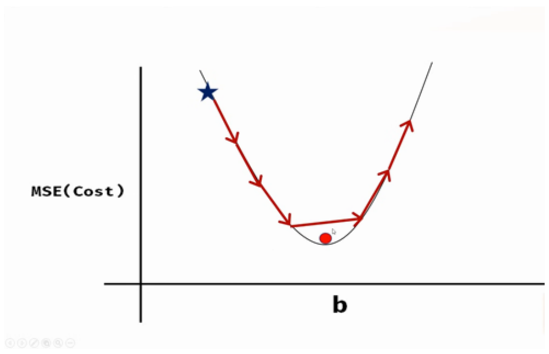
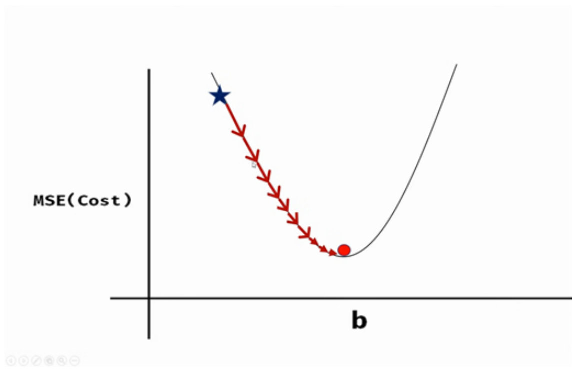
Unit08 Ex4 gradient_descent_cost_function

July 27, 2023

0.1 Calculating cost with gradient descent and learning rate

- Change the iteration and learning rate vaules and see the impact on cost.
- Low iteration values with high learning rate (i.e. big steps) may lead to miss the global minimum
- Goal is to reach minimum cost with minimum iteration





[ ]:
```
# code credit:codebasics https://codebasics.io/coming-soon

import numpy as np

def gradient_descent(x,y):
    m_curr = b_curr = 0
    iterations = 100        #change value
    n = len(x)
    learning_rate = 0.08   #change value

    for i in range(iterations):
        y_predicted = m_curr * x + b_curr
        cost = (1/n) * sum([val**2 for val in (y-y_predicted)])
        md = -(2/n)*sum(x*(y-y_predicted))
        bd = -(2/n)*sum(y-y_predicted)
        m_curr = m_curr - learning_rate * md
        b_curr = b_curr - learning_rate * bd
        print ("m {}, b {}, cost {} iteration {}".format(m_curr,b_curr,cost, i))

x = np.array([1,2,3,4,5])
```

```
y = np.array([5,7,9,11,13])

gradient_descent(x,y)
```

[ ]:

```
```