GRADIENT DESCENT

- **Optimization** is a big part of **machine learning**. Almost Every machine learning algorithm has an optimization algoritms.
- Gradient Descent is best use when the **parametrs cannot be calculated analytically** or when you need an optimized way to calculate those parameters.
- It is used to **find the values of parameters (coffecient) of a function that minimize the cost function.**
- It is an **iterative algorithm** use in **loss function to find global minima** and gradient descent algorithm is most **popular in machine learning and deep learning**

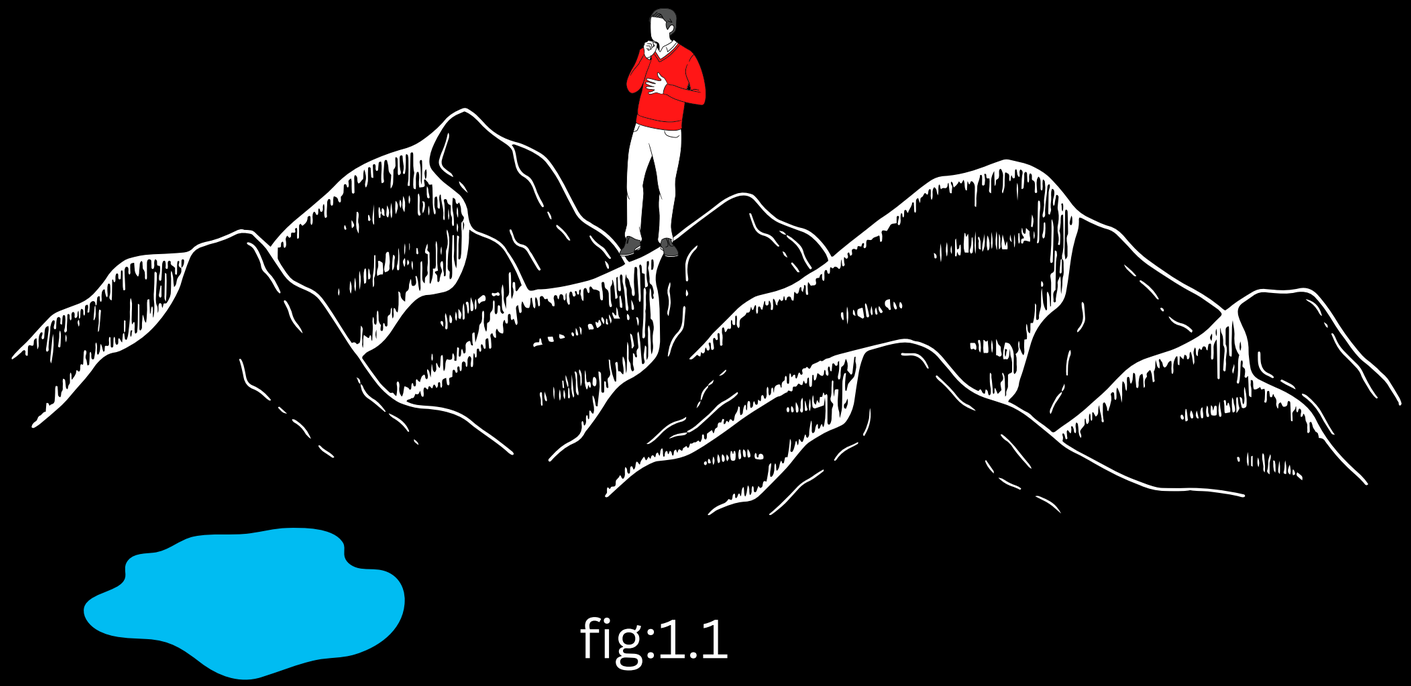

fig:1.1

Above figure let's assume you are at **top of the mountain** and you **eyes was tighed** by cloth your **aim is you need to reach down towards the water**. Generally what we do we try to **step any one side if the side is slope we'll go in that direction else we change**. like gradient also use **same concept to find the global minima gradient descent ultimate goal is finding the gloabal minima** here we called **step** but in **gradient descent algorithm we called as learining rate**.

NOTE: THIS EXAMPLE THAT EXPLAINED WHAT I UNDERSTOOD

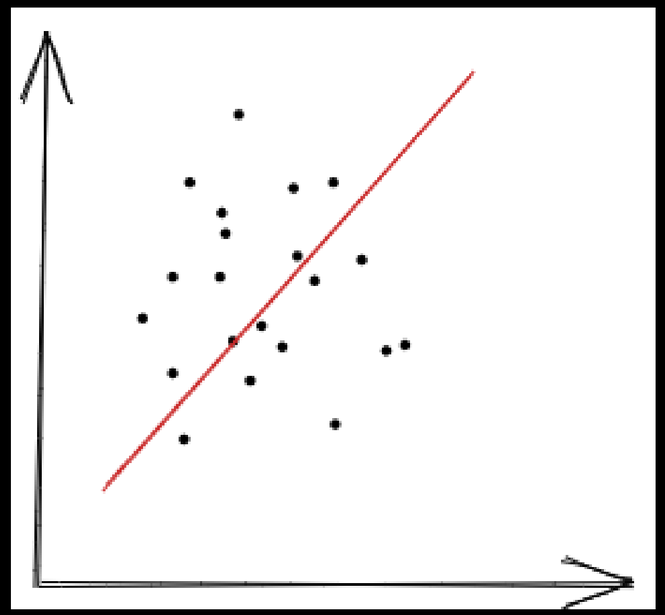

fig1.2

In fig1.2 here the fit line is simple to **find minimum cost using the mx+c** is the general equation of any straight line where m is the gradient of the line (how steep the line is) and c is the y -intercept (the point in which the line crosses the y -axis)

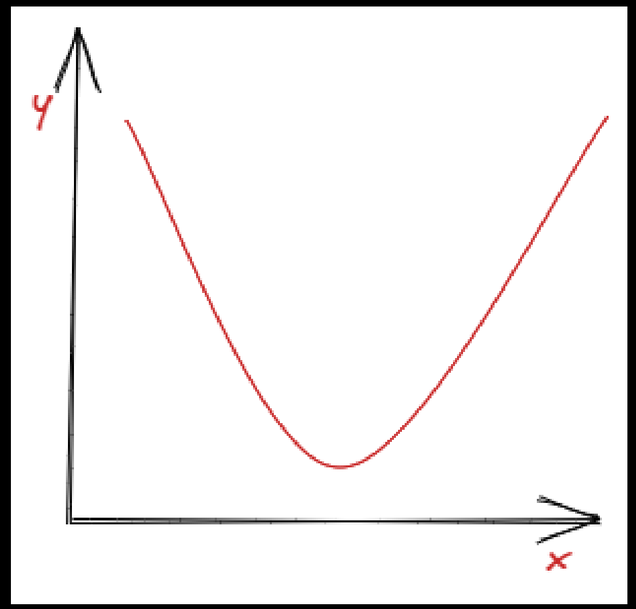

fig1.3

In fig1.2 if fit line is like this it is **very hard to find the minimum cost** to solve this we use **Gradient descent is an optimization algorithm** Training data helps these models learn over time, and the cost function within gradient descent specifically **acts as a barometer, gauging its accuracy with each iteration of parameter updates.** Until the function is **close to or equal to zero**, the model will continue to adjust its parameters to yield the smallest possible error.

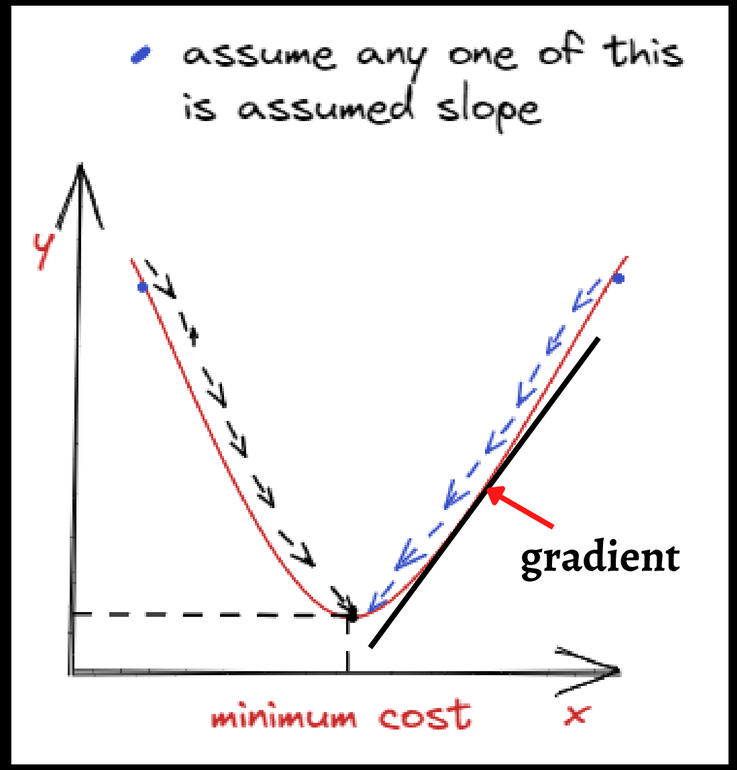

fig:1.4

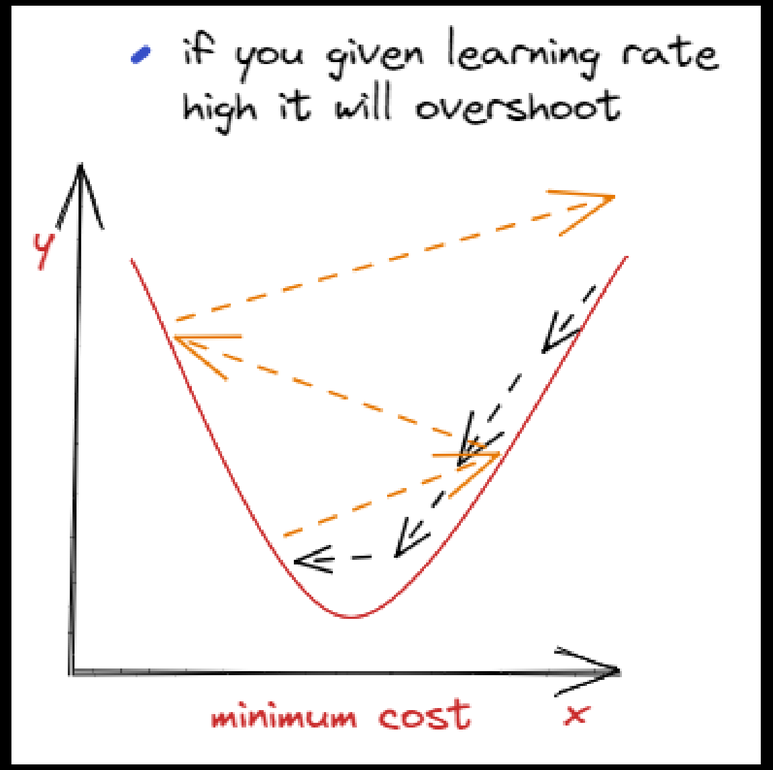

fig:1.5

In above figures you can find the **gradient line** first we **initial weight randomly** after that check for **slope** and taking the **small steps(learning rate)** towards the global minima, taking the large steps leads to over-shooting in fig:1.5 here **"m"** and **"c"** will **update** using **gradient descent** to find the best fit line best line nothing but minima or loss function.

Gradient Descent for One Varible

In [1]:
```
import numpy as np
```

In [2]:
```
data = np.loadtxt("data.csv", delimiter=",")
data.shape
```

Out[2]:
```
(100, 2)
```

In [3]:
```
def step_gradient(points, learning_rate, m , c):
    m_slope = 0
    c_slope = 0
    M = len(points)
    for i in range(M):
        x = points[i, 0]
        y = points[i, 1]
        m_slope += (-2/M)* (y - m * x - c)*x
        c_slope += (-2/M)* (y - m * x - c)
    new_m = m - learning_rate*m_slope
    new_c = c - learning_rate*c_slope
    return new_m, new_c
```

In [4]:
```
def gd(points, learning_rate, num_iterations):
    m = 0
    c = 0
    for i in range(num_iterations):
        m, c = step_gradient(points, learning_rate, m , c)
        print(i, " Cost: ", cost(points, m, c))
    return m, c
```

In [7]:
```
def cost(points, m, c):
    total_cost = 0
    M = len(points)
    for i in range(M):
        x = points[i, 0]
        y = points[i, 1]
        total_cost += (1/M)*((y - m*x - c)**2)
    return total_cost
```

In [14]:
```
def run(n):
    data = np.loadtxt("data.csv", delimiter=",")
    learning_rate = n
    num_iterations = 100 # you can give how many iteration you want
    m, c = gd(data, learning_rate, num_iterations)
    print(m, c)
```

In [11]:
```
# give learning rate 0.1 to 1.0
run(0.001)
```

```
0  Cost:  86648.79500400844
1  Cost:  1373530.9288716826
2  Cost:  21797683.77391045
3  Cost:  345950151.47582006
4  Cost:  5490585677.50353
5  Cost:  87141281978.78787
6  Cost:  1383022432889.3513
7  Cost:  21949998997164.734
8  Cost:  348369227101707.4
9  Cost:  5528980589368769.0
10  Cost:  8.775065068735354e+16
11  Cost:  1.3926937473537413e+18
12  Cost:  2.2103492780113805e+19
13  Cost:  3.5080533247805876e+20
14  Cost:  5.567644105811217e+21
15  Cost:  8.836428075366614e+22
16  Cost:  1.402432692305693e+24
17  Cost:  2.225805992729919e+25
18  Cost:  3.5325847325530905e+26
19  Cost:  5.60657799171512e+27
20  Cost:  8.898220186346784e+28
21  Cost:  1.412239740563884e+30
22  Cost:  2.2413707944518334e+31
23  Cost:  3.5572876855991514e+32
24  Cost:  5.64578413774247e+33
25  Cost:  8.960444402352507e+34
26  Cost:  1.4221153683667963e+36
27  Cost:  2.257044439016949e+37
28  Cost:  3.582163383514893e+38
29  Cost:  5.685264447776564e+39
30  Cost:  9.023103745043844e+40
31  Cost:  1.432060055283185e+42
32  Cost:  2.272827687550557e+43
33  Cost:  3.607213034285011e+44
34  Cost:  5.725020839014316e+45
35  Cost:  9.08620125721096e+46
36  Cost:  1.4420742842353809e+48
37  Cost:  2.2887213065004435e+49
38  Cost:  3.6324378543415493e+50
39  Cost:  5.7650552420594125e+51
40  Cost:  9.149740002921913e+52
41  Cost:  1.4521585415227255e+54
42  Cost:  2.304726067674044e+55
43  Cost:  3.6578390686228904e+56
44  Cost:  5.805369601015973e+57
45  Cost:  9.213723067671385e+58
46  Cost:  1.4623133168452018e+60
47  Cost:  2.3208427482759715e+61
48  Cost:  3.6834179106332717e+62
49  Cost:  5.84596587358303e+63
50  Cost:  9.278153558530604e+64
51  Cost:  1.4725391033272058e+66
52  Cost:  2.3370721309457386e+67
53  Cost:  3.7091756225027007e+68
54  Cost:  5.886846031149612e+69
55  Cost:  9.343034604298187e+70
56  Cost:  1.4828363975414934e+72
57  Cost:  2.353415003795759e+73
58  Cost:  3.735113455047232e+74
59  Cost:  5.9280120588904e+75
60  Cost:  9.408369355652045e+76
```

```
61  Cost:  1.4932056995332964e+78
62  Cost:  2.3698721604496293e+79
63  Cost:  3.761232667829744e+80
64  Cost:  5.969465955862183e+81
65  Cost:  9.474160985302457e+82
66  Cost:  1.503647512844606e+84
67  Cost:  2.386444400080657e+85
68  Cost:  3.787534529221072e+86
69  Cost:  6.011209735100916e+87
70  Cost:  9.540412688146056e+88
71  Cost:  1.514162344538625e+90
72  Cost:  2.4031325274506905e+91
73  Cost:  3.8140203164616646e+92
74  Cost:  6.053245423719487e+93
75  Cost:  9.607127681420982e+94
76  Cost:  1.5247507052243778e+96
77  Cost:  2.419937352949151e+97
78  Cost:  3.840691315723497e+98
79  Cost:  6.095575063006123e+99
80  Cost:  9.674309204863226e+100
81  Cost:  1.5354131090815417e+102
82  Cost:  2.4368596926324663e+103
83  Cost:  3.867548822172671e+104
84  Cost:  6.138200708523602e+105
85  Cost:  9.741960520863845e+106
86  Cost:  1.5461500738853693e+108
87  Cost:  2.4539003682636117e+109
88  Cost:  3.894594140032174e+110
89  Cost:  6.181124430208961e+111
90  Cost:  9.81008491462743e+112
91  Cost:  1.5569621210318738e+114
92  Cost:  2.471060207352074e+115
93  Cost:  3.921828582645306e+116
94  Cost:  6.224348312474069e+117
95  Cost:  9.878685694331551e+118
96  Cost:  1.567849775563111e+120
97  Cost:  2.488340043194009e+121
98  Cost:  3.949253472539409e+122
99  Cost:  6.267874454306871e+123
-1.585655822608496e+60 -3.116558206251511e+58
```

In [13]:
```
run(0.0001) # here you can see the big difference it optimizing the cost function
```

```
0  Cost:  1484.5865574086486
1  Cost:  457.8542575737672
2  Cost:  199.5099857255389
3  Cost:  134.50591058200533
4  Cost:  118.1496934223995
5  Cost:  114.0341490603815
6  Cost:  112.99857731713657
7  Cost:  112.73798187568467
8  Cost:  112.6723843590911
9  Cost:  112.65585181499745
10  Cost:  112.65166489759581
11  Cost:  112.6505843615011
12  Cost:  112.65028544701502
13  Cost:  112.65018320293967
14  Cost:  112.650130445072
15  Cost:  112.65009013922885
16  Cost:  112.6500529669463
17  Cost:  112.65001658353178
18  Cost:  112.64998039901865
19  Cost:  112.64994426496071
20  Cost:  112.64990814400622
21  Cost:  112.64987202675677
22  Cost:  112.64983591084761
23  Cost:  112.64979979568368
24  Cost:  112.64976368111523
25  Cost:  112.64972756710469
26  Cost:  112.64969145364236
27  Cost:  112.64965534072611
28  Cost:  112.64961922835512
29  Cost:  112.64958311652944
30  Cost:  112.64954700524868
31  Cost:  112.64951089451318
32  Cost:  112.64947478432279
33  Cost:  112.64943867467744
34  Cost:  112.64940256557728
35  Cost:  112.64936645702221
36  Cost:  112.64933034901203
37  Cost:  112.64929424154704
38  Cost:  112.64925813462712
39  Cost:  112.6492220282522
40  Cost:  112.64918592242235
41  Cost:  112.64914981713754
42  Cost:  112.64911371239779
43  Cost:  112.64907760820296
44  Cost:  112.64904150455324
45  Cost:  112.64900540144845
46  Cost:  112.64896929888867
47  Cost:  112.64893319687388
48  Cost:  112.6488970954041
49  Cost:  112.64886099447922
50  Cost:  112.64882489409929
51  Cost:  112.64878879426433
52  Cost:  112.64875269497436
53  Cost:  112.64871659622933
54  Cost:  112.64868049802914
55  Cost:  112.648644400374
56  Cost:  112.64860830326366
57  Cost:  112.64857220669828
58  Cost:  112.64853611067772
59  Cost:  112.64850001520212
60  Cost:  112.64846392027131
```

```
61  Cost:  112.64842782588545
62  Cost:  112.64839173204442
63  Cost:  112.6483556387483
64  Cost:  112.64831954599697
65  Cost:  112.64828345379043
66  Cost:  112.64824736212877
67  Cost:  112.64821127101193
68  Cost:  112.64817518043986
69  Cost:  112.64813909041264
70  Cost:  112.64810300093015
71  Cost:  112.64806691199259
72  Cost:  112.64803082359971
73  Cost:  112.64799473575155
74  Cost:  112.64795864844827
75  Cost:  112.64792256168963
76  Cost:  112.64788647547579
77  Cost:  112.64785038980668
78  Cost:  112.64781430468226
79  Cost:  112.64777822010265
80  Cost:  112.6477421360677
81  Cost:  112.64770605257743
82  Cost:  112.64766996963193
83  Cost:  112.64763388723107
84  Cost:  112.64759780537483
85  Cost:  112.64756172406335
86  Cost:  112.6475256432965
87  Cost:  112.64748956307432
88  Cost:  112.64745348339677
89  Cost:  112.64741740426388
90  Cost:  112.6473813256756
91  Cost:  112.64734524763193
92  Cost:  112.64730917013293
93  Cost:  112.6472730931785
94  Cost:  112.64723701676861
95  Cost:  112.64720094090339
96  Cost:  112.64716486558265
97  Cost:  112.64712879080662
98  Cost:  112.64709271657513
99  Cost:  112.64705664288809
1.4788027175308358 0.035074970592341756
```