BITS Pilani

Pilani Campus

Mathematical Foundations

MFDS Team

innovate achieve lead

BITS Pilani

Pilani Campus

Mathematical Foundations

Webinar 4

Agenda

- Gradient descent
- Stochastic gradient descent method
- Momentum based methods
- Ada Grad method, RMS Prop method, Adam method
- Previous Year Problems

Steps involved in Gradient Descent

Step 1: Initialize Parameters

Start by initializing the parameters of the model you're trying to optimize.

Step 2: Compute the Gradient

For the current set of parameters, compute the gradient of the loss function.

Step 3: Choose a Learning Rate

The learning rate, often denoted by α, controls how much we adjust the parameters with respect to the gradient of the loss function.

A too-small learning rate makes the convergence slow, while a too-large learning rate can lead to overshooting the minimum or diverging.

Steps involved in Gradient Descent

Step 4: Update Parameters

Update the parameters in the direction opposite to the gradient because we're seeking to minimize the loss function. The update rule for each parameter is:

$$x_{i+1} = x_i + \alpha_i\left(-\nabla f(x_i)\right).$$

Step 5: Repeat Until Convergence

Repeat Steps 3 and 4 for a set number of iterations, or until the change in the loss function from one iteration to the next is below a certain threshold, indicating convergence.

Example on GDM

Suppose we have a function $f(x) = x^2 - 4x + 4$. Find the value of x that minimizes $f(x)$ using the Gradient Descent Method.

Solution: For $f(x) = x^2 - 4x + 4$, the gradient is: $f'(x) = 2x - 4$

Gradient Descent Steps :

Initialization: Let's choose ($x_0 = 0$) as the starting point.

Learning Rate (α): Select a learning rate. For simplicity, let's choose α = 0.1.

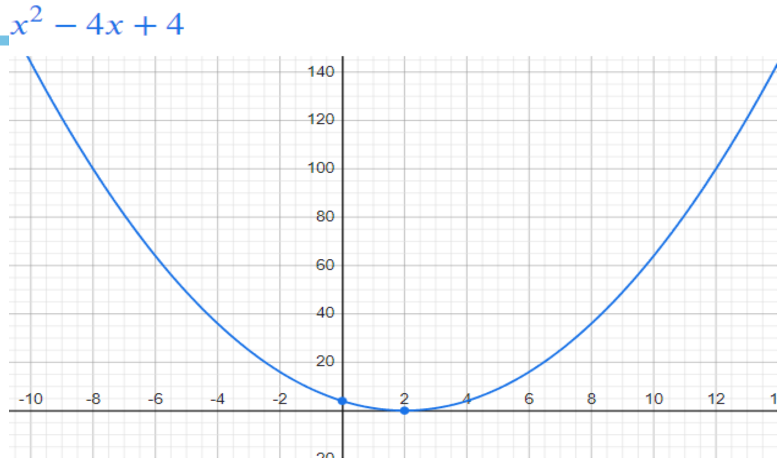
Update Rule: Apply the Gradient Descent update rule: $x_{i+1} = x_i + \alpha_i\left(-\nabla f(x_i)\right)$

$$\boldsymbol{x_{k+1} = x_k - \alpha\,(2(x_k) - 4)}$$

Iteration: We iterate this update rule to move (x) closer to the function’s minimum with each step.

Numerical illustration of GDM

Iteration	Value of x	Value of f(x)
0	$x_0 = 0$	4.0
1	$x_1 = 0.4000$	2.56
2	$x_2 = 0.7200$	1.6384
3	$x_3 = 0.9760$	1.0485
4	$x_4 = 1.1808$	0.671088
5	$x_5 = 1.3446$	0.4294
6	$x_6 = 1.4757$	0.274877
7	$x_7 = 1.5806$	0.175921
8	$x_8 = 1.6645$	0.1125
9	$x_9 = 1.7316$	0.07205
10	$x_{10} = 1.7853$	0.04611
11	$x_{11} = 1.8282$	0.02951
12	$x_{12} = 1.8626$	0.018889
13	$x_{13} = 1.8901$	0.012089
14	$x_{14} = 1.9121$	0.007732
15	$x_{15} = 1.9297$	0.004951
16	$x_{16} = 1.9438$	0.003169
17	$x_{17} = 1.9550$	0.002028
18	$x_{18} = 1.9640$	0.00129
19	$x_{19} = 1.9712$	0.0008308
20	$x_{20} = 1.9770$	0.0005317



After a few iterations, notice $(\boldsymbol{x_k})$ converging towards 2, which is the minimum of $f(x) = x^2 - 4x + 4$.

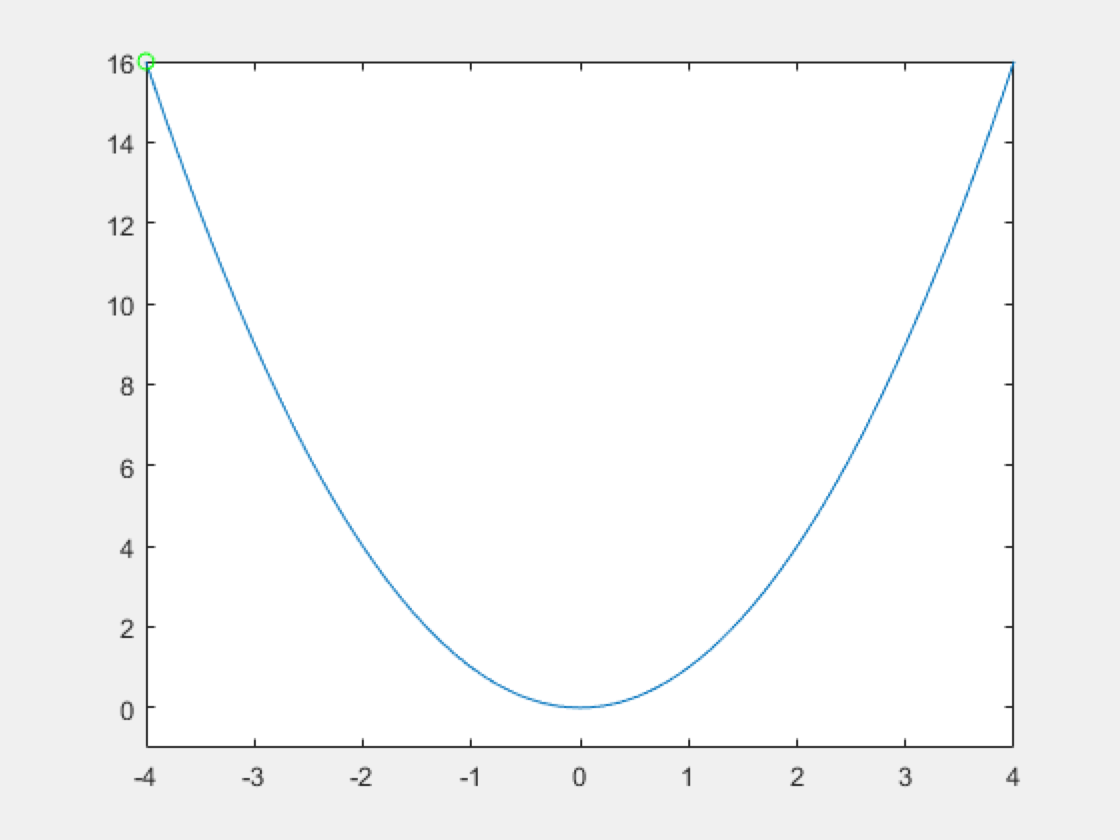
Conclusion: This example demonstrates the application of the GDM to minimize a simple quadratic function. By iteratively updating our estimate based on the derivative of the function, we effectively navigate towards the function's minimum.

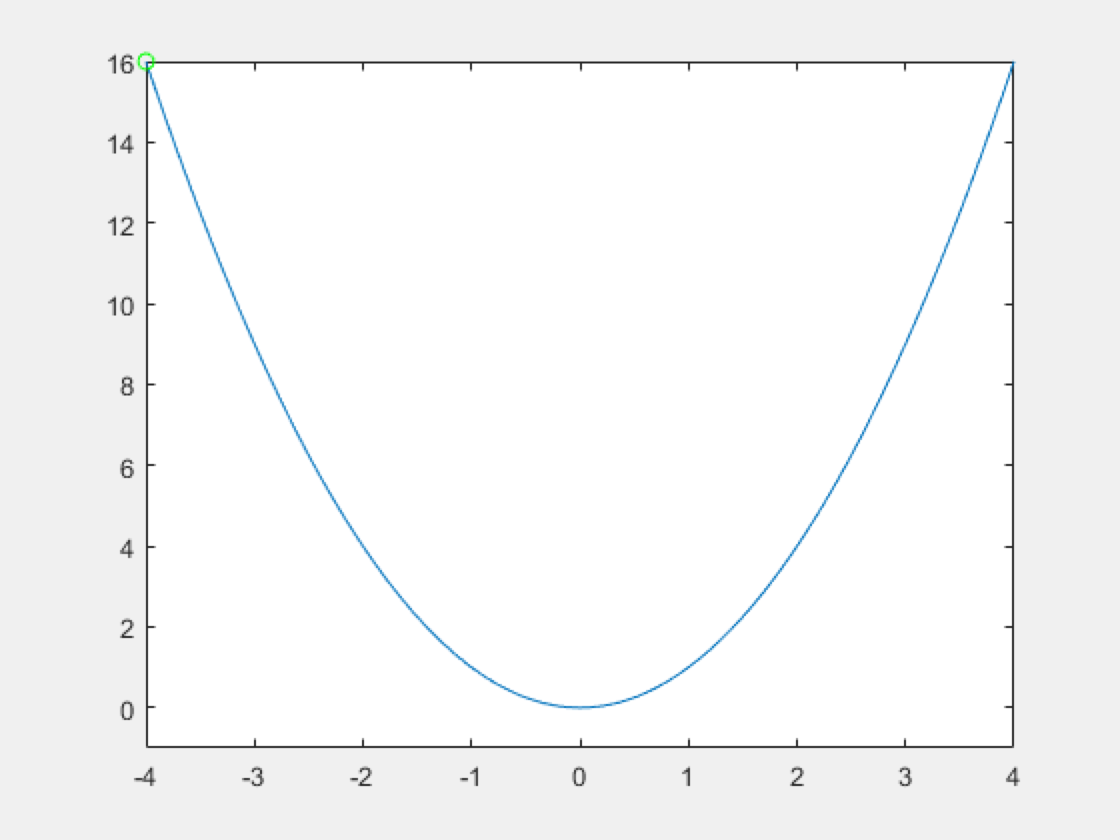
Let's consider the function $f(x) = x^2 - 4x + 4$ with different learning rates α (upto 3 iterations)

	Iterations (x)	f(x)	Analysis
Case I α = 0.01 $x_0 = 0$	$x_1 = 0.04$ $x_2 = 0.0792$ $x_3 = 0.1176$	3.8416 3.6881 3.5394	Slow Convergence
Case II α = 0.1 $x_0 = 0$	$x_1 = 0.4$ $x_2 = 0.72$ $x_3 = 0.976$	2.56 1.6384 1.0485	Efficient Convergence
Case III α = 1.2 $x_0 = 0$	$x_1 = 4.8$ $x_2 = -1.92$ $x_3 = 7.480$	7.84 15.3664 30.0561	Overshooting

Conclusion: This tabular comparison highlights the critical role of selecting an appropriate learning rate (α) in the gradient descent process, balancing convergence speed and the risk of overshooting or not converging at all.

$$f(x) = x^2, x_0 = -4,$$

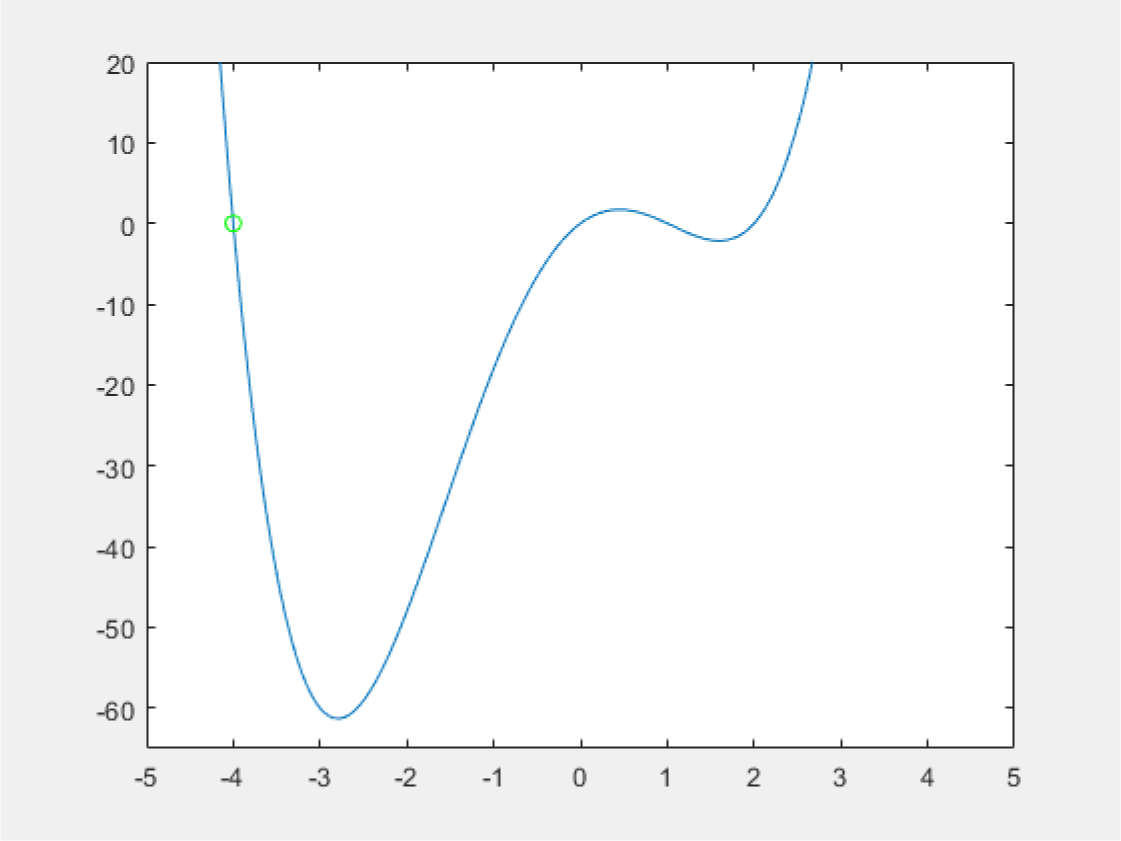




$\propto = 0.9$

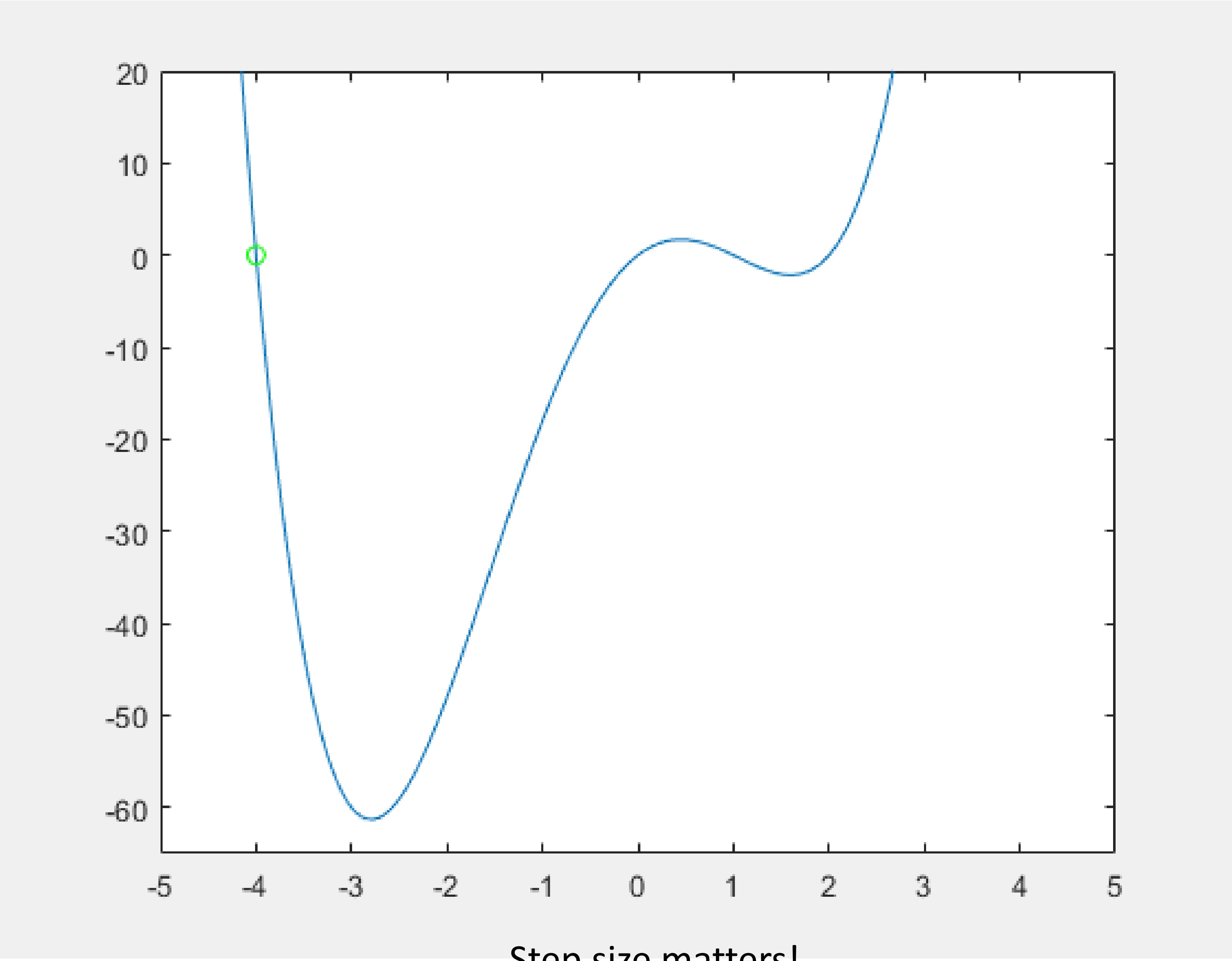
$\propto = 0.2$

Gradient Descent



Step size matters!

Gradient Descent



Step size matters!

innovate achieve lead

Previous Year Problems

Q1:

Consider the following optimization problem (A) on the data $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_n, y_n)$ of the following form:

$$\max \sum_{i=1}^{i=n} \alpha_i - \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \alpha_i \alpha_j \boldsymbol{x}_i . \boldsymbol{x}_j$$

$$\text{subject to } \sum_{i=1}^{i=n} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \forall i$$

Note that here y_i is $\pm 1, \forall i$ and each $\boldsymbol{x}_i$ is a $n \times 1$ vector. The variables in the problem are the α_i. We form a new optimization problem (B) as follows:

$$\max \sum_{i=1}^{i=n+1} \alpha_i - \sum_{i=1}^{i=n+1} \sum_{j=1}^{j=n+1} \alpha_i \alpha_j \boldsymbol{x}_i . \boldsymbol{x}_j$$

$$\text{subject to } \sum_{i=1}^{i=n+1} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \forall i, 1 \leq i \leq n+1$$

where $\boldsymbol{x}_{n+1} = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_j)$ for some values of i and j such that $y_i = y_j$. We set $y_{n+1} = y_i$. Show that the maximum value of the objective function for the problem (B) is greater than or equal to the maximum value of the objective function for problem (A). Justify your solution mathematically.

[3 Marks]

Previous Year Problems continued….

Q1) Solution: Let $(\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)$ be the optimal solution for problem (A), and let the optimal objective value be O_A.
For problem (B), let us set:

$$(\alpha_1, \alpha_2, \ldots, \alpha_n, \alpha_{n+1}) = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*, 0).$$

It is easy to see that this tuple is a feasible solution for problem (B) since:

- Both constraints in problem (B) are satisfied.
- Since $\alpha_{n+1} = 0$, its contribution to the objective function is null.

Evaluating the objective function of problem (B) for this assignment:

$$\begin{aligned}\text{Objective function of (B)} &= \sum_{i=1}^{n+1} \alpha_i - \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} \alpha_i \alpha_j x_i \cdot x_j.\\ &= \sum_{i=1}^{n} \alpha_i^* - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^* \alpha_j^* x_i \cdot x_j.\\ &= O_A.\end{aligned}$$

Since O_A is a feasible objective value for problem (B), and the optimal solution must be greater than or equal to any feasible solution, we conclude:

$$O_B \geq O_A.$$

Thus, the maximum value of the objective function for problem (B) is greater than or equal to the maximum value of the objective function for problem (A).

Previous Year Problems continued….

Q2: Consider a gradient update rule given by:

$$a_{t+1} = \gamma a_t + (1-\gamma)\nabla_w(L)$$
$$w_{t+1} = w_t - a_{t+1}$$

What is the contribution of a_0 while computing the value of a_5

[2 Marks]

Q2) Solution:

Given update rule:

$$a_{t+1} = \gamma a_t + (1-\gamma)\nabla_w L \qquad (1)$$

Expanding iteratively:

$$\begin{aligned}
a_1 &= \gamma a_0 + \dots \\
a_2 &= \gamma a_1 + \dots = \gamma^2 a_0 + \dots \\
a_3 &= \gamma a_2 + \dots = \gamma^3 a_0 + \dots \\
a_4 &= \gamma a_3 + \dots = \gamma^4 a_0 + \dots \\
a_5 &= \gamma a_4 + \dots = \gamma^5 a_0 + \dots
\end{aligned}$$

Conclusion: The contribution of a_0 in computing a_5 is $\gamma^5 a_0$.
This shows an exponential decay of a_0 controlled by γ. This means that a_0 decays exponentially by a factor of γ^t at each iteration. The larger γ, the more influence a_0 retains over time. Conversely, for smaller γ, the contribution of a_0 diminishes rapidly.

Previous Year Problems continued….

Q3: Consider a function $f(x,y) = 3x^2+2y^2$. Assume that we use the gradient descent algorithm with momentum term to find the minimum of this function $f(x,y)$. Let the momentum/friction parameter used in this algorithm be referred to as β. Find the value of β if you are given the following information about the algorithm:

(i) Initial point of algorithm is $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

(ii) The iterates obtained after 3 iterations is given as $\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} -7.36 \\ 1.12 \end{bmatrix}$

(iii) A fixed step size is used for all iterations and its value is $\alpha = 0.5$

Q3) Solution:

Given:

- Function: $f(x, y) = 3x^2 + 2y^2$
- Gradient Descent with Momentum (β)
- Initial point: $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$
- Iterates after 3 steps: $\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} -7.36 \\ 1.12 \end{bmatrix}$
- Step size: $\alpha = 0.5$

The update step for gradient descent with momentum:

$$\mathbf{z}_{i+1} = \mathbf{z}_i - \alpha \nabla f(\mathbf{z}_i) + \mathbf{v}_i \tag{2}$$

where:

$$\mathbf{v}_i = \beta(\mathbf{z}_i - \mathbf{z}_{i-1}), \quad \mathbf{v}_0 = 0 \tag{3}$$

Gradient: $\nabla f = \begin{bmatrix} 6x \\ 4y \end{bmatrix}$ **Deriving** x_3**:**

$$x_1 = x_0 - 0.6x_0 = 2 - 1.2 = 0.8$$
$$x_2 = x_1 - 0.6x_1 + \beta(x_1 - x_0) = -6.8\beta + 0.8$$
$$x_3 = x_2 - 0.6x_2 + \beta(x_2 - x_1) = -6.8\beta^2 + 24\beta - 16.8$$

Solving quadratic equation for β, we get $\beta = 0.4$ (valid as $\beta \in (0, 1)$)
Deriving y_3**:**

$$y_1 = y_0 - 0.4y_0 = 4 - 1.6 = 2.4$$
$$y_2 = y_1 - 0.4y_1 + \beta(y_1 - y_0) = (4 - 8\beta)$$
$$y_3 = y_2 - 0.4y_2 + \beta(y_2 - y_1) = -8\beta^2 + 16\beta - 4$$

Solving quadratic equation for β, we get $\beta = 0.4$
From both approaches, we get $\beta = 0.4$ which is the valid momentum term.

Previous Year Problems continued….

Q4: Let $f(x) = ax^2 + bx + c$ where $a > 0$. We intend to find a local minimum of this function using gradient descent with hyperparameter λ. If x_n represents the value of the variable x after the nth step, is it possible to write $x_{n+k} = x_n P^k + Q$ where P and Q are constants? If so, find P and Q. Otherwise explain why you cannot write x_{n+k} in the form suggested. Does there exist a set of values of λ for which the algorithm is guaranteed to converge? If so find these values of λ. Give a mathematical justification for your answer. (5 Marks)

Q4) Solution:

Q4) Solution:

Given:

- Function: $f(x) = ax^2 + bx + c, \quad a > 0$
- Gradient descent update rule: $x_{n+1} = x_n - \lambda \frac{\partial f}{\partial x}$
- Find if x_{n+k} can be written as $x_{n+k} = x_n P^k + Q$
- Find values of λ for guaranteed convergence.

$$\frac{\partial f}{\partial x} = 2ax + b. \tag{9}$$

Update equation:

$$x_{n+1} = x_n - \lambda(2ax_n + b). \tag{10}$$

Rearrange:

$$x_{n+1} = x_n(1 - 2a\lambda) - \lambda b. \tag{11}$$

Compute x_{n+2}:

$$x_{n+2} = x_{n+1}(1 - 2a\lambda) - \lambda b. \tag{12}$$

Generalizing for k steps:

$$x_{n+k} = x_n(1 - 2a\lambda)^k - \lambda b \sum_{i=0}^{k-1} (1 - 2a\lambda)^i. \tag{13}$$

This is of the form:

$$x_{n+k} = x_n P^k + Q, \tag{14}$$

where $P = (1 - 2a\lambda)$ and $Q = -\lambda b \sum_{i=0}^{k-1} (1 - 2a\lambda)^i$.

For convergence, we require:

$$|1 - 2a\lambda| < 1. \tag{15}$$

Solving:

$$-1 < 1 - 2a\lambda < 1. \tag{16}$$

Rearrange:

$$0 < \lambda < \frac{1}{a}. \tag{17}$$

Thus, the algorithm converges if:

$$0 < \lambda < \frac{1}{a}. \tag{18}$$

Stochastic gradient descent method

Step 1: Randomly shuffle the data set of size m

Step 2: Select a learning rate α

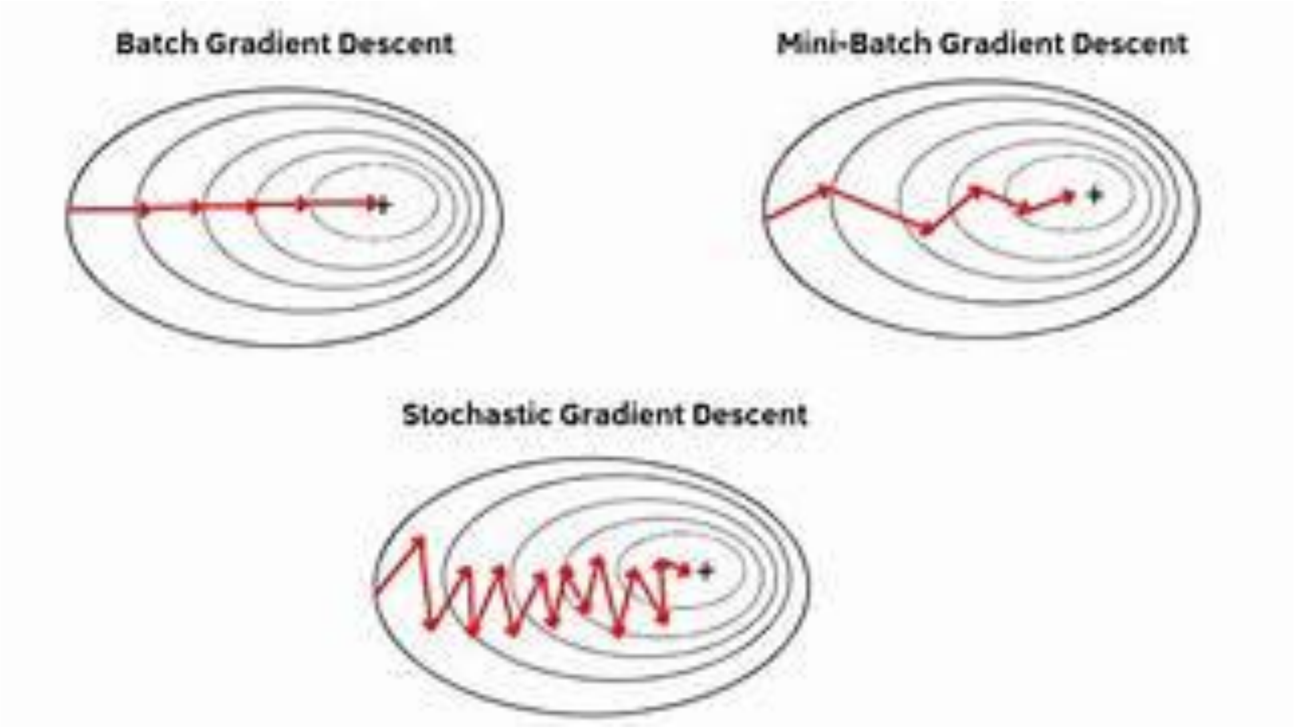
Step 3: Select initial parameter values θ as the starting point

Step 4: Update all parameters from the gradient of a single training example x^j, y^j, i.e. compute

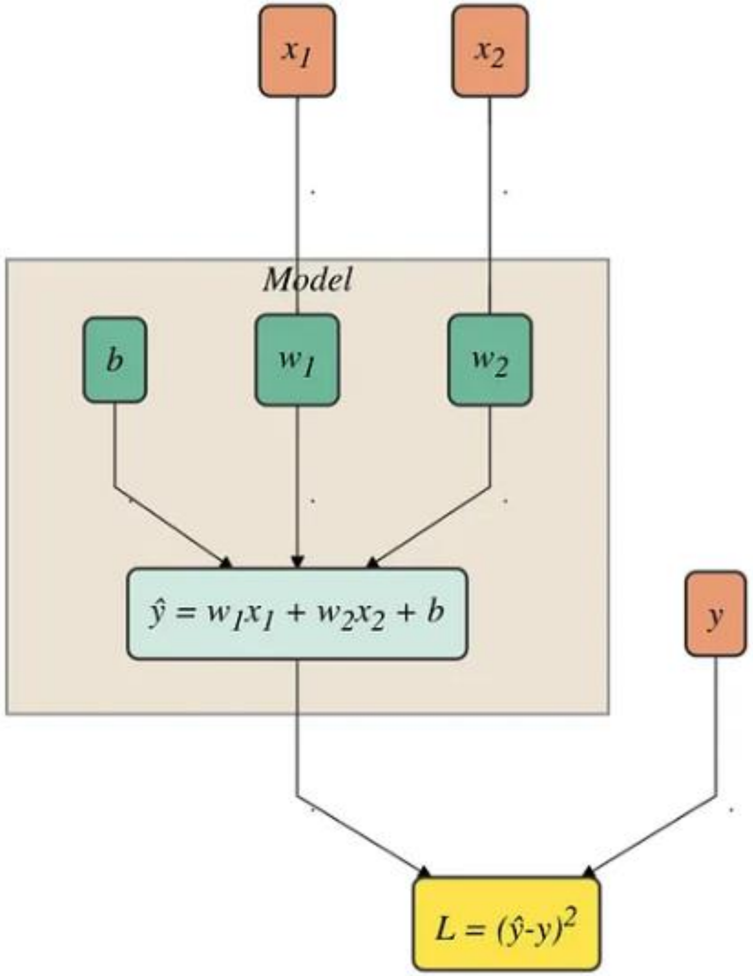
$$\theta_{i+1} = \theta_i - \alpha \times \nabla_\theta J(\theta; x^j; y^j)$$

Step 5: Repeat Step 4 until a local minimum is reached

Stochastic gradient descent method

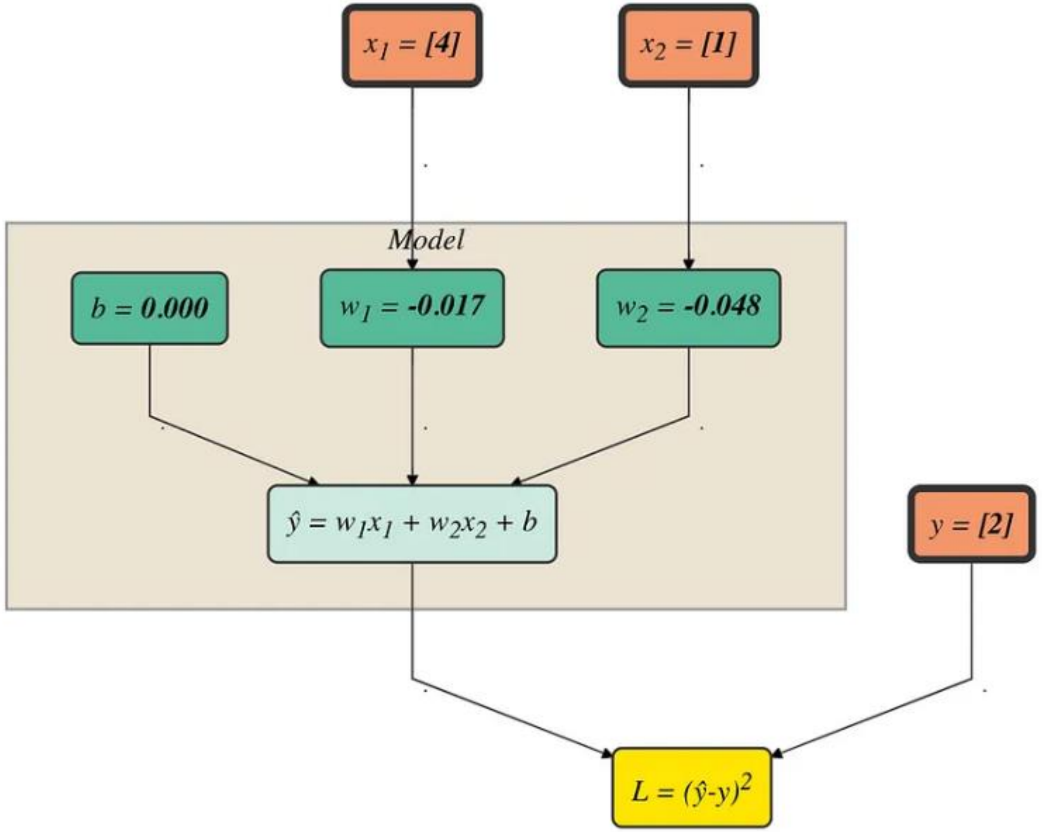


Example: SGD for Linear Regression



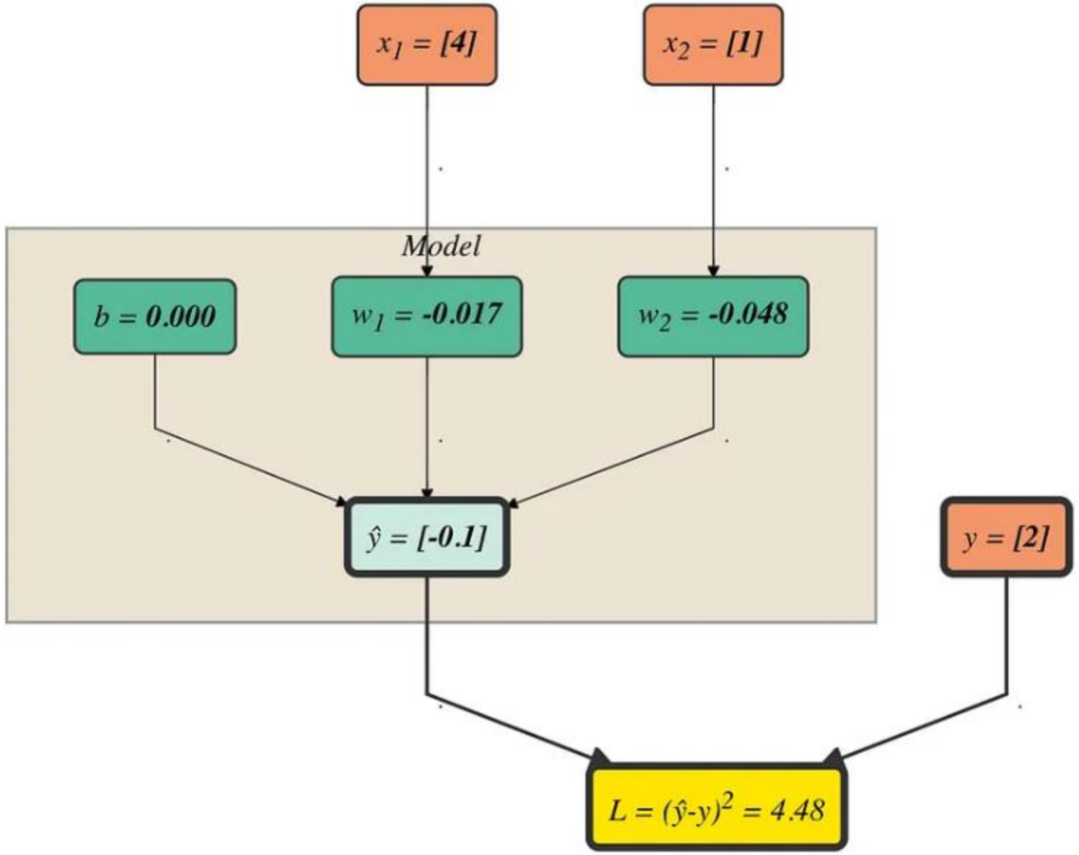
Dataset

	x1	x2	y
1)	**4**	**1**	**2**
2)	2	8	-14
3)	1	0	1
4)	3	2	-1
5)	1	4	-7
6)	6	7	-8



Feeding the first data into the model

Example: SGD for Linear Regression



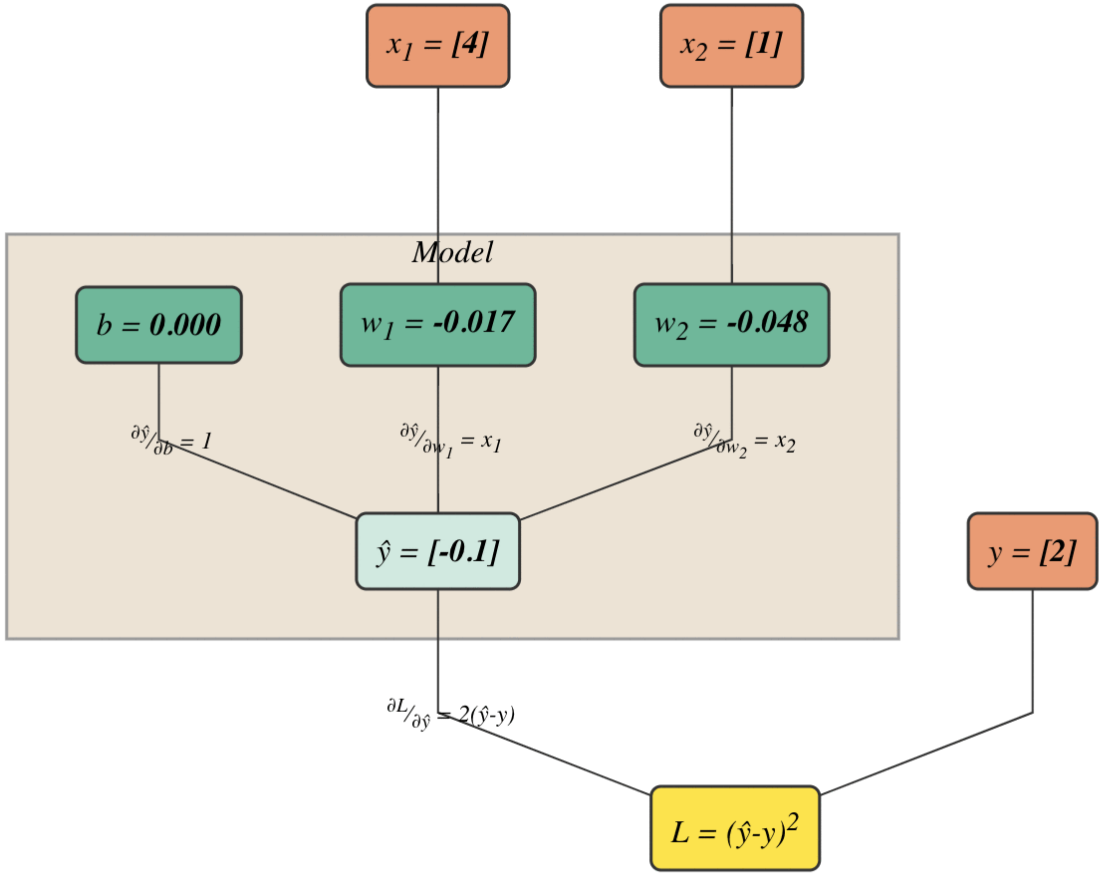
Stochastic gradient update formula:

$$
\begin{aligned}
b' &= b - \eta \frac{\partial L}{\partial b} \\
&= b - \eta \left(\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b}\right) \\
&= b - \eta [2(\hat{y} - y) \cdot 1] \\
&= 0.000 - 0.05[2(-0.116 - 2) \cdot 1] \\
&= 0.212
\end{aligned}
$$

$$
\begin{aligned}
w_1' &= w_1 - \eta \frac{\partial L}{\partial w_1} \\
&= w_1 - \eta \left(\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1}\right) \\
&= w_1 - \eta [2(\hat{y} - y) \cdot x_1] \\
&= -0.017 - 0.05[2(-0.116 - 2) \cdot 4] \\
&= 0.829
\end{aligned}
$$

Example: SGD for Linear Regression

```
Epoch: 1/1 Batch: 1/6
Backpropagation
```



$$
\begin{aligned}
w_2' &= w_2 - \eta \frac{\partial L}{\partial w_2} \\
&= w_2 - \eta\left(\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}\right) \\
&= w_2 - \eta[2(\hat{y} - y) \cdot x_2] \\
&= -0.048 - 0.05[2(-0.116 - 2) \cdot 1] \\
&= 0.164
\end{aligned}
$$

This iteration is performed on the remaining data points to obtain the final model.

Final Model:

$$\hat{y} = 0.43x_1 - 0.21x_2 + 0.77$$

Momentum based methods

Gradient Descent Algorithm:

Pick an initial point x_0 and $\propto$

Iterate until convergence

$$x_{k+1} = x_k - \propto \nabla f(x_k)$$

where,

$\propto$ is the step size and also called as learning rate.

Ada Grad

Ada Grad:

Pick an initial point x_0, $\propto$, A_i and ε

Iterate until convergence

$$x_{k+1} = x_k - \frac{\propto}{\sqrt{A_i + \varepsilon}} \nabla f(x_k)$$

where,

$\propto$ is the step size and also called as learning rate.

$A_i = A_i + \left(\frac{\partial f}{\partial x_i}\right)^2$ and $\varepsilon = 10^{-8}$.

RMS Prop

RMS Prop:

Pick an initial point x_0, $\propto$, A_i , ρ and ε

Iterate until convergence

$$x_{k+1} = x_k - \frac{\propto}{\sqrt{A_i + \varepsilon}} \nabla f(x_k)$$

where,

$\propto$ is the step size and also called as learning rate.

$A_i = \rho A_i + (1 - \rho) \left(\frac{\partial f}{\partial x_i}\right)^2$ (ρ is decay rate)

and $\varepsilon = 10^{-8}$.

Adam method

Adam:

Pick an initial point x_0, $\propto$, A$_i$, β_1, β_2 and ε

Iterate until convergence

$$x_{k+1} = x_k - \frac{\propto}{\sqrt{\frac{A_i}{(1-\beta_2)} + \varepsilon}} \frac{B_i}{(1-\beta_1)}$$

where, $\propto$ is the step size and also called as learning rate.

$$A_i = \beta_2 A_i + (1-\beta_2)\left(\frac{\partial f}{\partial x_i}\right)^2$$ (β_2- Exponential decay rate for the second moment estimates)

$$\mathrm{B}_i = \beta_1 A_i + (1-\beta_1)\left(\frac{\partial f}{\partial x_i}\right)$$ (β_1- Exponential decay rate for the first moment estimates)

and $\varepsilon = 10^{-8}$.

Problem on Gradient descent method

1.Find the minimum of f(x,y) = $3x^2+y^2$ with initial values $x_0 = 1$ and $y_0 = 3$ with learning rate $\alpha = 0.1$ using Gradient descent method.

Solution:

1^{st} Iteration

$$[x_1 \quad y_1] = [x_0 \quad y_0] - \alpha\left[\left(\frac{\partial f}{\partial x}\right)_{x=x_0} \quad \left(\frac{\partial f}{\partial y}\right)_{y=y_0}\right]$$

$$[x_1 \quad y_1] = [1 \quad 3] - (0.1)[6*1 \quad 2*3]$$

$$[x_1 \quad y_1] = [1 \quad 3] - [0.6 \quad 0.6] = [0.4 \quad 2.4]$$

```
Iteration 1 - x: 0.4000, y: 2.4000, f(x, y): 6.2400
Iteration 2 - x: 0.1600, y: 1.9200, f(x, y): 3.7632
Iteration 3 - x: 0.0640, y: 1.5360, f(x, y): 2.3716
Iteration 4 - x: 0.0256, y: 1.2288, f(x, y): 1.5119
Iteration 5 - x: 0.0102, y: 0.9830, f(x, y): 0.9667
Iteration 6 - x: 0.0041, y: 0.7864, f(x, y): 0.6185
Iteration 7 - x: 0.0016, y: 0.6291, f(x, y): 0.3958
Iteration 8 - x: 0.0007, y: 0.5033, f(x, y): 0.2533
Iteration 9 - x: 0.0003, y: 0.4027, f(x, y): 0.1621
Iteration 10 - x: 0.0001, y: 0.3221, f(x, y): 0.1038
Iteration 11 - x: 0.0000, y: 0.2577, f(x, y): 0.0664
Iteration 12 - x: 0.0000, y: 0.2062, f(x, y): 0.0425
Iteration 13 - x: 0.0000, y: 0.1649, f(x, y): 0.0272
Iteration 14 - x: 0.0000, y: 0.1319, f(x, y): 0.0174
Iteration 15 - x: 0.0000, y: 0.1056, f(x, y): 0.0111
Iteration 16 - x: 0.0000, y: 0.0844, f(x, y): 0.0071

Minimum found at x: 0.0000, y: 0.0844
Minimum value of the function: 0.0071
```

Problem on Ada Grad

2.Find the minimum of f(x,y) = $3x^2+y^2$ with initial values $x_0 = 1$ and $y_0 = 3$ with learning rate $\alpha = 0.9$ using Ada Grad method.

Solution:

1^{st} Iteration

$$[x_1 \quad y_1] = [x_0 \quad y_0] - \frac{\alpha}{\sqrt{A_i + \varepsilon}} \left[\left(\frac{\partial f}{\partial x}\right)_{x=x_0} \quad \left(\frac{\partial f}{\partial y}\right)_{y=y_0} \right]$$

$$x_1 = 1 - \frac{(0.9)}{\sqrt{36+10^{-8}}} * 6 = 0.1000000001$$

$$y_1 = 3 - \frac{(0.9)}{\sqrt{36+10^{-8}}} * 6 = 2.1$$

```
Iteration 1 - x: 0.1000, y: 2.1000, f(x, y): 4.4400
Iteration 2 - x: 0.0104, y: 1.5839, f(x, y): 2.5090
Iteration 3 - x: 0.0011, y: 1.2266, f(x, y): 1.5046
Iteration 4 - x: 0.0001, y: 0.9621, f(x, y): 0.9257
Iteration 5 - x: 0.0000, y: 0.7600, f(x, y): 0.5776
Iteration 6 - x: 0.0000, y: 0.6028, f(x, y): 0.3633
Iteration 7 - x: 0.0000, y: 0.4792, f(x, y): 0.2297
Iteration 8 - x: 0.0000, y: 0.3816, f(x, y): 0.1456
Iteration 9 - x: 0.0000, y: 0.3042, f(x, y): 0.0925
Iteration 10 - x: 0.0000, y: 0.2426, f(x, y): 0.0588
Iteration 11 - x: 0.0000, y: 0.1935, f(x, y): 0.0375
Iteration 12 - x: 0.0000, y: 0.1544, f(x, y): 0.0239
Iteration 13 - x: 0.0000, y: 0.1233, f(x, y): 0.0152
Iteration 14 - x: 0.0000, y: 0.0984, f(x, y): 0.0097
Iteration 15 - x: 0.0000, y: 0.0785, f(x, y): 0.0062
Iteration 16 - x: 0.0000, y: 0.0627, f(x, y): 0.0039


Optimization finished
Final values - x: 0.0000, y: 0.0627, f(x, y): 0.0039
```

Problem on RMS Prop

3.Find the minimum of f(x,y) = $3x^2+y^2$ with initial values $x_0 = 1$ and $y_0 = 3$ with learning rate $\alpha = 0.1$, $\rho = 0.9$ using method.

Solution:

1^{st} Iteration

$$[x_1 \quad y_1] = [x_0 \quad y_0] - \frac{\alpha}{\sqrt{A_i + \varepsilon}} \left[\left(\frac{\partial f}{\partial x}\right)_{x=x_0} \quad \left(\frac{\partial f}{\partial y}\right)_{y=y_0} \right]$$

$$x_1 = 1 - \frac{(0.9)}{\sqrt{((1-0.9)*36)+10^{-8}}} * 6 = 0.6837$$

$$y_1 = 3 - \frac{(0.9)}{\sqrt{((1-0.9)*36)+10^{-8}}} * 6 = 2.6837$$

```
Iteration 1 - x: 0.6838, y: 2.6838, f(x, y): 8.6053
Iteration 2 - x: 0.4989, y: 2.4668, f(x, y): 6.8318
Iteration 3 - x: 0.3692, y: 2.2918, f(x, y): 5.6611
Iteration 4 - x: 0.2728, y: 2.1411, f(x, y): 4.8074
Iteration 5 - x: 0.1998, y: 2.0067, f(x, y): 4.1466
Iteration 6 - x: 0.1443, y: 1.8843, f(x, y): 3.6131
Iteration 7 - x: 0.1024, y: 1.7712, f(x, y): 3.1686
Iteration 8 - x: 0.0712, y: 1.6655, f(x, y): 2.7893
Iteration 9 - x: 0.0484, y: 1.5661, f(x, y): 2.4598
Iteration 10 - x: 0.0321, y: 1.4721, f(x, y): 2.1701
Iteration 11 - x: 0.0207, y: 1.3827, f(x, y): 1.9130
Iteration 12 - x: 0.0130, y: 1.2974, f(x, y): 1.6838
Iteration 13 - x: 0.0079, y: 1.2160, f(x, y): 1.4787
Iteration 14 - x: 0.0046, y: 1.1380, f(x, y): 1.2950
Iteration 15 - x: 0.0026, y: 1.0632, f(x, y): 1.1304
Iteration 16 - x: 0.0014, y: 0.9915, f(x, y): 0.9831


Optimization finished.
Final values - x: 0.0014, y: 0.9915, f(x, y): 0.9831
```

Problem on Adam

4. Find the minimum of f(x,y) = $3x^2+y^2$ with initial values $x_0 = 1$ and $y_0 = 3$ with learning rate $\alpha = 0.2$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ using Adam method.

Solution:

1^{st} Iteration

$$[x_1 \quad y_1] = [x_0 \quad y_0] - \frac{\alpha}{\sqrt{\frac{\beta_2 A_i + (1-\beta_2)\left(\frac{\partial f}{\partial x_i}\right)^2}{(1-\beta_2)} + \varepsilon}} \left[\frac{\beta_1 A_i + (1-\beta_1)\left(\frac{\partial f}{\partial x_i}\right)}{(1-\beta_1)}\right]$$

$$x_1 = 1 - \frac{(0.2)}{\sqrt{\frac{((0.999*0)+(1-0.999)*36)}{(1-0.999)} + 10^{-8}}} * \left[\frac{((0.9*0)+(1-0.9)*6)}{(1-0.9)}\right] = 0.8$$

$$y_1 = 3 - \frac{(0.2)}{\sqrt{\frac{((0.999*0)+(1-0.999)*36)}{(1-0.999)} + 10^{-8}}} * \frac{((0.9*0)+(1-0.9)*6)}{(1-0.9)} = 2.8$$

```
Iteration 1 - x: 0.8000, y: 2.8000, f(x, y): 9.7600
Iteration 2 - x: 0.6024, y: 2.6005, f(x, y): 7.8511
Iteration 3 - x: 0.4097, y: 2.4018, f(x, y): 6.2723
Iteration 4 - x: 0.2258, y: 2.2044, f(x, y): 5.0124
Iteration 5 - x: 0.0556, y: 2.0087, f(x, y): 4.0442
Iteration 6 - x: -0.0949, y: 1.8153, f(x, y): 3.3222
Iteration 7 - x: -0.2196, y: 1.6246, f(x, y): 2.7841
Iteration 8 - x: -0.3139, y: 1.4374, f(x, y): 2.3617
Iteration 9 - x: -0.3759, y: 1.2543, f(x, y): 1.9971
Iteration 10 - x: -0.4066, y: 1.0759, f(x, y): 1.6537
Iteration 11 - x: -0.4093, y: 0.9031, f(x, y): 1.3182
Iteration 12 - x: -0.3882, y: 0.7367, f(x, y): 0.9948
Iteration 13 - x: -0.3480, y: 0.5774, f(x, y): 0.6967
Iteration 14 - x: -0.2933, y: 0.4261, f(x, y): 0.4396
Iteration 15 - x: -0.2284, y: 0.2836, f(x, y): 0.2370
Iteration 16 - x: -0.1577, y: 0.1506, f(x, y): 0.0973
```

Momentum-Based Gradient Descent

Gradient descent is an optimization algorithm commonly used to minimize a loss or cost function. It iteratively updates the parameters of a model in the direction opposite to the gradient of the function at the current point. The update rule for gradient descent is given by

$$x_{t+1} = x_t - \alpha\nabla f(x_t)$$

Where:

- x_t is the current value of the parameter vector.
- α is the learning rate.
- $\nabla f(x_t)$ is the gradient of the function f with respect to the parameter vector x evaluated at x_t.

Momentum-based gradient descent introduces a momentum term to accelerate convergence, especially in regions with high curvature or noisy gradients. The update rule for momentum-based gradient descent is given by:

$v_{t+1} = \beta v_t - \alpha\nabla f(x_t)$

$x_{t+1} = x_t + v_{t+1}$

Where:

- v_t is the velocity vector at time t.
- β is the momentum parameter.
- α is the learning rate.
- $\nabla f(x_t)$ is the gradient of the function f with respect to the parameter vector x evaluated at x_t.

Momentum-Based Gradient Descent

Problem:1

Consider the following quadratic function: $f(x) = x^2 + 10x + 25$

We want to use momentum-based gradient descent to minimize this function.
Let's start with an initial guess $x_0 = 0$, learning rate $\alpha = 0.1$, momentum parameter $\beta = 0.9$, and maximum iterations of 1000.

1. Initialize parameters: $x_0 = 0$, $v_0 = 0$.
2. For $t = 0,1,2,...$ until convergence or maximum iterations:

(a) Compute gradient: $\nabla f(x_t) = 2x_t + 10$.
(b) Update velocity: $v_{t+1} = 0.9v_t - 0.1(2x_t + 10)$.
(c) Update parameter: $x_{t+1} = x_t + v_{t+1}$.
(d) Check for convergence: If $|x_{t+1} - x_t| < \epsilon$, stop.

After running the algorithm,
the minimum of the function is found at $x \approx -5$ with a minimum function value of $f(x) \approx -25$.

Mini batch Gradient Descent:

- Mini batch gradient descent is a variant of the gradient descent algorithm that
- It computes the gradient and updates the parameters using a subset of the training data, called a mini batch.
- This approach is commonly used in large-scale machine learning tasks
- It reduces the computational cost and accelerates convergence compared to using the entire dataset.

Mini batch Gradient Descent:

- The mini batch gradient descent update rule is similar to the standard gradient descent. It computes the gradient and updates the parameters using a mini batch of the data
- The update rule is given by:

$$w_{t+1} = w_t - \alpha \nabla F(X_{minibatch})$$

Where:

- w_t is the current value of the parameter vector.
- α is the learning rate.
- $\nabla F(X_{minibatch})$ is the gradient of the loss function F with respect to the

parameter vector w computed using the mini batch $X_{minibatch}$.

Problem:

Consider a linear regression problem where we want to minimize the mean squared error (MSE) loss function:

$$F(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$$

Where **w** is the parameter vector

xi are the input features

yi are the target values

n is the number of samples in the dataset

Execution:

We'll start with an initial guess $w_0 = 0$,
learning rate α = 0.01,
Mini batch size m = 10,
maximum iterations of 1000.

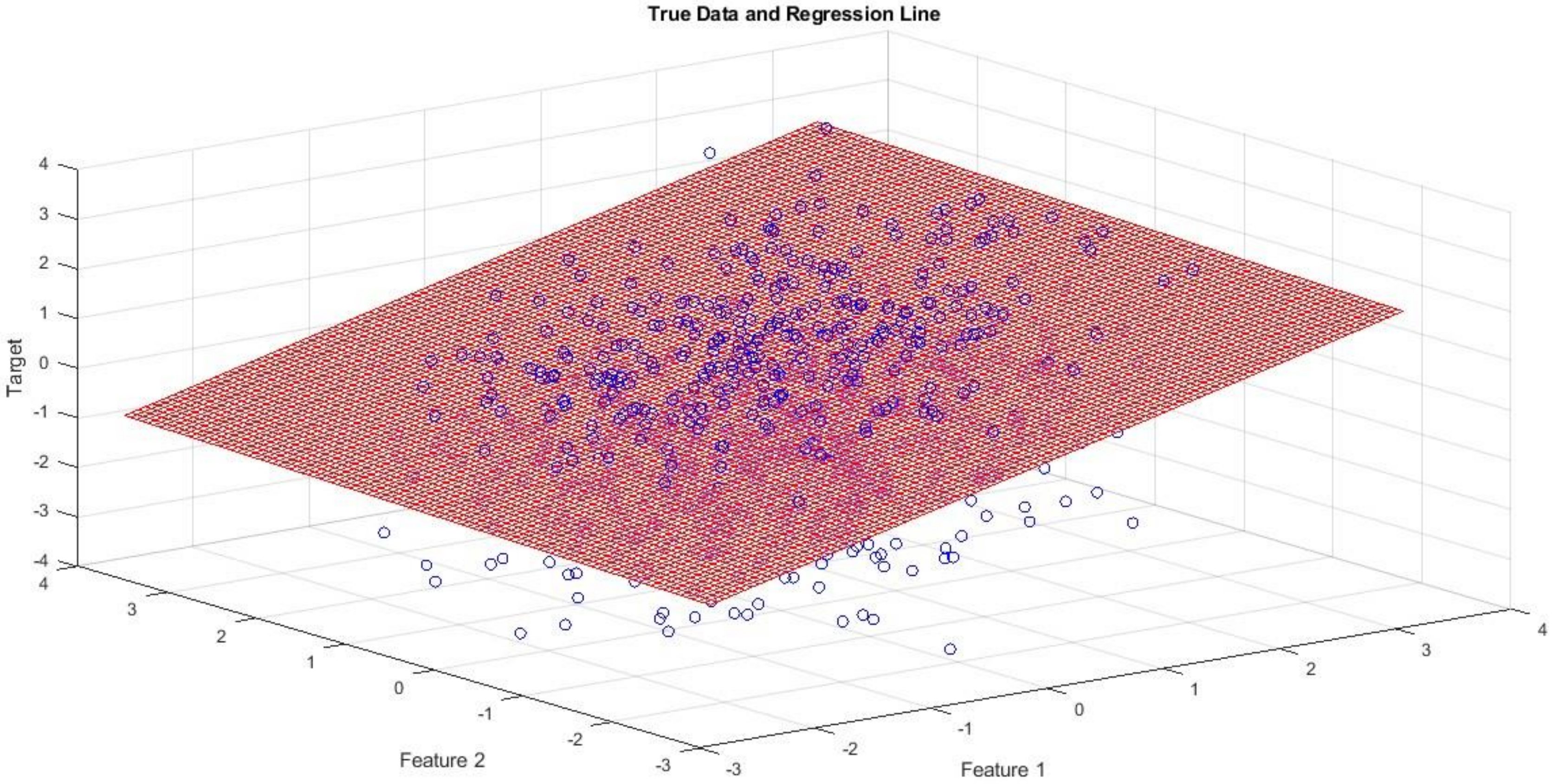
1. Initialize parameters: $w_0 = 0$.
2. For t = 0, 1, 2, . . . until convergence or maximum iterations:
(a) Sample a mini batch of size m from the dataset.
(b) Compute gradient: $\nabla F(X_{Minibatch}) = \frac{2}{m} X^T_{\text{Xminibatch}} (X_{Minibatch} w'_t - Y_{minibatch})$
(c) Update parameter: $w_{t+1} = w_t - \alpha \nabla F(X_{minibatch})$.
(d) Check for convergence: If the change in w is small or maximum iterations reached, stop.

Output:

```
Iteration 10 - Loss: 7.6776
Iteration 20 - Loss: 5.5925
Iteration 30 - Loss: 4.1993
Iteration 40 - Loss: 2.9389
Iteration 50 - Loss: 2.2062
Iteration 60 - Loss: 1.8224
Iteration 70 - Loss: 1.5082
Iteration 80 - Loss: 1.3028
Iteration 90 - Loss: 1.1937
Iteration 100 - Loss: 1.1467
```

```
Optimal parameter vector:     0.5889
    0.0558
    0.6917

True parameter vector:     0.5591
    0.1532
    0.7327
```


True Data and Regression Line
Target
Feature 2
Feature 1
4
3
2
1
0
-1
-2
-3
-4

Thank You