END Seminar Review

Comparison of Optimizer's in Neural Network

Name: Bapuram Venkateswar Reddy

Roll no: 21MAE0003

Email: bv21mae0003@student.nitw.ac.in

Guide: Prof. R. S. Selvaraj Sir

Content:

Introduction

Optimizers are crucial in training neural networks by adjusting model weights to minimize loss. Traditional methods like Gradient Descent (GD) are slow, while SGD and Momentum-based optimizers improve speed and stability. Adaptive methods like Adagrad, RMSprop, and Adam adjust learning rates dynamically, enhancing performance in deep networks. Recent advancements like AdamW and LAMB optimize large-scale models. This seminar explores different optimizers, their strengths, and their impact on neural network training

Gradient

- It is the measure of degree of Steepness.
- It points in the direction of steepest ascent or descent.
- The gradient of a hill is a measure of how steep it is, and it points in the direction of the steepest ascent
- If you walk in the direction of the gradient, you will be going straight up the hill.

Formula : Gradient Descent

observation i

learning rate

$$\underbrace{w^{(t+1)}}_{\text{position of next iteration}} = w - \underbrace{\alpha \nabla f_i(w^{(t)})}_{\text{step}}$$

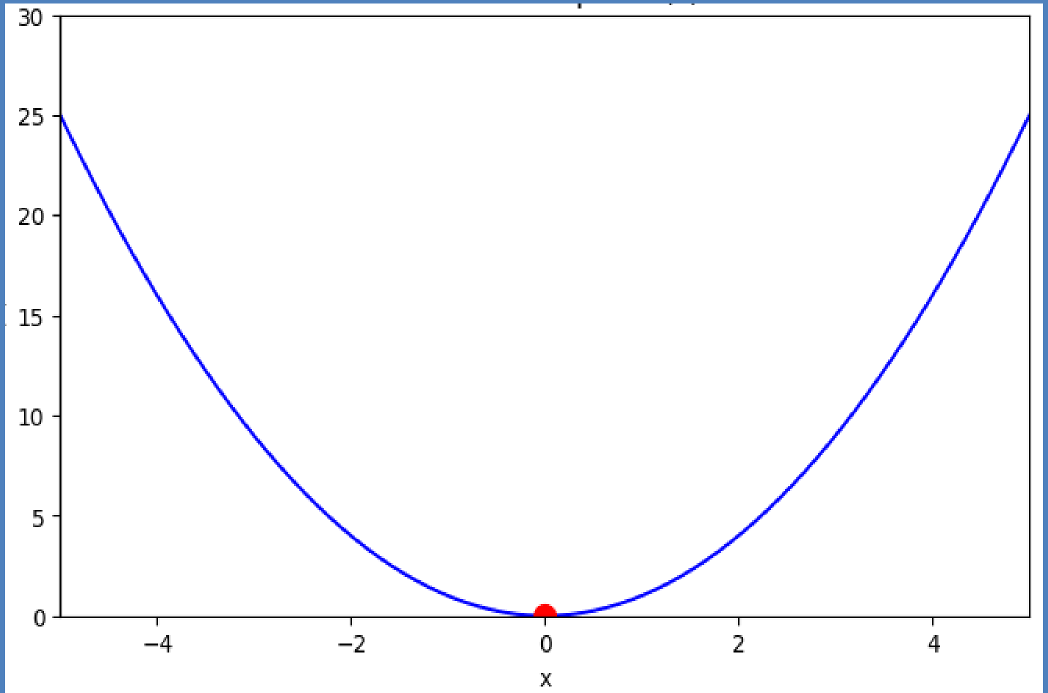
position of previous step

Absence of Learning Rate term in Gradient Descent

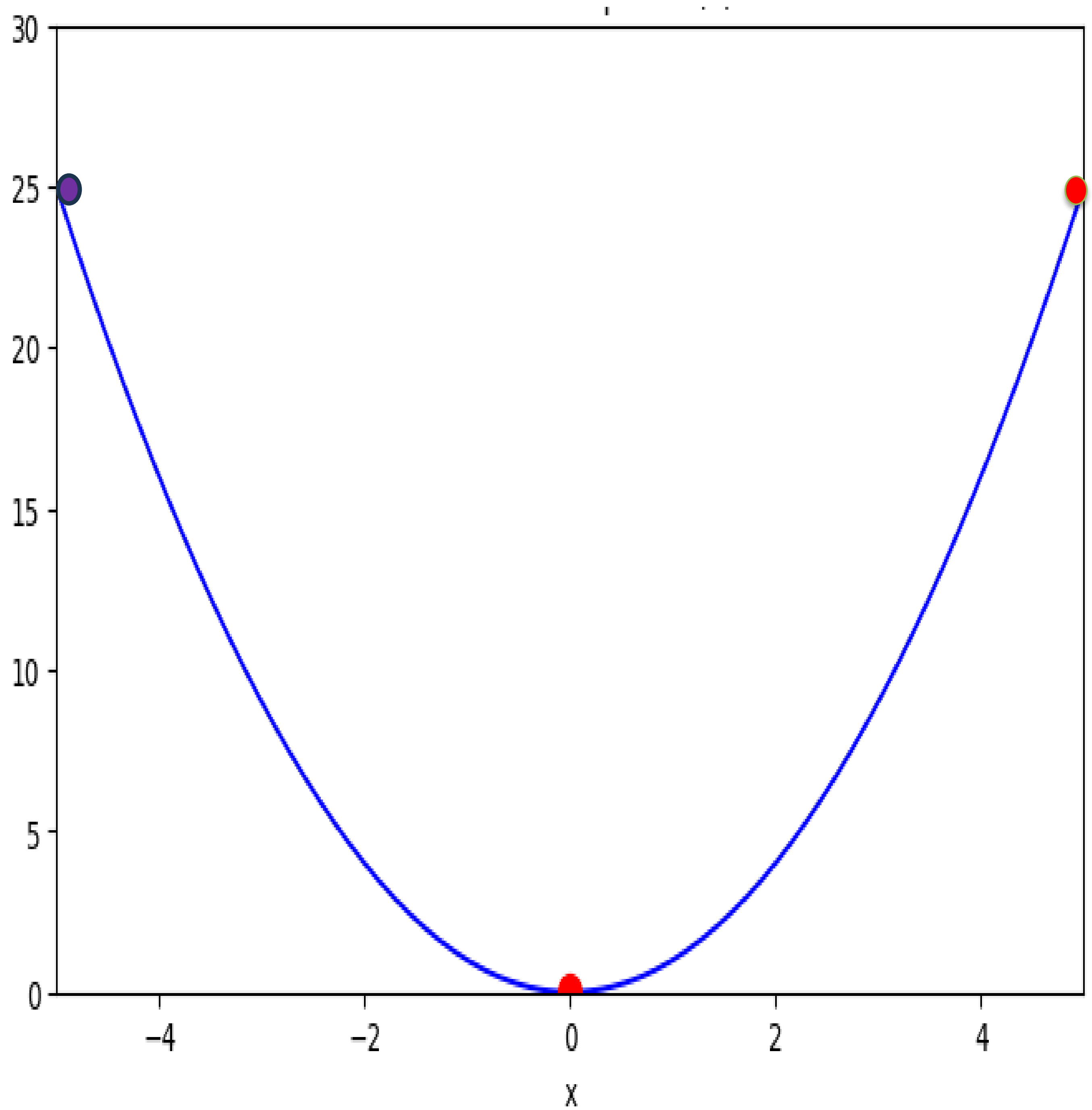
$$New\_m = Old\_m - Gradient$$

Assumptions

- ✓ Error function = $x^{2.}$
- ✓ Learning rate = 0.1
- ✓ Gradient = $2 \times x$
- ✓ Initial value m = 5



$y = x^2$



New_m = Old_m - Gradient

Gradient =Step_size

Iteration :1 , Gradient= 10, New_m= - 5

Iteration :2 , Gradient = -10, New_m= 5

Iteration :3 , Gradient= 10, New_m= - 5

Iteration :4 , Gradient = -10, New_m=5

Iteration :5 , Gradient= 10, New_m= - 5

Iteration :6 , Gradient = -10, New_m= 5

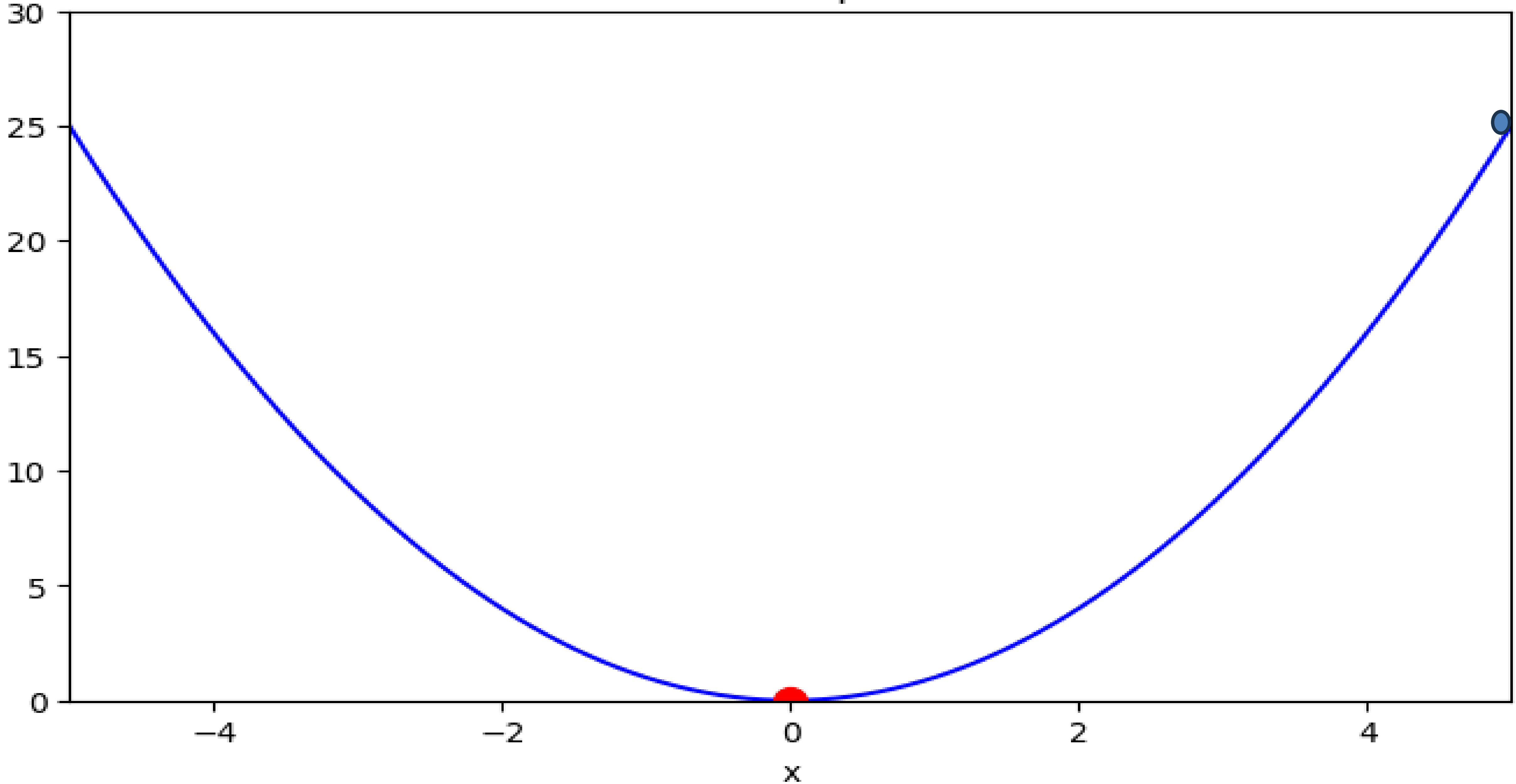
Iteration :7 , Gradient= 10, New_m= - 5

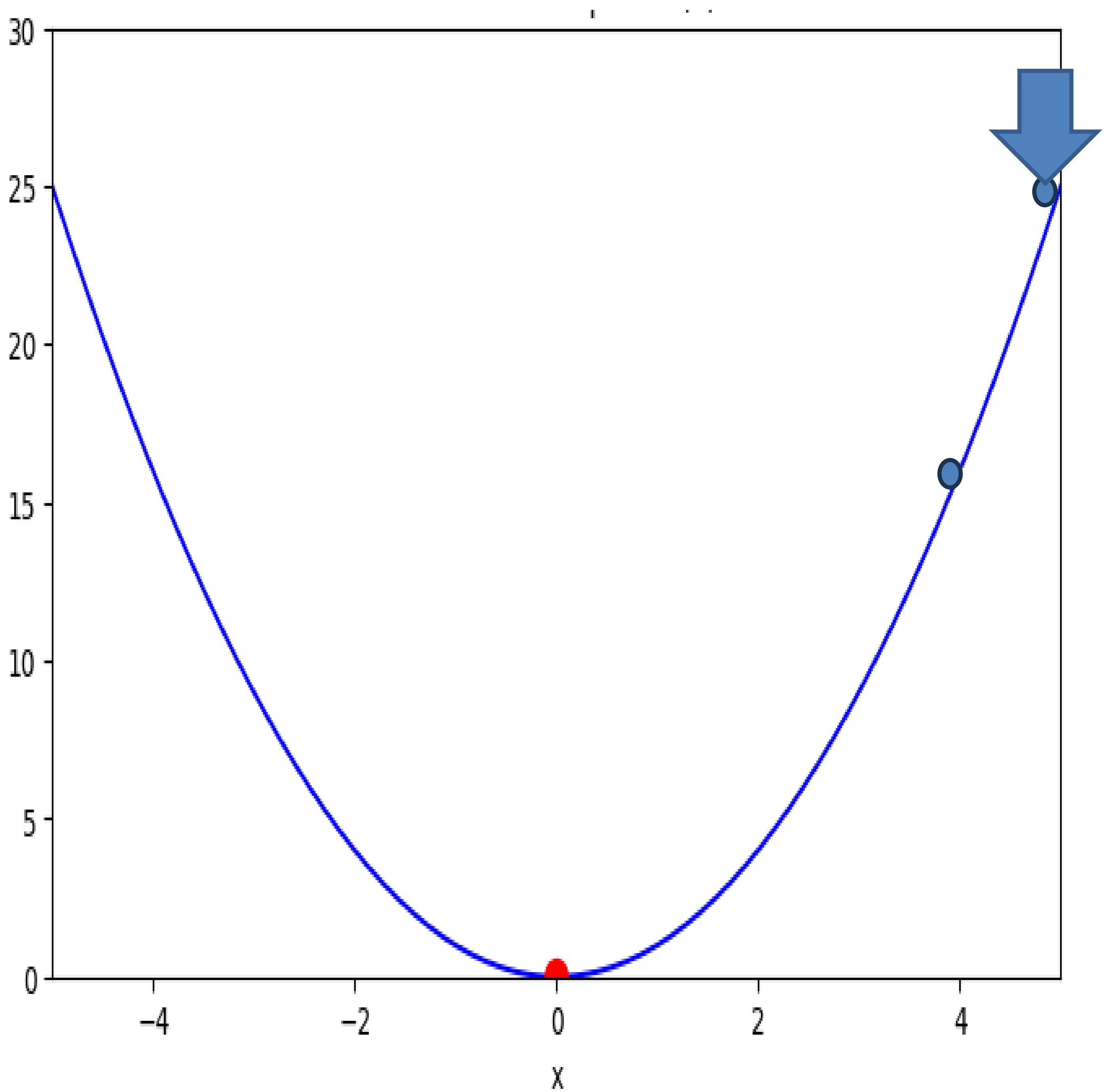
Iteration :8 , Gradient = -10, New_m= 5

Iteration :30 , Gradient = -10, New_m= 5

Iteration :50 , Gradient = -10, New_m= 5

Visualization of Iterations





Learning Rate =0.1

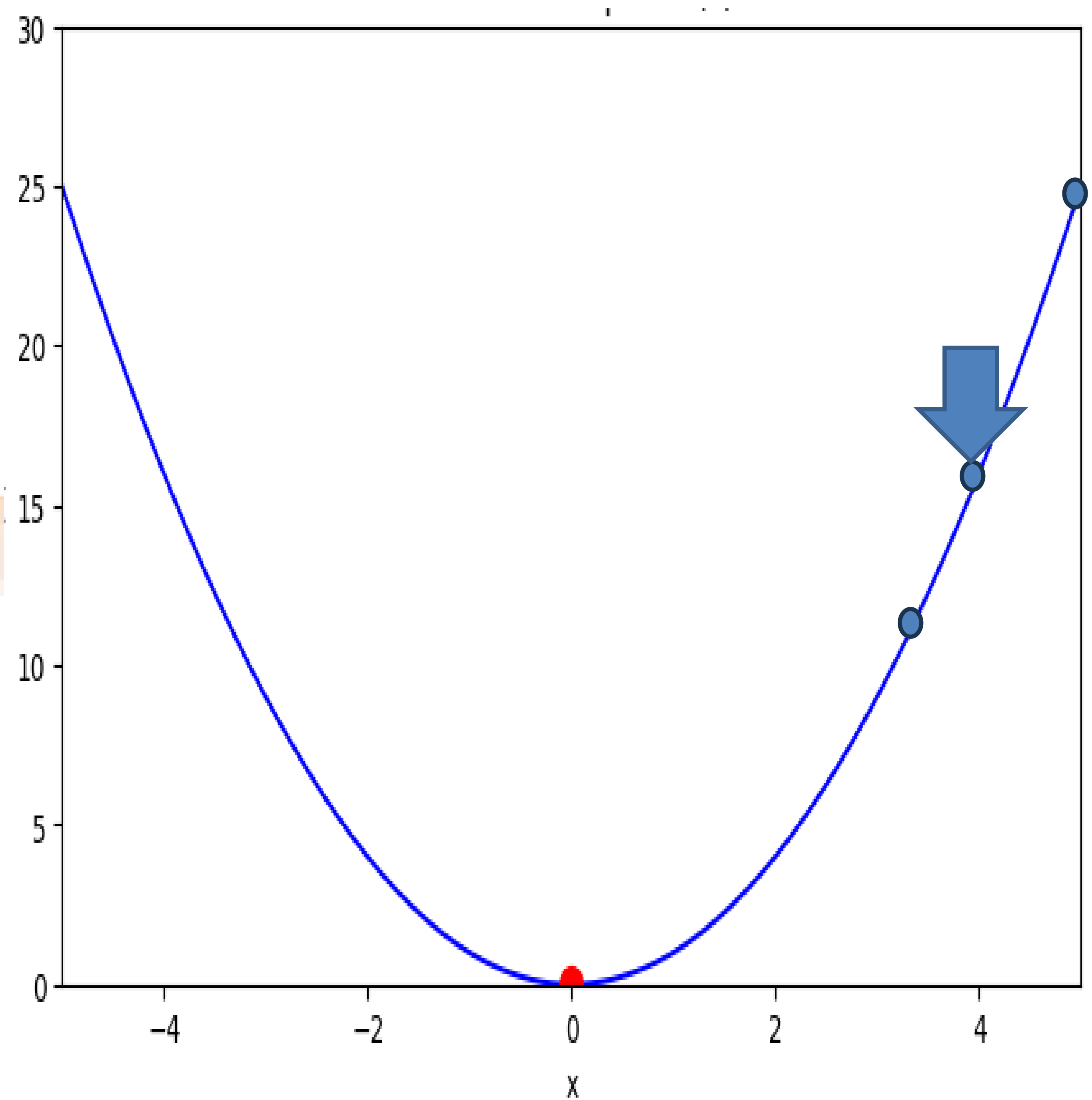
Iteration 1 :

Gradient = 10

Step Size = 1.0

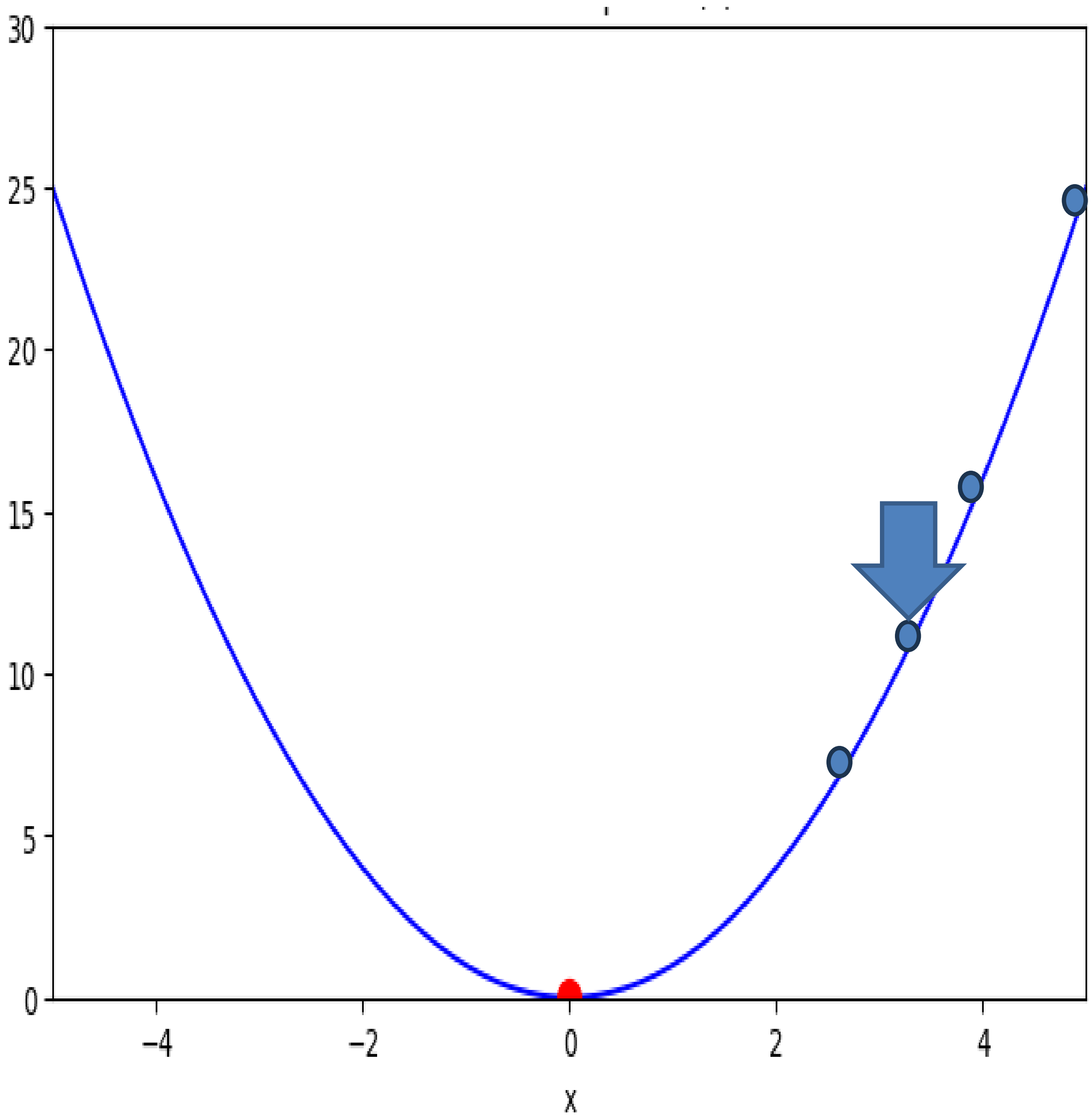
New X_value = 4

Learning Rate =0.1



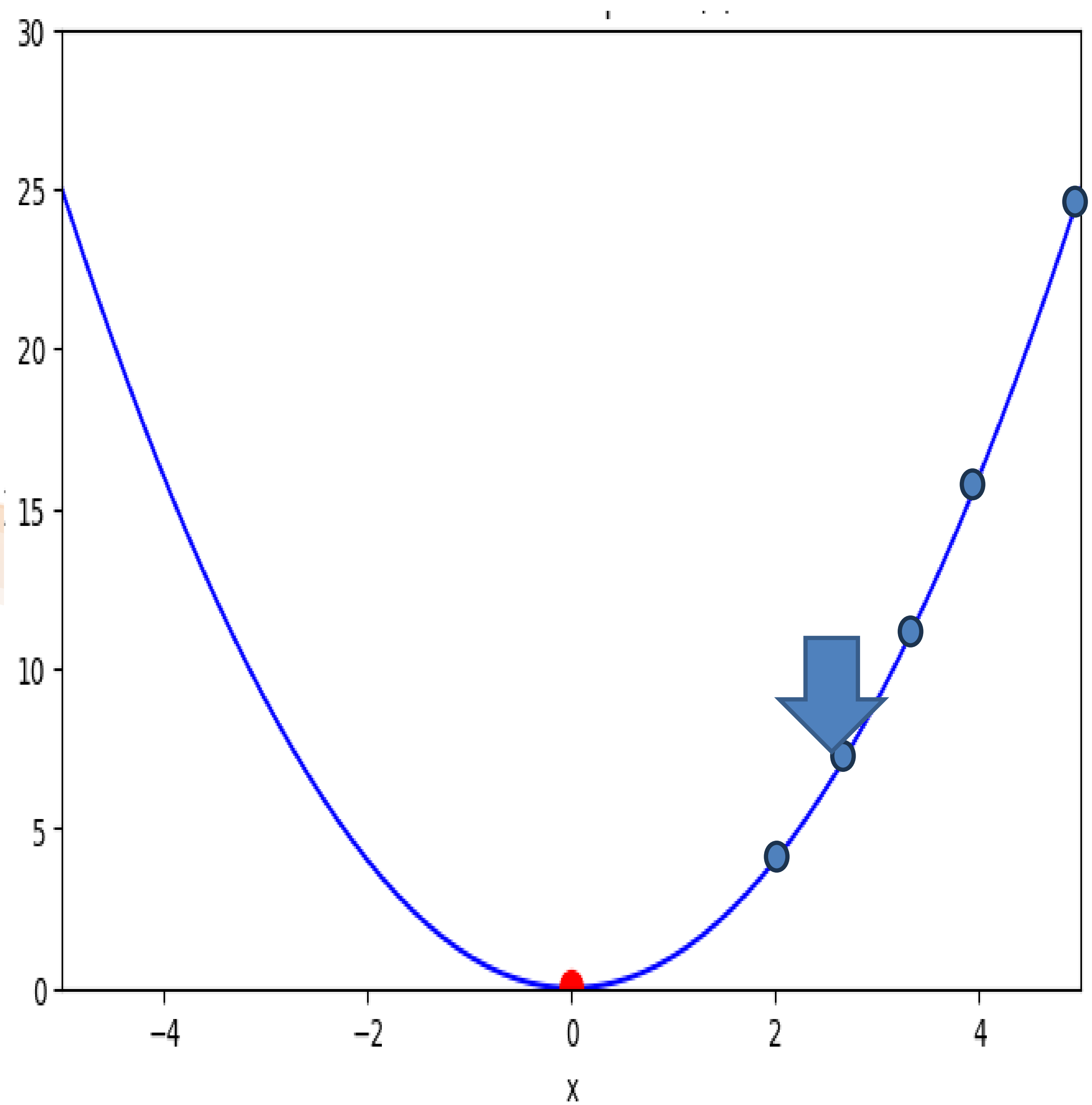
Iteration 1 : Step Size = 1.0

Iteration 2 : Step Size = 0.8



Learning Rate =0.1

Iteration 1 :	Step Size = 1.0
Iteration 2 :	Step Size = 0.8
Iteration 3 :	Step Size = 0.64



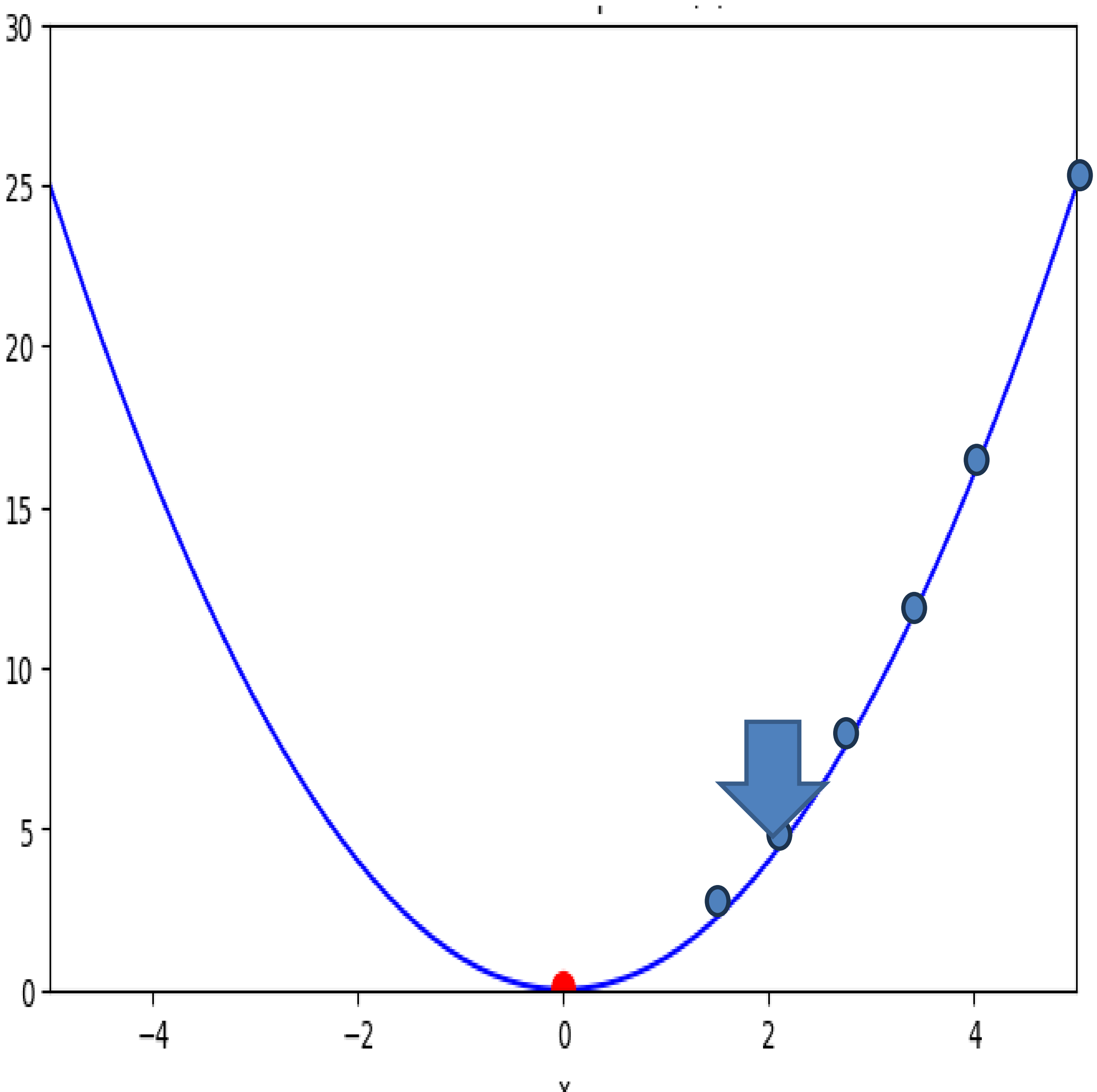
Learning Rate =0.1

Iteration 1 : Step Size = 1.0

Iteration 2 : Step Size = 0.8

Iteration 3 : Step Size = 0.64

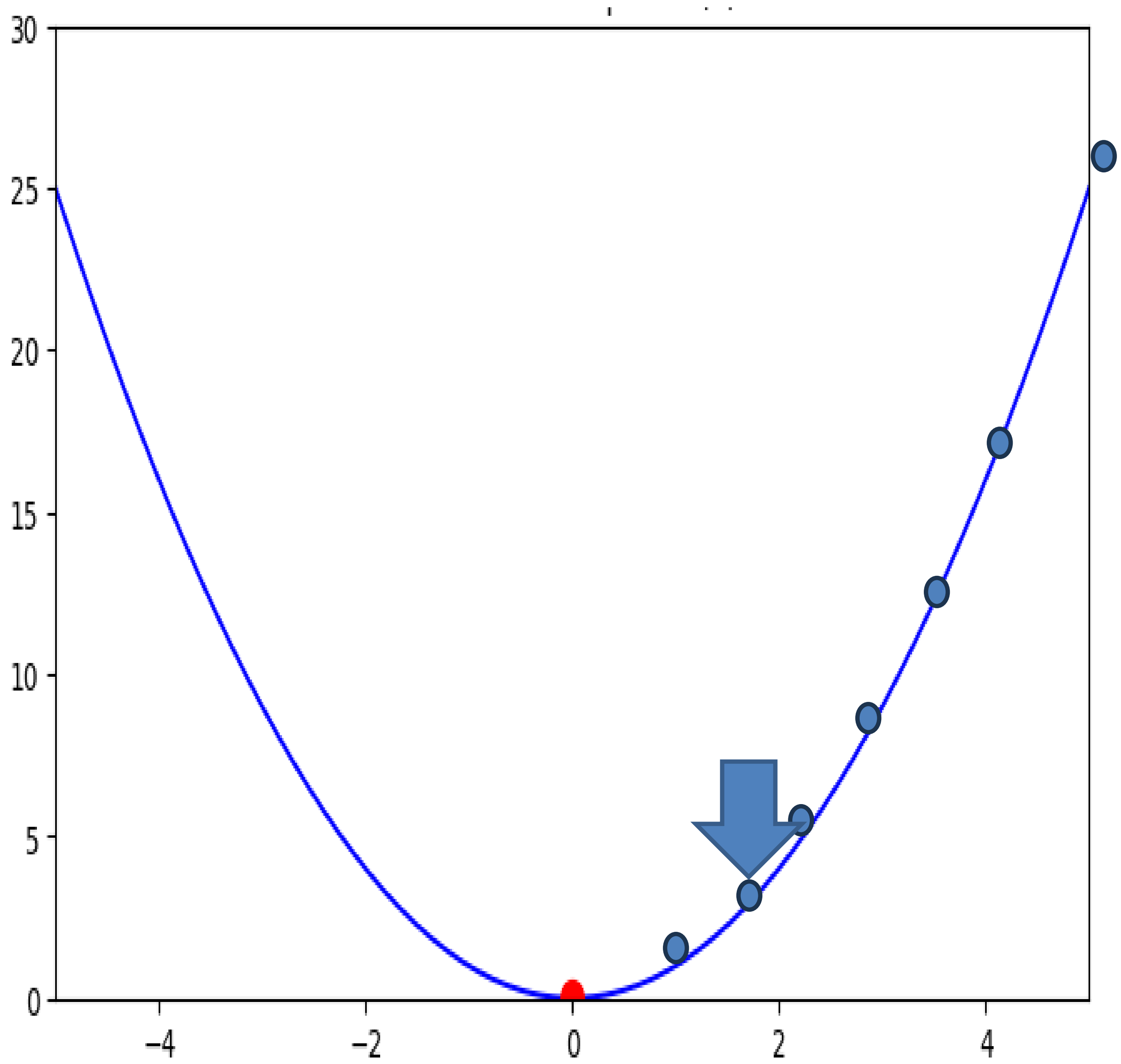
Iteration 4 : Step Size = 0.512



Learning Rate =0.1

Iteration 1 :	Step Size = 1.0
Iteration 2 :	Step Size = 0.8
Iteration 3 :	Step Size = 0.64
Iteration 4 :	Step Size = 0.512
Iteration 5 :	Step Size = 0.4096

Learning Rate =0.1



Iteration 1 : Step Size = 1.0

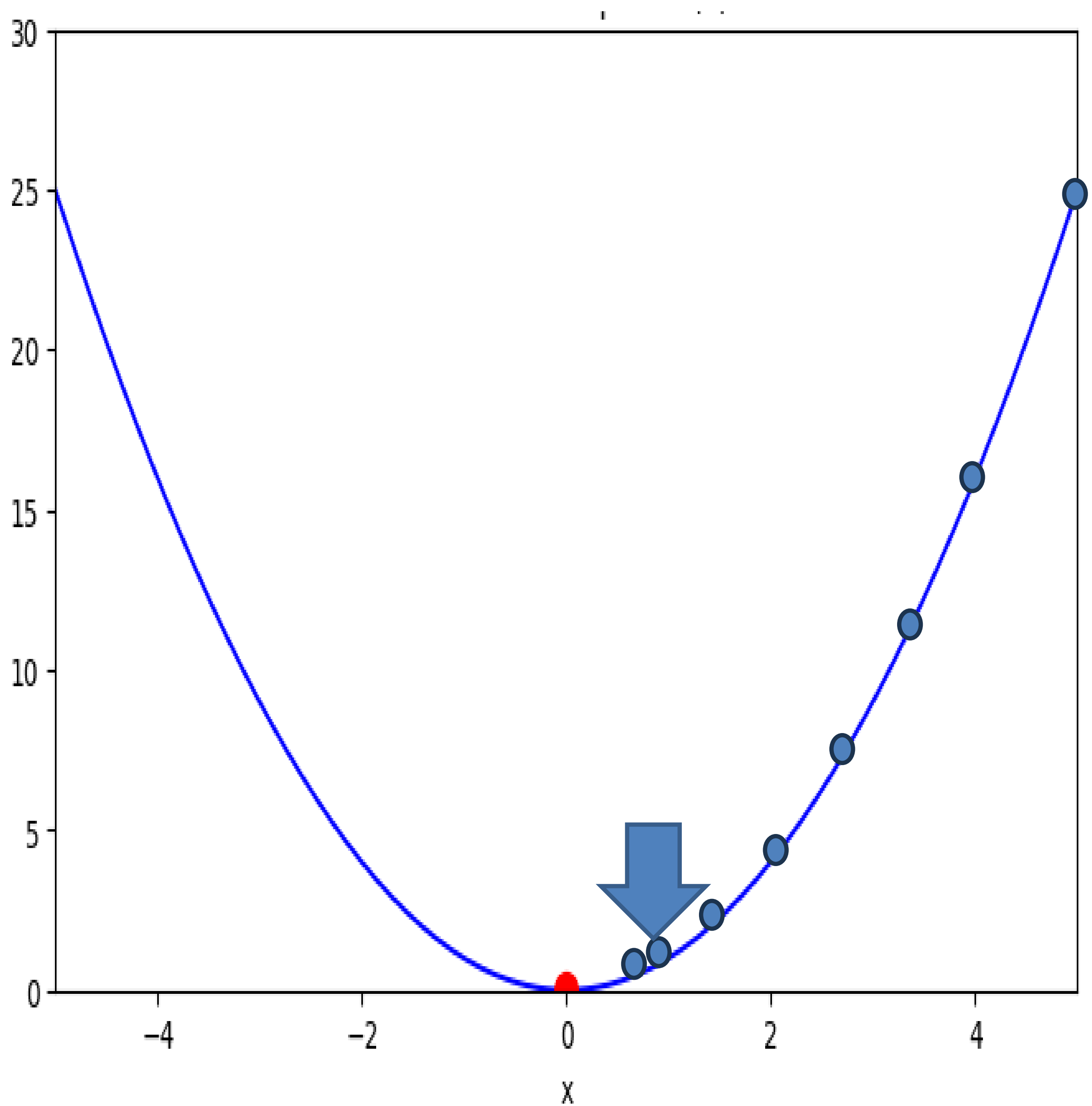
Iteration 2 : Step Size = 0.8

Iteration 3 : Step Size = 0.64

Iteration 4 : Step Size = 0.512

Iteration 5 : Step Size. = 0.4096

Iteration 6 : Step Size = 0.3277



Learning Rate =0.1

Iteration 1 : Step Size = 1.0

Iteration 2 : Step Size = 0.8

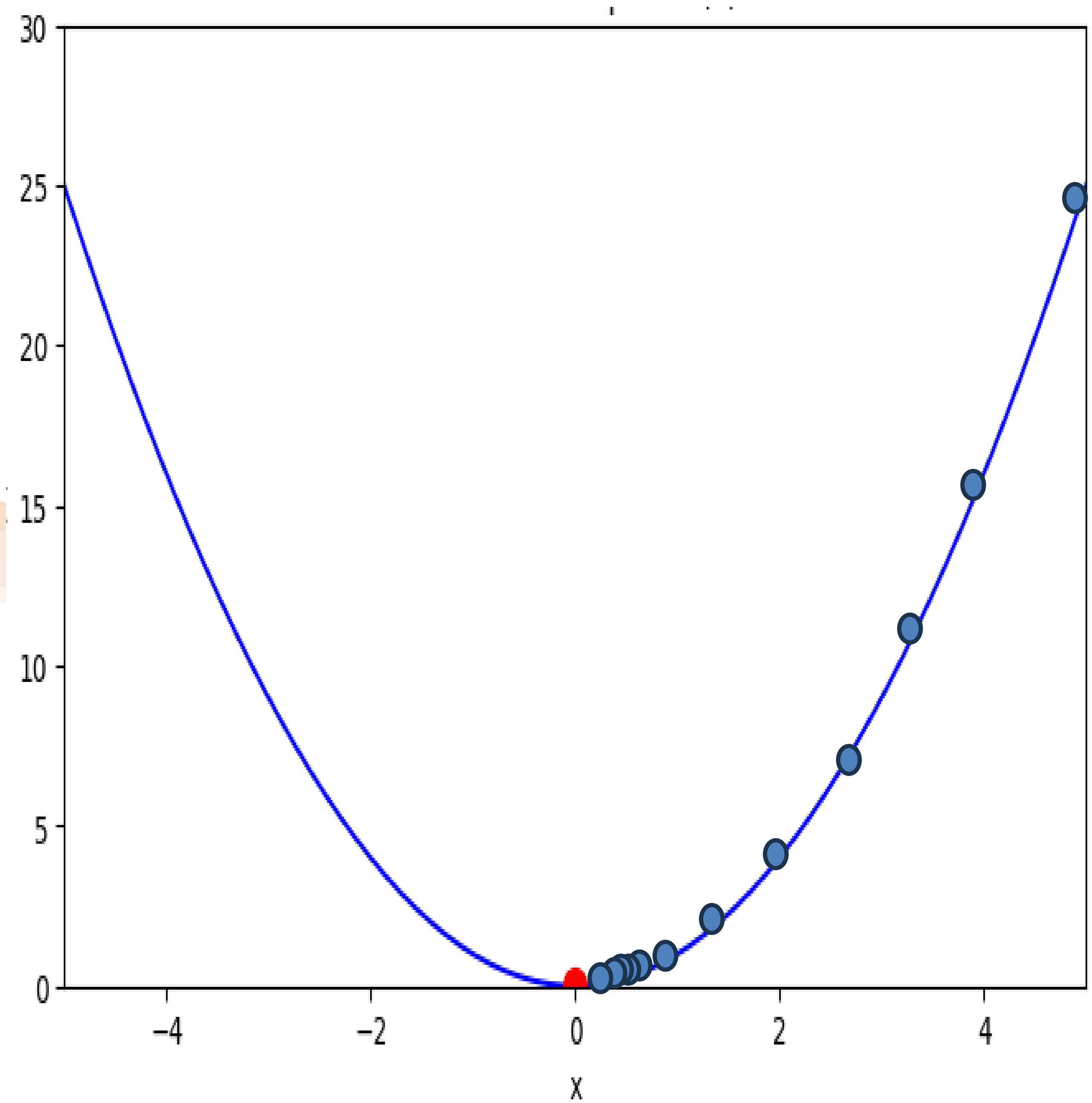
Iteration 3 : Step Size = 0.64

Iteration 4 : Step Size = 0.512

Iteration 5 : Step Size. = 0.4096

Iteration 6 : Step Size = 0.3277

Iteration 7 : Step size. = 0.2621



Learning Rate =0.1

Iteration 1 : Step Size = 1.0

Iteration 2 : Step Size = 0.8

Iteration 3 : Step Size = 0.64

Iteration 4 : Step Size = 0.512

Iteration 5 : Step Size. = 0.4096

Iteration 6 : Step Size = 0.3277

Iteration 7 : Step Size. = 0.2621

Iteration 8 : Step size = 0.2621

Iteration 10 : Step Size = 0.1342

Iteration 20 : Step Size = 0.0144

Iteration 50 : Step Size = 0.000017

Sl.No	x_1	x_2	y
1	1	2	5
2	2	1	6
3	3	3	10
4	4	5	13
5	5	4	14
6	6	6	17

Let's Work On This Dataset

Calculations of Gradient Descent

- Initial Parameters:

$$w_1 = 0,\ w_2 = 0,\ b = 0$$

- Learning Rate:

$$\eta = 0.01$$

Step 1: Initial Predictions:

For each data point, the prediction $\hat{y}_i = w_1 \cdot x_1^{(i)} + w_2 \cdot x_2^{(i)} + b = 0$

Thus, $\hat{y}_i = 0$ for each row initially.

Step 2: Calculate Gradients for w_1, w_2, and b

Using $n = 6$:

1. *Gradient*

$$\frac{\partial L}{\partial w_1} = -\frac{2}{6}\sum_{i=1}^{6} x_1^{(i)} \cdot (y_i - \hat{y}_i)$$

$$= -\frac{2}{6}(1 \cdot 5 + 2 \cdot 6 + 3 \cdot 10 + 4 \cdot 13 + 5 \cdot 14 + 6 \cdot 17)$$

$$\frac{\partial L}{\partial w_1} = -90.33$$

Similarly, $\frac{\partial L}{\partial w_2} = -89.67$ and $\frac{\partial L}{\partial b} = -21.67$

Calculations of Gradient Descent

Step 3: Update Parameters Using Gradients:

1. *Updated* w_1:

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0 + 0.01 \times 90.33 = 0.9033$$

2. *Updated* w_2 :

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0 + 0.01 \times 89.67 = 0.8967$$

3. *Updated* b:

$$b = b - \eta \cdot \frac{\partial L}{\partial b} = 0 + 0.01 \times 21.67 = 0.2167$$

Stochastic Gradient Descent

- *Stochastic Gradient Descent (SGD) updates model parameters after processing each individual data point, allowing for faster updates and more frequent adjustments.*
- *This approach introduces randomness, which can help escape local minima, making it particularly useful for complex, non-convex loss surfaces.*
- *While faster than traditional methods, SGD may exhibit noisy convergence, requiring careful tuning of hyperparameters like the learning rate.*
- *Commonly used for large datasets, SGD often yields quicker results compared to Batch Gradient Descent.*

Calculations of Stochastic Gradient Descent

- Initial Parameters:

$$w_1 = 0,\ w_2 = 0,\ b = 0$$

- Learning Rate:

$$\eta = 0.01$$

1. First Data Point: $\left(x_1^{(1)}, x_2^{(1)}, y^{(1)}\right) = (1,2,5)$

2. ***Prediction:***

$$\hat{y}_1 = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b = 0 \cdot 1 + 0 \cdot 2 + 0 \ = 0$$

2. ***Calculate Gradients:***

$$\frac{\partial L}{\partial w_1} = -2 \cdot x_1^{(1)} \cdot \left(y^{(1)} - \hat{y}_1\right) = -2 \cdot 1 \cdot (5 - 0) = -10$$

$$\frac{\partial L}{\partial w_2} = -2 \cdot x_2^{(1)} \cdot \left(y^{(1)} - \hat{y}_1\right) = -2 \cdot 2 \cdot (5 - 0) = -20$$

$$\frac{\partial L}{\partial b} = -2 \cdot \left(y^{(1)} - \hat{y}_1\right) = -2 \cdot (5 - 0) = -10$$

3. ***Update Parameters:***

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0 + 0.01 \cdot 10 = 0.1$$

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0 + 0.01 \cdot 20 = 0.2$$

$$b = b - \eta \cdot \frac{\partial L}{\partial b} = 0 + 0.01 \cdot 10 = 0.1$$

Calculations of Stochastic Gradient Descent

Second Data Point: $\left(x_1^{(2)}, x_2^{(2)}, y^{(2)}\right) = (2,1,6)$

Prediction:

$$\hat{y}_2 = w_1 \cdot x_1^{(2)} + w_2 \cdot x_2^{(2)} + b = 0.1 \cdot 2 + 0.2 \cdot 1 + 0.1 = 0.5$$

Calculate Gradients:

$$\frac{\partial \mathrm{L}}{\partial w_1} = -2 \cdot x_1^{(2)} \cdot \left(y^{(2)} - \hat{y}_2\right) = -2 \cdot 2 \cdot (6 - 0.5) \quad = \quad -22$$

$$\frac{\partial L}{\partial w_2} = -2 \cdot x_2^{(2)} \cdot \left(y^{(2)} - \hat{y}_2\right) = -2 \cdot 1 \cdot (6 - 0.5). \quad = \quad -11$$

$$\frac{\partial L}{\partial b} = -2 \cdot \left(y^{(2)} - \hat{y}_2\right) = -2 \cdot (6 - 0.5) \quad = \quad -11$$

Update Parameters:

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} = 0.1 + 0.01 \cdot 22 = . \ 0.32$$

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} = 0.2 + 0.01 \cdot 11 = \quad 0.31$$

$$b = \ b - \eta \cdot \frac{\partial L}{\partial b} = \ 0.1 + 0.01 \cdot 11 = \quad 0.21$$

Default Values of learning rates in different libraries in SGD

TensorFlow/Keras 0.01

PyTorch 0.01

Scikit-Learn 0.001

Gradient Descent Vs Stochastic Gradient Descent

	Gradient Descent (GD)	Stochastic Gradient Descent (SGD)
Data Used	All data points at once	One data point at a time
Speed	Slower	Faster
Noise Level	Less noise, smoother	More noise, fluctuates more
Best For	Small datasets	Large datasets

SGD with Momentum

Momentum helps accelerate SGD by accumulating a velocity term, allows for ***faster convergence.***

$$V_t = \gamma \times V_{t-1} + \eta \times g_t$$
$$\theta_{t+1} = \theta_t - V_t$$

Momentum factor γ typically around 0.9.

Nesterov Accelerated Gradient (NAG)

NAG is an improvement over Momentum, where it "looks ahead" by computing the gradient at an approximate future position.

$$V_t = \gamma \times V_{t-1} + \eta \times (\theta_t + \gamma \times V_{t-1})$$

$$\theta_{t+1} = \theta_t - \eta \times (\theta_t + \gamma \times V_{t-1})$$

Adagrad

Adagrad adapts the learning rate for each parameter based on the cumulative sum of past squared gradients

$$V_t = V_{t-1} + (g_t)^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t + \epsilon}} g_t$$

RMSProp

RMSprop modifies Adagrad by introducing a moving average of squared gradients rather than a cumulative sum.

$$V_t = \rho \times V_{t-1} + (1 - \rho) \times (g_t)^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t + \epsilon}} g_t$$

ADAM Optimizer

Adaptive Moment Estimation

Adaptive Learning Rates: Adjusts the learning rate for each parameter based on how frequently and strongly each is updated, helping in noisy and sparse data environments.

Combination of Adagrad and RMSprop: Uses *Adagrad* for adaptive learning rates and *RMSprop* for scaling rates based on recent gradients, making it both stable and responsive.

Bias Correction: Corrects initial bias in moment estimates to ensure accurate updates early in training.

Widely Used in Deep Learning: Often the go-to optimizer for deep learning models due to its fast convergence and robustness across diverse datasets.

Mathematical Calculations

Dataset :

X	1	2	3
Y	2	4	6

Goal :

We want to fit a line $y = \theta_0 + \theta_1 x$ to the data.

Initial Parameters :

- $\theta_0 = 0$
- $\theta_1 = 0$
- Learning rate $\eta = 0.1$

Calculations :

$$\hat{y} = \theta_0 + \theta_1 \cdot x = 0 + 0 \cdot 1 = 0$$

$$\text{Error(L)} = \hat{y} - y = 0 - 2 = -2$$

Weights updation using ADAM

We have , $\frac{\partial L}{\partial \theta_0} = -4$ and $\frac{\partial L}{\partial \theta_1} = -4$

Let ,

- $m_{\theta_0,0} = 0$ & $V_{\theta_{0,0}} = 0$
- $m_{\theta_0,1} = 0$ & $V_{\theta_{0,1}} = 0$
- $\beta_1 = 0.9,$ $\beta_2 = 0.999,$ & $\epsilon = 10^{-8}$

✓ ***First momentum:***

$$m_{\theta_0,1} = \beta_1 \cdot m_{\theta_0,0} + (1-\beta_1).\frac{\partial L}{\partial \theta_0}$$
$$= 0.9 \cdot 0 + (1-0.9) \cdot (-4)$$
$$= -0.4$$

✓ ***Second momentum:***

$$V_{\theta_0,1} = \beta_2 \cdot V_{\theta_0,0} + (1-\beta_2) \cdot (\frac{\partial L}{\partial \theta_0})^2$$
$$= 0.999 \cdot 0 + (1-0.999) \cdot (-4)^2$$
$$= 0.016$$

✓ ***First momentum:***

$$m_{\theta_1,1} = \beta_1 \cdot m_{\theta_1,0} + (1-\beta_1).\frac{\partial L}{\partial \theta_1}$$
$$= 0.9 \cdot 0 + (1-0.9) \cdot (-4)$$
$$= -0.4$$

✓ ***Second momentum:***

$$V_{\theta_1,1} = \beta_2 \cdot V_{\theta_1,0} + (1-\beta_2) \cdot (\frac{\partial L}{\partial \theta_1})^2$$
$$= 0.999 \cdot 0 + (1-0.999) \cdot (-4)^2$$
$$= 0.016$$

Bias Updation:

$$\hat{m}_{\theta_{0},1} = \frac{m_{\theta_{0},1}}{1-\beta_1^1}$$

$$= \frac{-0.4}{1-0.9}$$

$$\hat{m}_{\theta_{0},1} = -4$$

$$\hat{V}_{\theta_{0},1} = \frac{V_{\theta_{0},0}}{1-\beta_2^1}$$

$$= \frac{0.016}{1-0.999}$$

$$\hat{V}_{\theta_{0},1} = 16$$

Weights updation:

$$\theta_{0,1} = \theta_{0,0} - \frac{\eta}{\sqrt{\hat{V}_{\theta_{0},1}+\epsilon}} . \hat{m}_{\theta_{0},1}$$

$$\theta_{0,1} = 0 - \frac{0.1}{\sqrt{16}+10^{-8}} \cdot (-4)$$

$$\theta_{0,1} = 0.1$$

Bias Updation:

$$\hat{m}_{\theta_{1,1}} = \frac{m_{\theta_{0},1}}{1-\beta_1^1}$$

$$= \frac{-0.4}{1-0.9}$$

$$\hat{m}_{\theta_{1,1}} = -4$$

$$\hat{V}_{\theta_{1},1} = \frac{V_{\theta_{1},0}}{1-\beta_2^1}$$

$$= \frac{0.016}{1-0.999}$$

$$\hat{V}_{\theta_{1},1} = 16$$

Weights updation:

$$\theta_{1,1} = \theta_{1,0} - \frac{\eta}{\sqrt{\hat{V}_{\theta_{1},1}+\epsilon}} \cdot \hat{m}_{\theta_{1},1}$$

$$\theta_{1,1} = 0 - \frac{0.1}{\sqrt{16}+10^{-8}} \cdot (-4)$$

$$\theta_{1,1} = 0.1$$

SGD VS ADAM

SGD Parameter Updates:

Iteration 1: $\theta_0 = 0.4000$, $\theta_1 = 0.4000$
Iteration 2: $\theta_0 = 0.9600$, $\theta_1 = 1.5200$
Iteration 3: $\theta_0 = 1.0560$, $\theta_1 = 1.8080$
Iteration 4: $\theta_0 = 0.8832$, $\theta_1 = 1.6352$
Iteration 5: $\theta_0 = 0.8525$, $\theta_1 = 1.5738$
Iteration 6: $\theta_0 = 0.9377$, $\theta_1 = 1.8295$
Iteration 7: $\theta_0 = 0.7843$, $\theta_1 = 1.6761$
Iteration 8: $\theta_0 = 0.7570$, $\theta_1 = 1.6215$
Iteration 9: $\theta_0 = 0.8327$, $\theta_1 = 1.8486$

Adam Parameter Updates:

Iteration 1: $\theta_0 = 0.1000$, $\theta_1 = 0.1000$
Iteration 2: $\theta_0 = 0.2308$, $\theta_1 = 0.2200$
Iteration 3: $\theta_0 = 0.3828$, $\theta_1 = 0.3578$
Iteration 4: $\theta_0 = 0.4968$, $\theta_1 = 0.4553$
Iteration 5: $\theta_0 = 0.6194$, $\theta_1 = 0.5609$
Iteration 6: $\theta_0 = 0.7515$, $\theta_1 = 0.6809$
Iteration 7: $\theta_0 = 0.8579$, $\theta_1 = 0.7753$
Iteration 8: $\theta_0 = 0.9642$, $\theta_1 = 0.8690$
Iteration 9: $\theta_0 = 1.0740$, $\theta_1 = 0.9694$

Observation: Adam makes a more conservative update due to the influence of the moments, whereas SGD makes a larger step.

TABLE I: MNIST

Epoch	SGD	Momen tum	Ada grad	Ada delta	RMS Prop	Adam
Epoch 000	0.10	0.20	0.18	0.06	0.14	0.16
Epoch 100	0.54	0.90	0.84	0.04	0.96	0.98
Epoch 200	0.74	0.92	0.94	0.16	**1.00**	**1.00**
Epoch 300	0.86	**0.96**	0.94	0.18	1.00	1.00
Epoch 400	0.80	0.96	0.96	0.12	0.98	0.98
Epoch 500	0.82	0.94	0.96	0.28	0.98	0.92
Epoch 600	0.88	0.96	0.96	0.32	1.00	1.00
Epoch 700	0.86	0.92	0.94	**0.46**	1.00	1.00
Epoch 800	**0.96**	0.98	**1.00**	0.40	1.00	1.00
Epoch 900	0.88	0.88	0.96	0.44	1.00	1.00

Comparision of Various Optimisers on MNIST dataset

Comparision of Various Optimisers on FashionMNIST Dataset

TABLE II: FashionMNIST

Epoch	SGD	Momen tum	Ada grad	Ada delta	RMS Prop	Adam
Epoch 000	0.12	0.22	0.20	0.08	0.20	0.22
Epoch 100	0.56	0.86	0.80	0.16	0.90	0.96
Epoch 200	0.84	0.90	0.90	0.10	0.96	0.98
Epoch 300	0.98	0.94	0.92	0.10	0.98	0.98
Epoch 400	0.88	0.90	**0.96**	0.20	**1.00**	**1.00**
Epoch 500	**0.94**	0.96	0.86	0.18	0.98	0.96
Epoch 600	0.92	**1.00**	0.90	0.22	0.98	1.00
Epoch 700	0.94	0.98	0.96	0.32	1.00	0.98
Epoch 800	0.90	0.94	0.96	**0.38**	1.00	1.00
Epoch 900	0.96	0.94	0.94	0.36	1.00	1.00

Comparision of Various Optimisers on Cifar10 Dataset

TABLE III: Cifar10

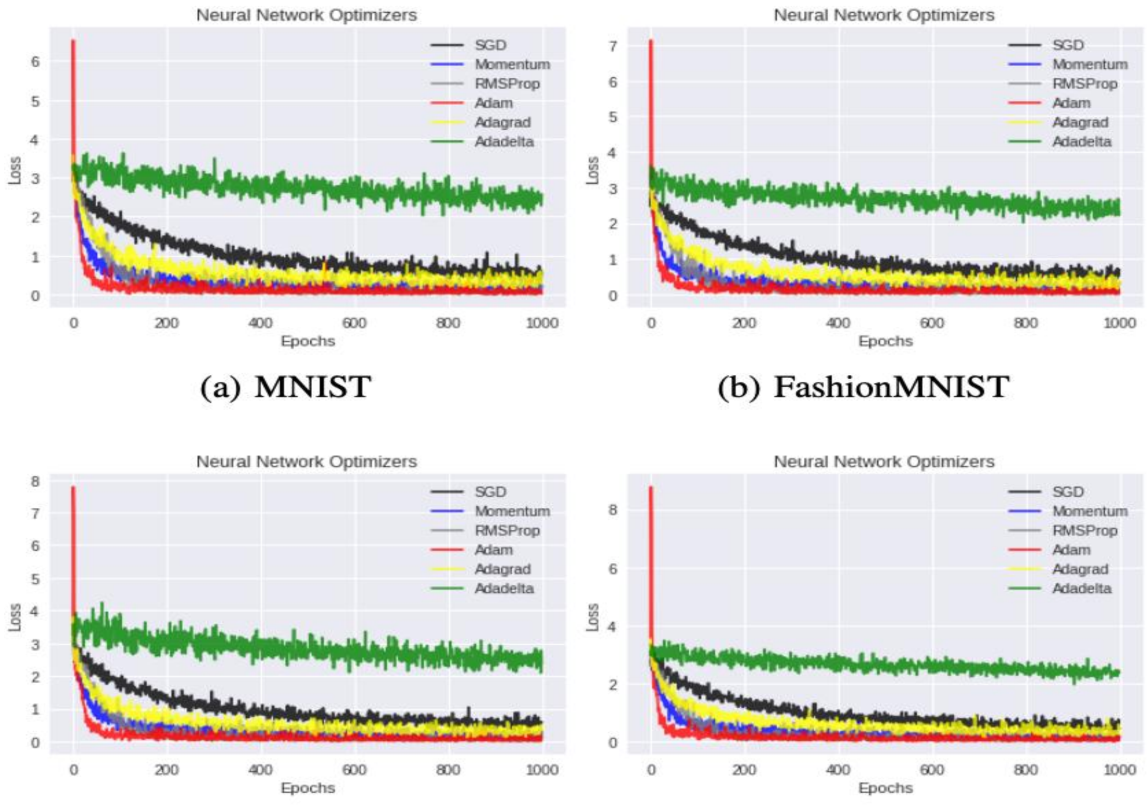
Epoch	SGD	Momen tum	Ada grad	Ada delta	RMS Prop	Adam
Epoch 000	0.12	0.26	0.14	0.06	0.18	0.16
Epoch 100	0.80	0.88	0.92	0.18	0.98	0.96
Epoch 200	0.76	0.92	0.96	0.14	**1.00**	0.98
Epoch 300	0.84	0.94	0.90	0.18	1.00	0.96
Epoch 400	0.84	0.92	**0.98**	0.20	1.00	**1.00**
Epoch 500	**0.90**	0.94	0.92	**0.34**	1.00	0.94
Epoch 600	0.90	**0.98**	0.88	0.28	1.00	0.98
Epoch 700	0.88	0.98	0.94	0.26	1.00	0.94
Epoch 800	0.82	0.96	0.92	0.40	1.00	0.98
Epoch 900	0.90	0.96	0.96	0.34	1.00	0.98

TABLE IV: Cifar100

Epoch	SGD	Momen tum	Ada grad	Ada delta	RMS Prop	Adam
Epoch 000	0.12	0.08	0.22	0.12	0.10	0.36
Epoch 100	0.62	0.90	0.78	0.10	0.84	**1.00**
Epoch 200	0.72	**0.98**	0.88	0.08	**1.00**	0.98
Epoch 300	0.84	0.98	0.94	0.16	0.98	0.98
Epoch 400	0.80	0.92	0.82	0.16	1.00	0.98
Epoch 500	0.86	0.92	**0.96**	0.12	1.00	0.98
Epoch 600	0.84	0.96	0.88	0.16	0.98	0.98
Epoch 700	**0.90**	0.92	0.96	0.20	1.00	0.98
Epoch 800	0.94	0.96	0.96	0.30	1.00	0.94
Epoch 900	0.84	0.96	0.96	**0.32**	1.00	0.98

Comparision of Various Optimisers on Cifar100 Dataset

Comparision of Various Optimisers on Various Dataset



(a) MNIST (b) FashionMNIST

(c) Cifar10 (d) Cifar100

TABLE V: Comparison of Results

Optimization Algorithm	MNIST	FashionMNIST	Cifar10	Cifar100
SGD	0.9086	0.9108	0.9089	0.9151
Momentum	0.9663	0.9666	0.9643	0.9650
Adagrad	0.9394	0.9406	0.9386	0.9334
Adadelta	0.4524	0.4346	0.4422	0.3491
RMSProp	0.9823	0.9838	0.9773	0.9795
Adam	**0.9826**	**0.9853**	**0.9855**	**0.9842**

Conclusion

Different optimizers offer trade-offs in speed, stability, and performance. Gradient Descent is slow but stable, while SGD is faster with more noise. Adaptive optimizers like Adam adjust learning rates dynamically, improving convergence. The best optimizer depends on the dataset and requires proper tuning for optimal results.

REFERENCES

1. Humera Shaziya , Raniah Zaheer,”**A Study of the Optimization Algorithms in Deep Learning”,** International Conference on Inventive Systems and Control (ICISC 2019) IEEE Xplore Part Number: CFP19J06-ART; ISBN: 978-1-5386-3950-4

2. S. Ruder**, “An overview of gradient descent optimization algorithms,”** arXiv preprint arXiv:1609.04747, 2016.

3. A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz, “**Sgd learns over-parameterized networks that provably generalize on linearly separable data,”** arXiv preprint arXiv:1710.10174, 2017.

4. C. Darken, J. Chang, and J. Moody, “**Learning rate schedules for faster stochastic gradient search,” in Neural Networks for Signal Processing** [1992] II., Proceedings of the 1992 IEEE-SP Workshop. IEEE, 1992, pp. 3–12.

5. H. Xiao, K. Rasul, and R. Vollgraf, “**Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms**,” arXiv preprint arXiv:1708.07747, 2017

THANK YOU