Project : IE 609 (Optimization Techniques)

Implementation of Variants of Gradient Descent

Group info:
1. Tirupathi Rao Chinnala (24M1516)
2. Ganesh Yegireddi (24M1518)
3. Siva Dogga (24M1528)
4. Trivikram Umanath (24M1535)
5. Sachin Pandey (24M1537)

Presentation Outline

1.Project Description

2.Introduction to Gradient Descent

3.Types of Gradient Descent

4. Analysis on Batch gradient descent

5.Analysis on Stochastic Gradient Descent

6.Analysis on Mini-batch Gradient Descent

7.Analysis on Momentum Gradient Descent

8. Analysis on Nesterov accelerated gradient

Project Description

Objective: To implement various gradient descent optimization algorithms from scratch using standard formulations

Compare their performance against in-built implementations provided by popular machine learning libraries.

Dataset Used: Life expectancy dataset

Evaluation Metrics and Criteria: Performance metrics (MSE), impact of learning rates & lambda(regularization rate); Convergence criteria (difference of current & previous MSE under a certain predefined limit)

Visualizations and Insights: Learning rate & regularization rate sensitivity analysis
Convergence Analysis- MSE vs Epochs (No. of iterations)
Performance insights comparisons between custom and in-built implementations

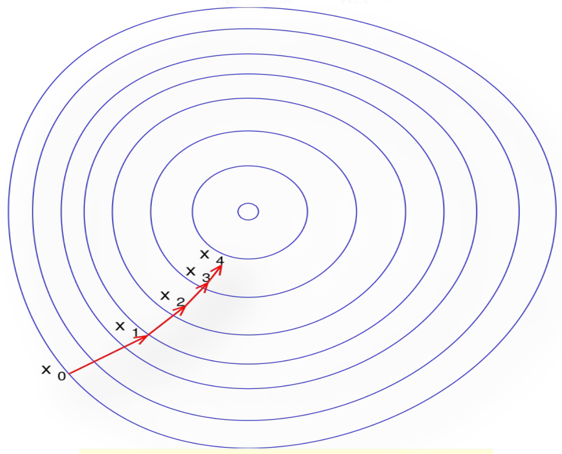
Tools and Frameworks: Python Libraries: NumPy, Pandas, Scikit-learn, Matplotlib

Gradient Descent

- A method for unconstrained mathematical optimization.
- It is a first order iterative algorithm for minimizing a differentiable multivariate function.
- Function being L-Lipschitz continuous prevents the algorithm from overshooting the minimum.
- useful in machine learning for minimizing the cost or loss function (like in linear regression).

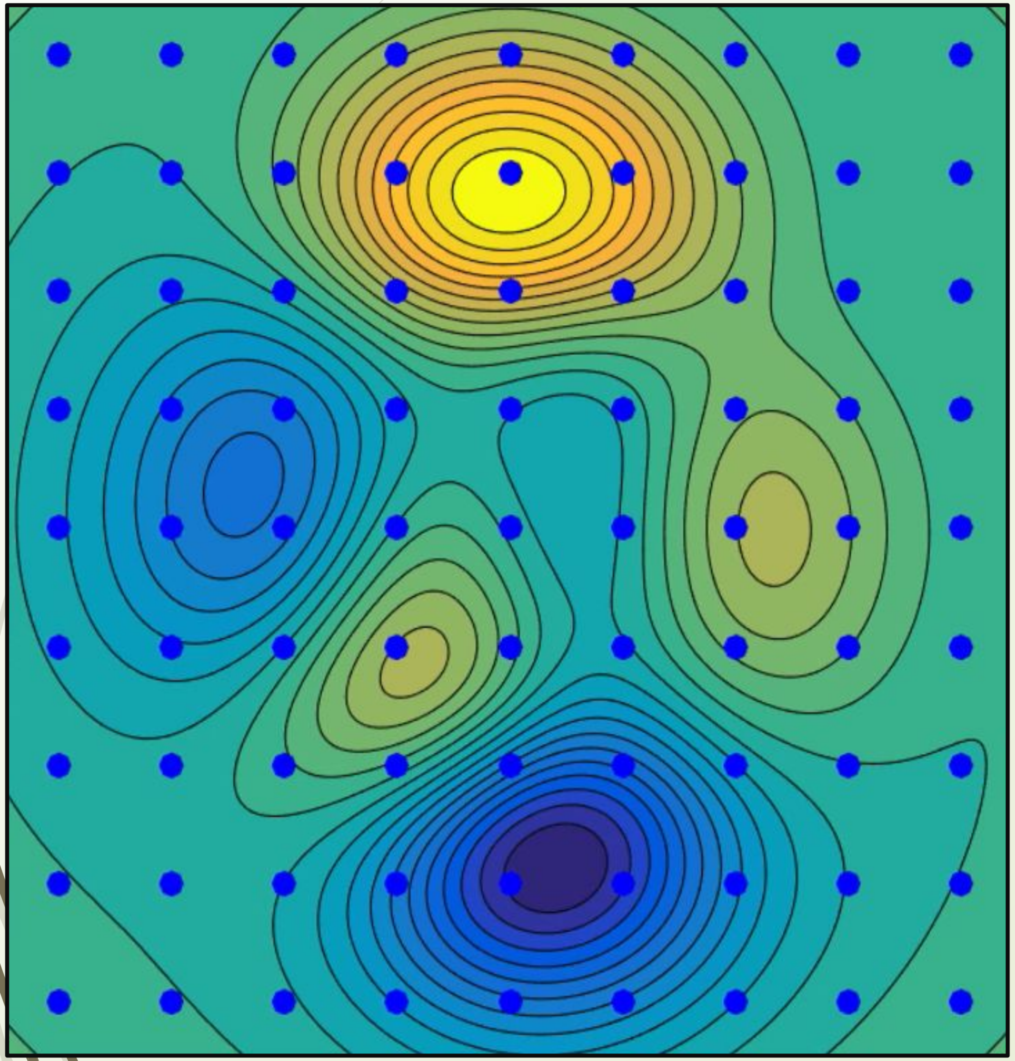
- **Idea**: Take repeated steps in the opposite direction of the gradient of the function at the current point (direction of steepest descent) till the stopping criteria. $\|\nabla f(x^{(k)})\| \leq \epsilon$

Choosing the step size
- slow convergence
- Overshoot and divergence



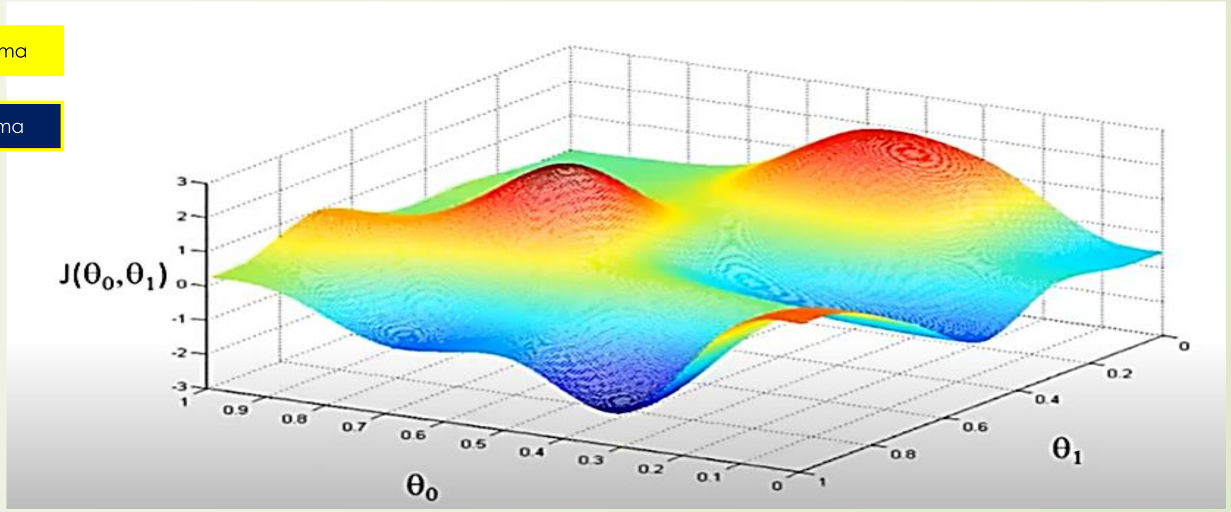
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n)$$

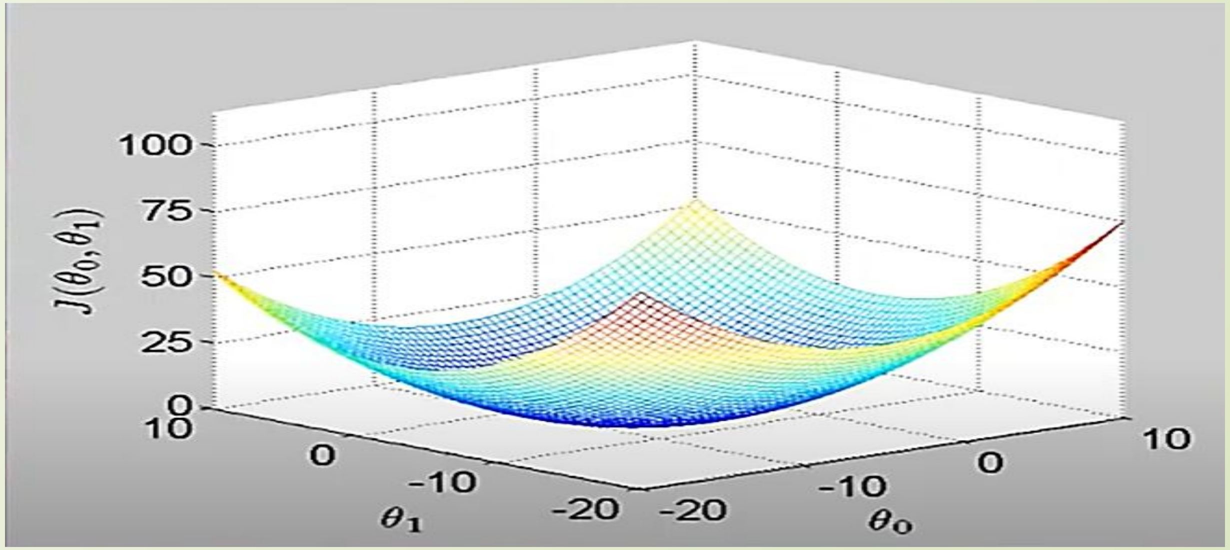
Loss Function – finding the minima



Maxima

Minima





Types of Gradient Descent

- Batch gradient descent
- Stochastic Gradient Descent (SGD)
- Mini-batch Gradient Descent
- Momentum Gradient Descent
- Nesterov Accelerated Gradient (NAG)

1. Analysis on Batch gradient descent (vanilla gradient descent)

Entire training dataset - used to compute the gradients of the cost function/Loss function with respect to the model parameters in each iteration.

Computationally expensive

Guarantees convergence -to a local minimum of the cost function

Mathematical Formulation:

$\theta = \theta - \alpha \cdot \nabla J(\theta)$

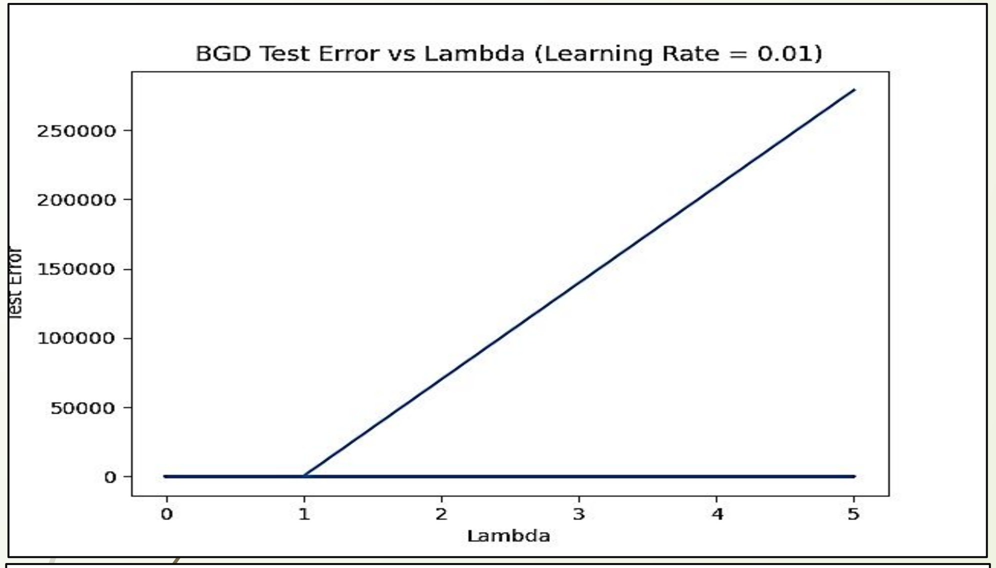
where:

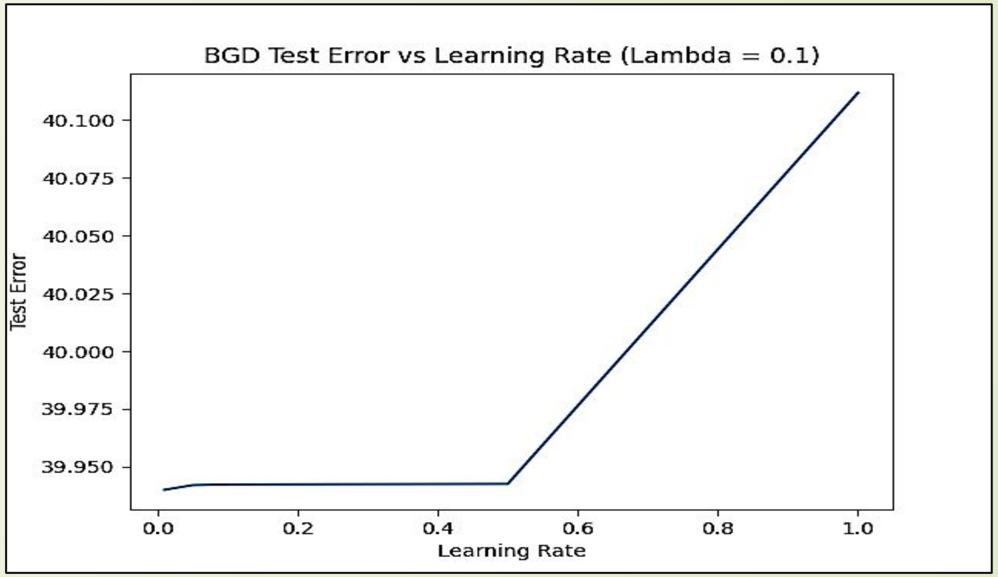
θ is the parameter vector

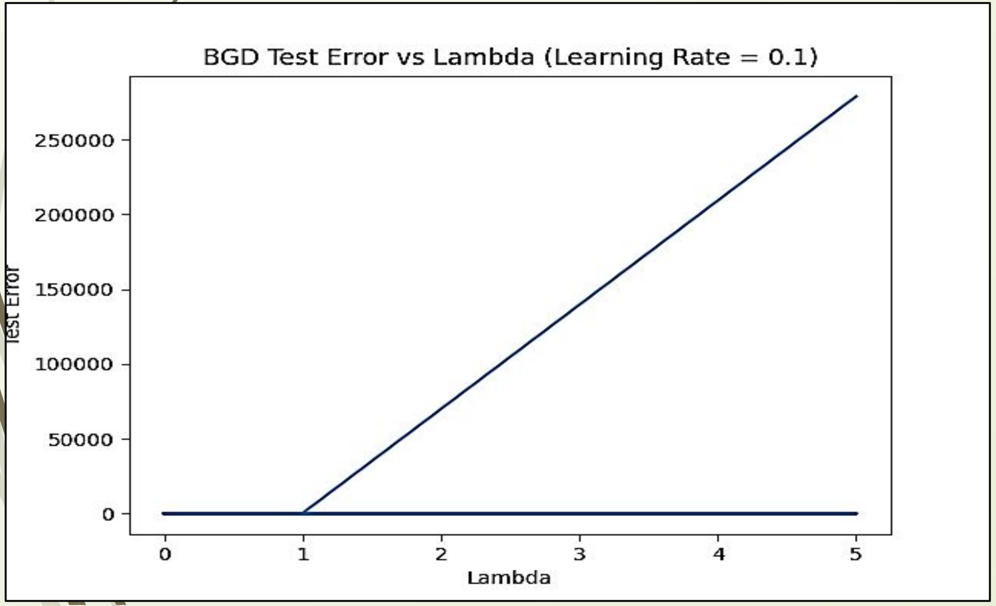
α is the learning rate

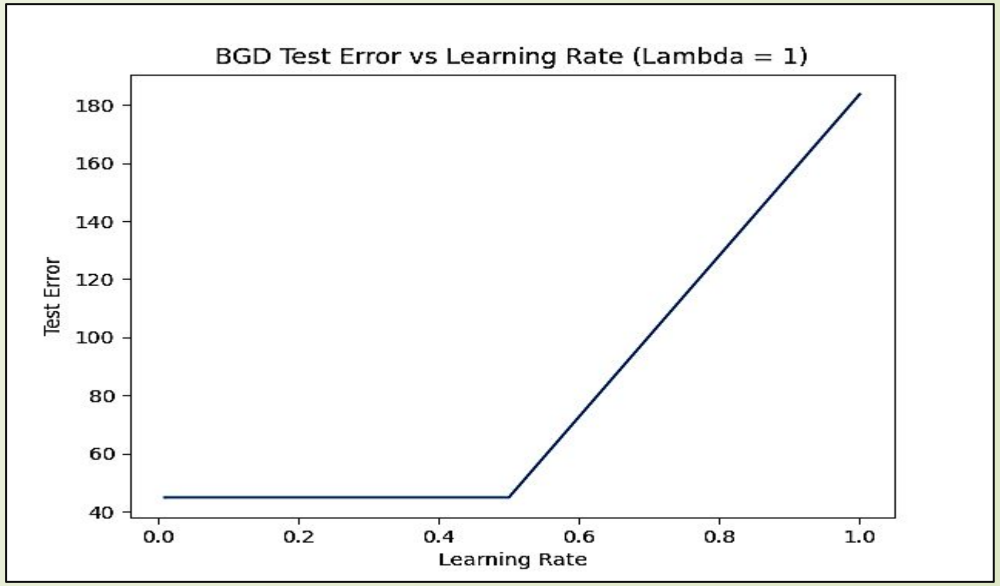
$\nabla J(\theta)$ is the gradient of the cost function J with respect to θ

1.1 Learning rate & regularization rate sensitivity analysis



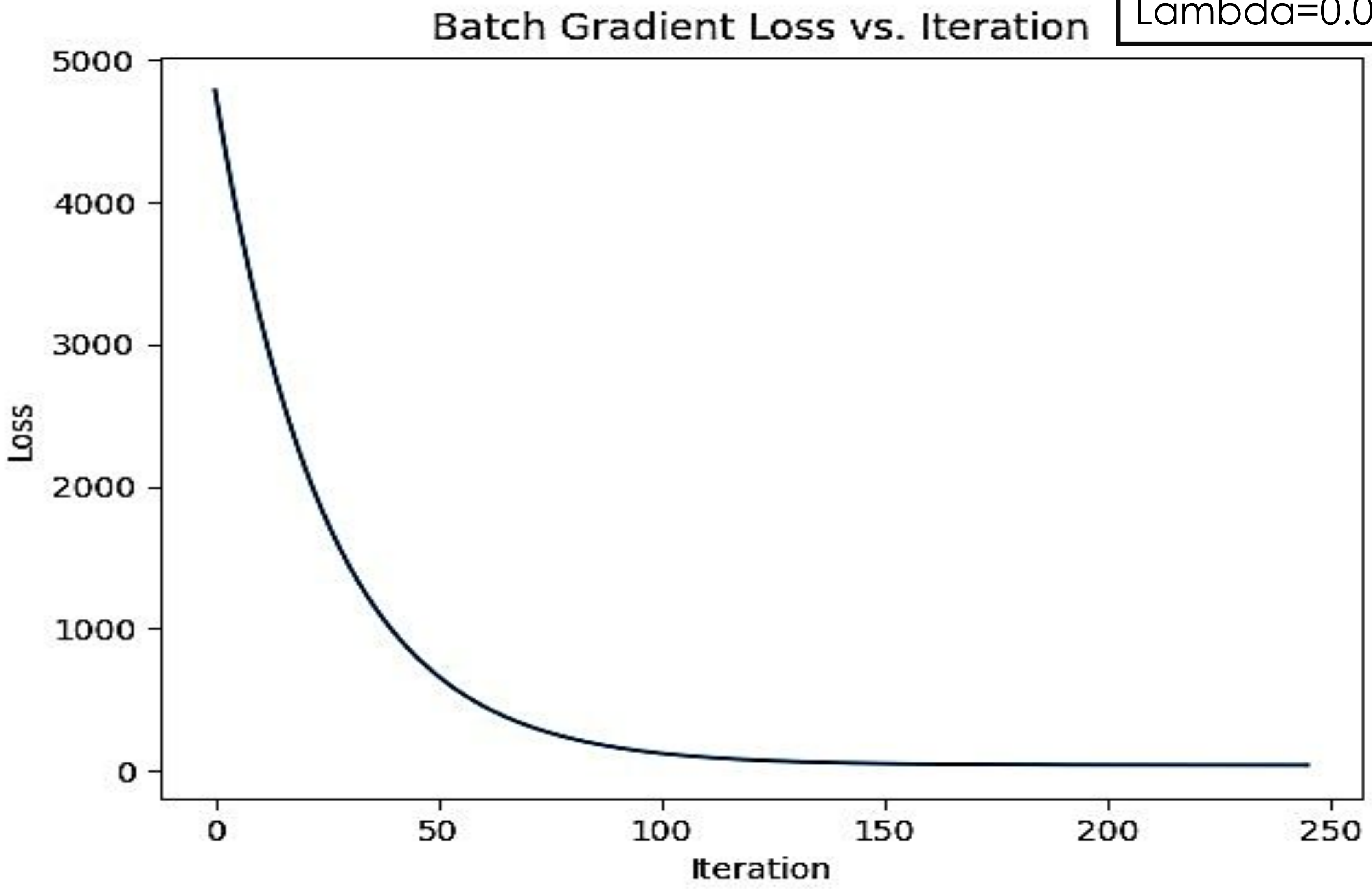






1.2 Convergence Analysis – Loss function vs Epochs

Learning rate= 0.01
Lambda=0.01



2. Analysis on Stochastic Gradient Descent (SGD)

Randomly selected training examples- used to update SGD parameters

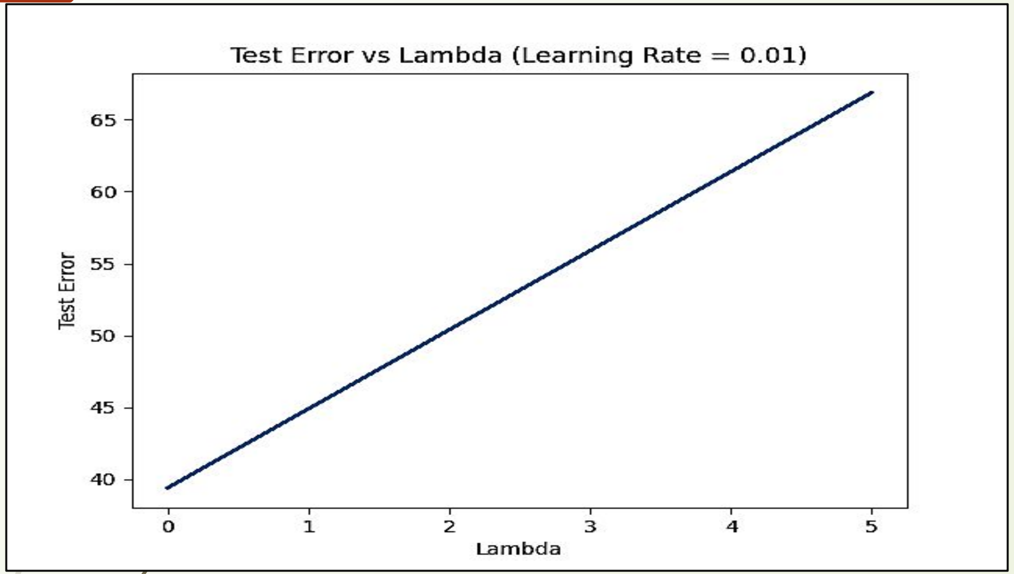
Faster convergence - updates are more frequent and noisy

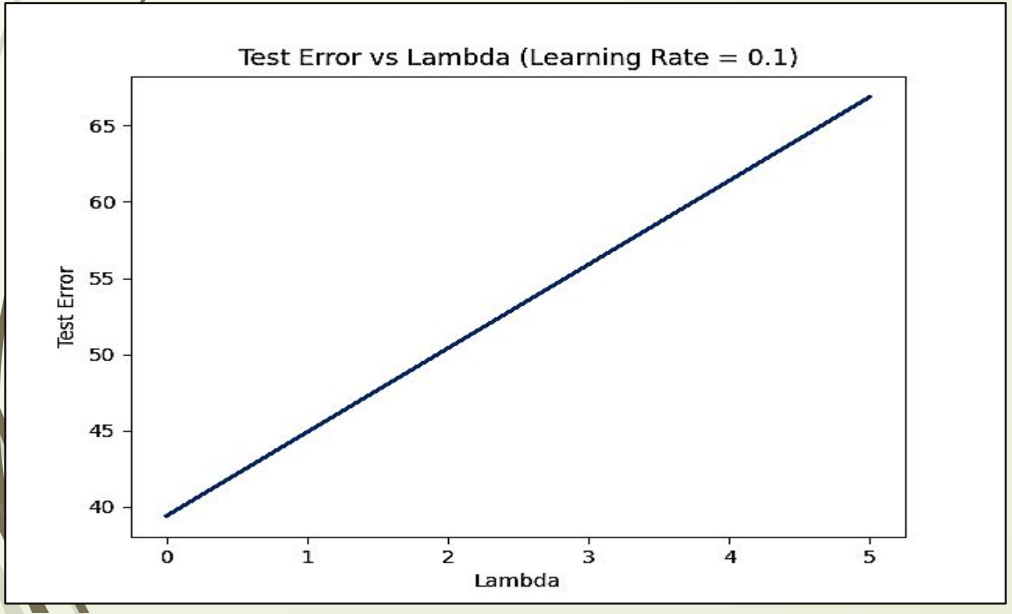
More oscillations in cost function – due to randomness of the parameters update

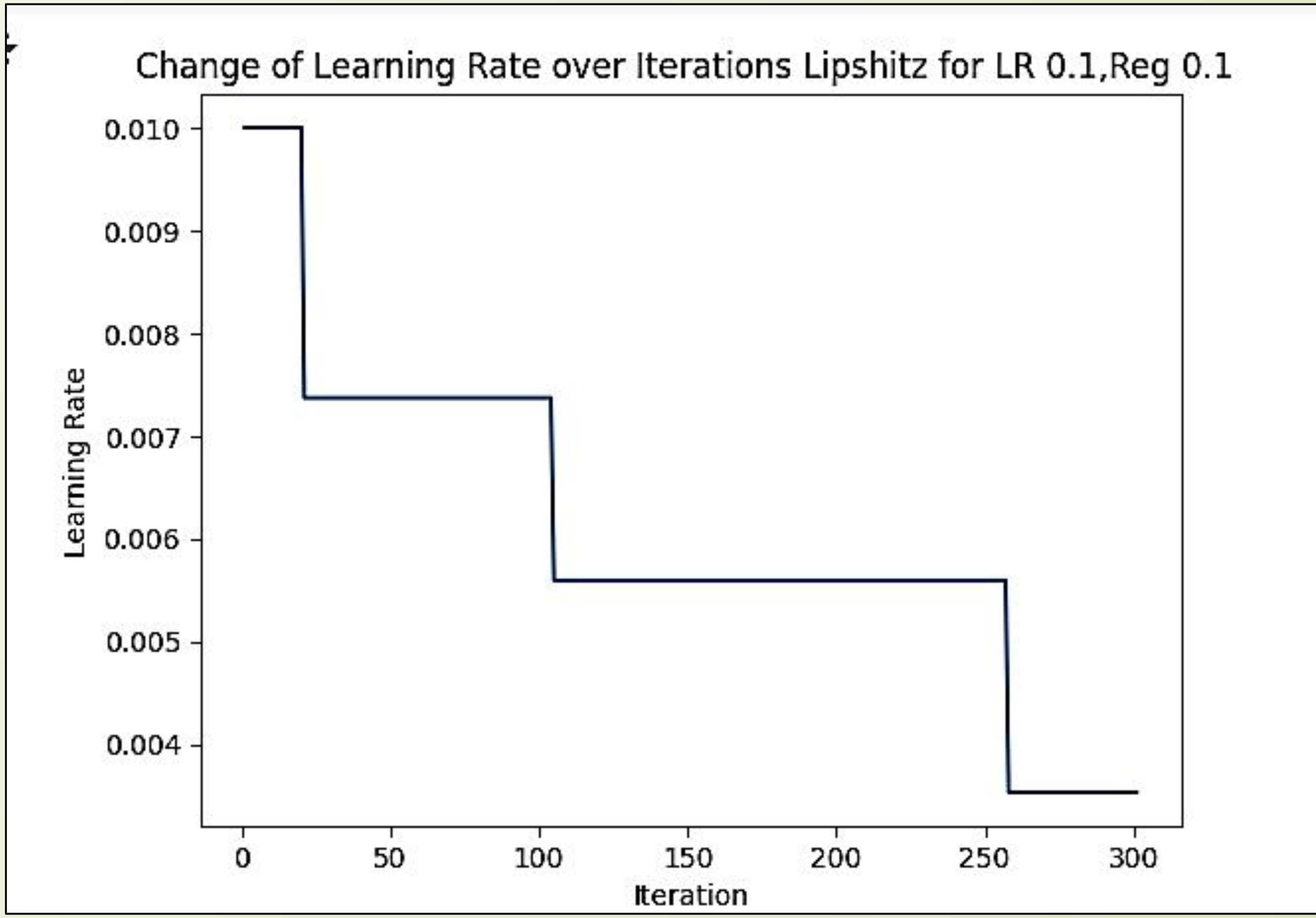
Mathematical formulation:

$\theta = \theta - \alpha \cdot \nabla J_i(\theta)$
where:
θ is the parameter vector
α is the learning rate
$\nabla J_i(\theta)$ is the gradient of the cost function J with respect to θ computed on a single randomly selected training example i

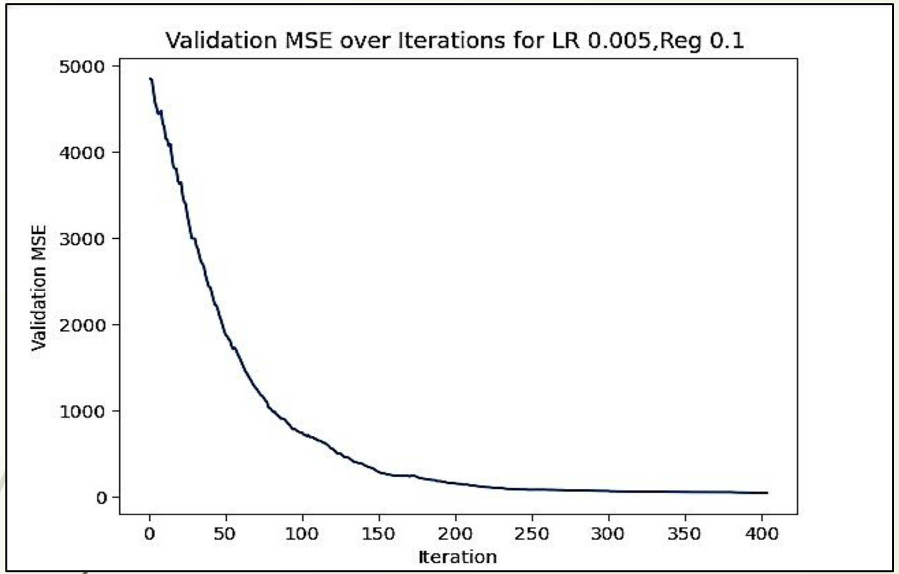
2.1 Learning rate sensitivity analysis

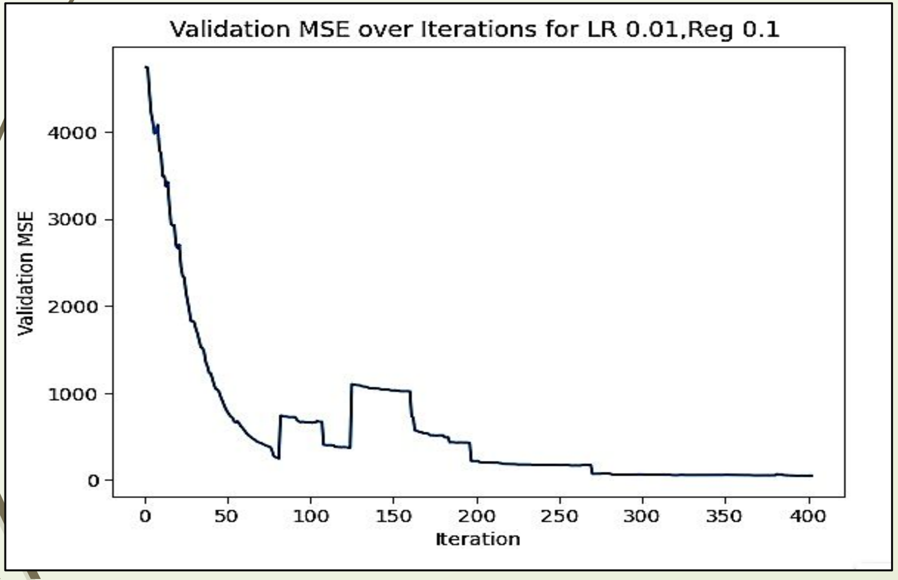


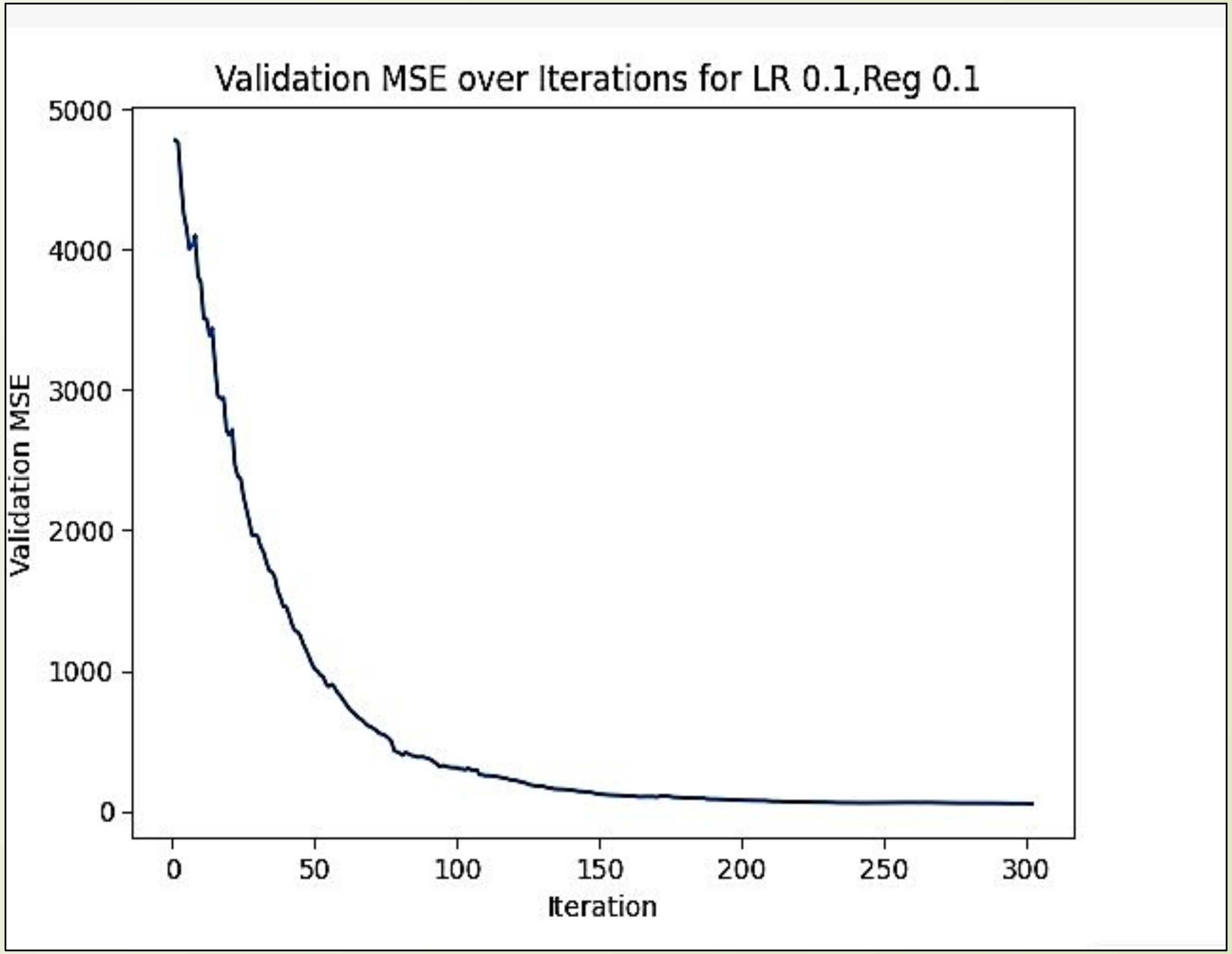




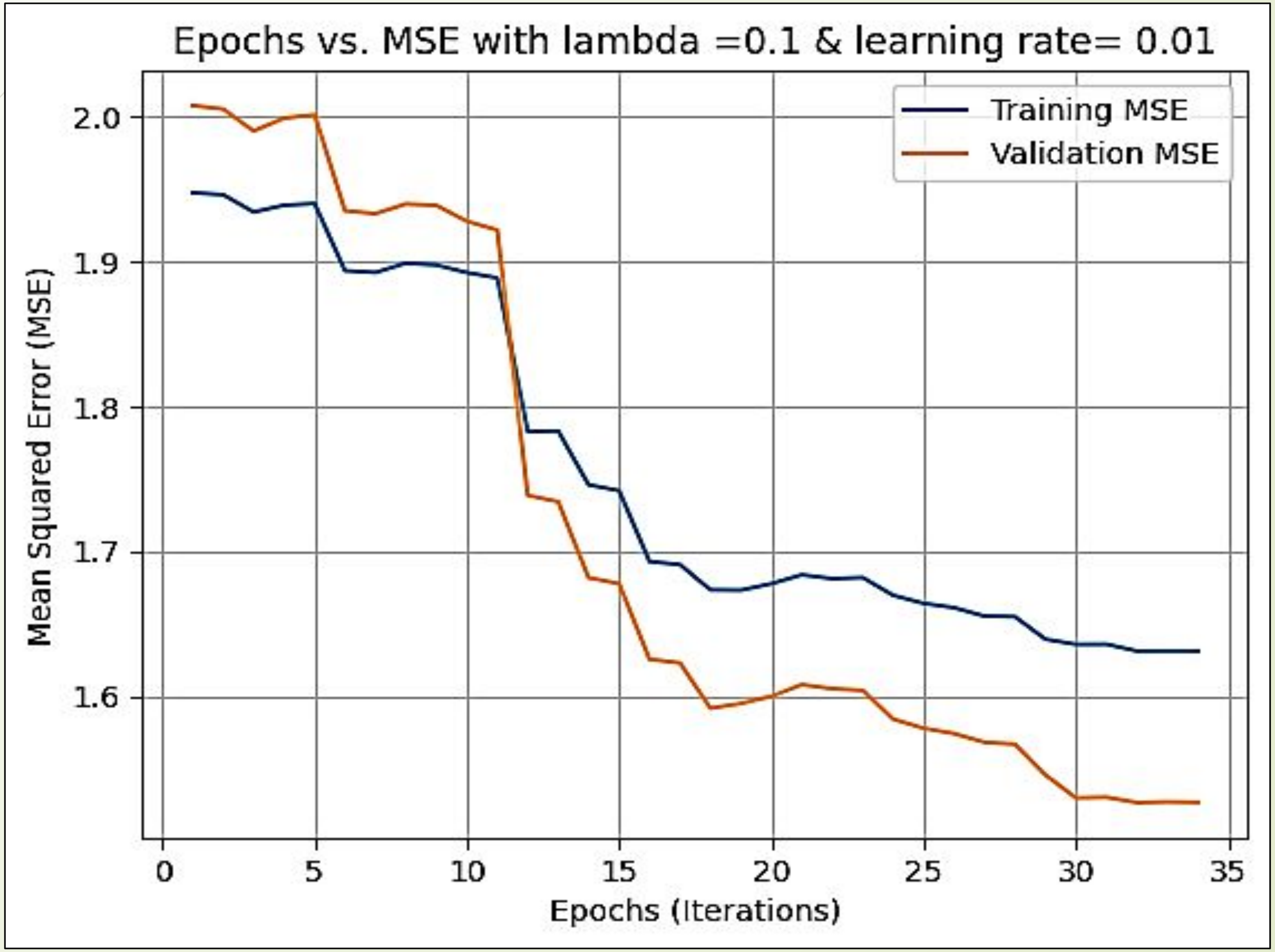
2.2 Convergence Analysis (validation MSE vs Epochs)







2.3 Convergence Analysis –Loss function vs Epochs



3. Analysis on Mini-batch Gradient Descent

Mini-batch (a small random subset of the training dataset, typically between 10 and 1000 examples) –used for gradient computation

Reduced computational cost of the algorithm ; compared to batch gradient descent- Reducing the variance of the updates compared to SGD

Good balance between convergence speed and stability

Mathematical Formulation:

$$\theta = \theta - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla J_i(\theta)$$

where:

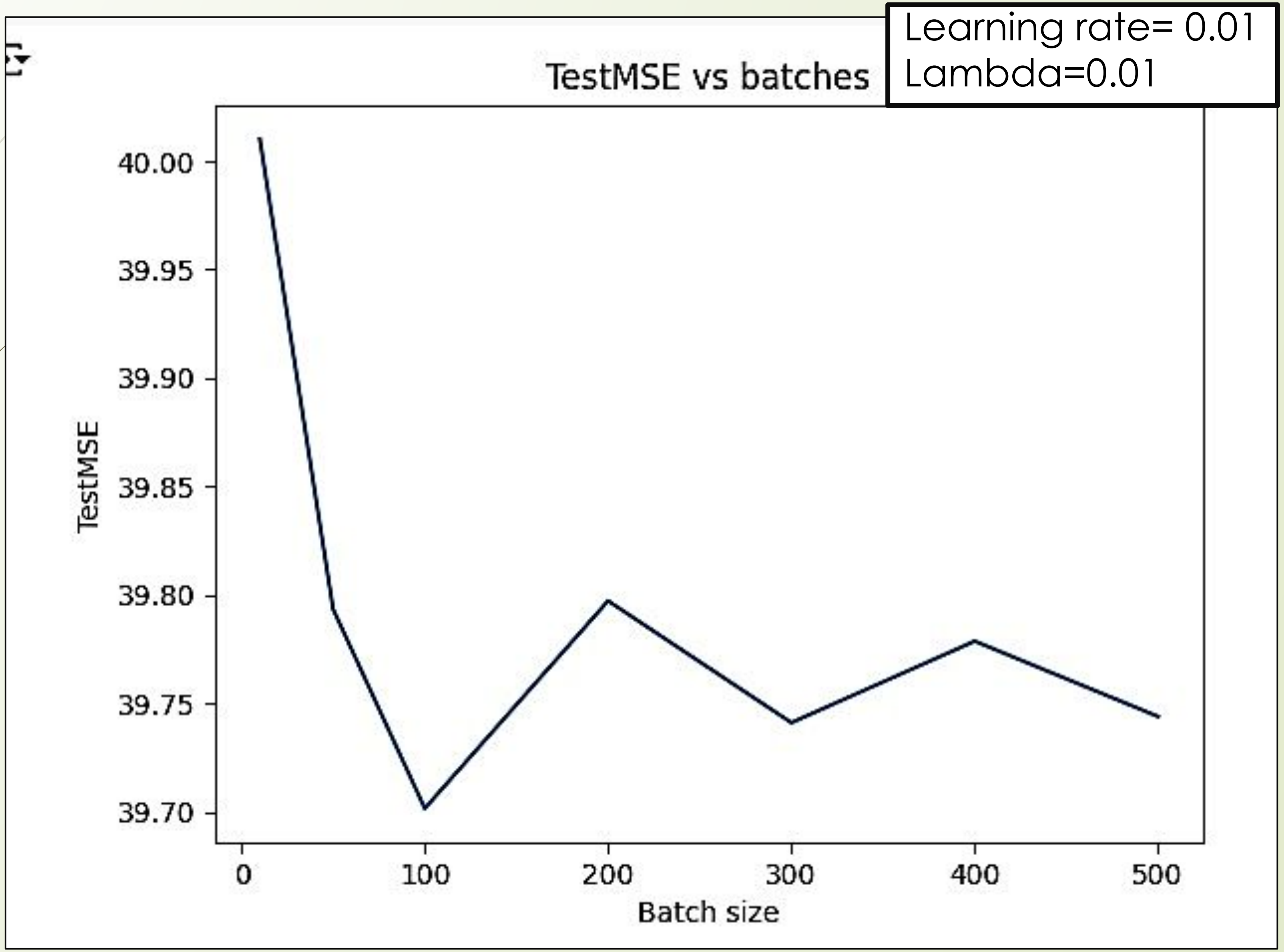
θ is the parameter vector

α is the learning rate

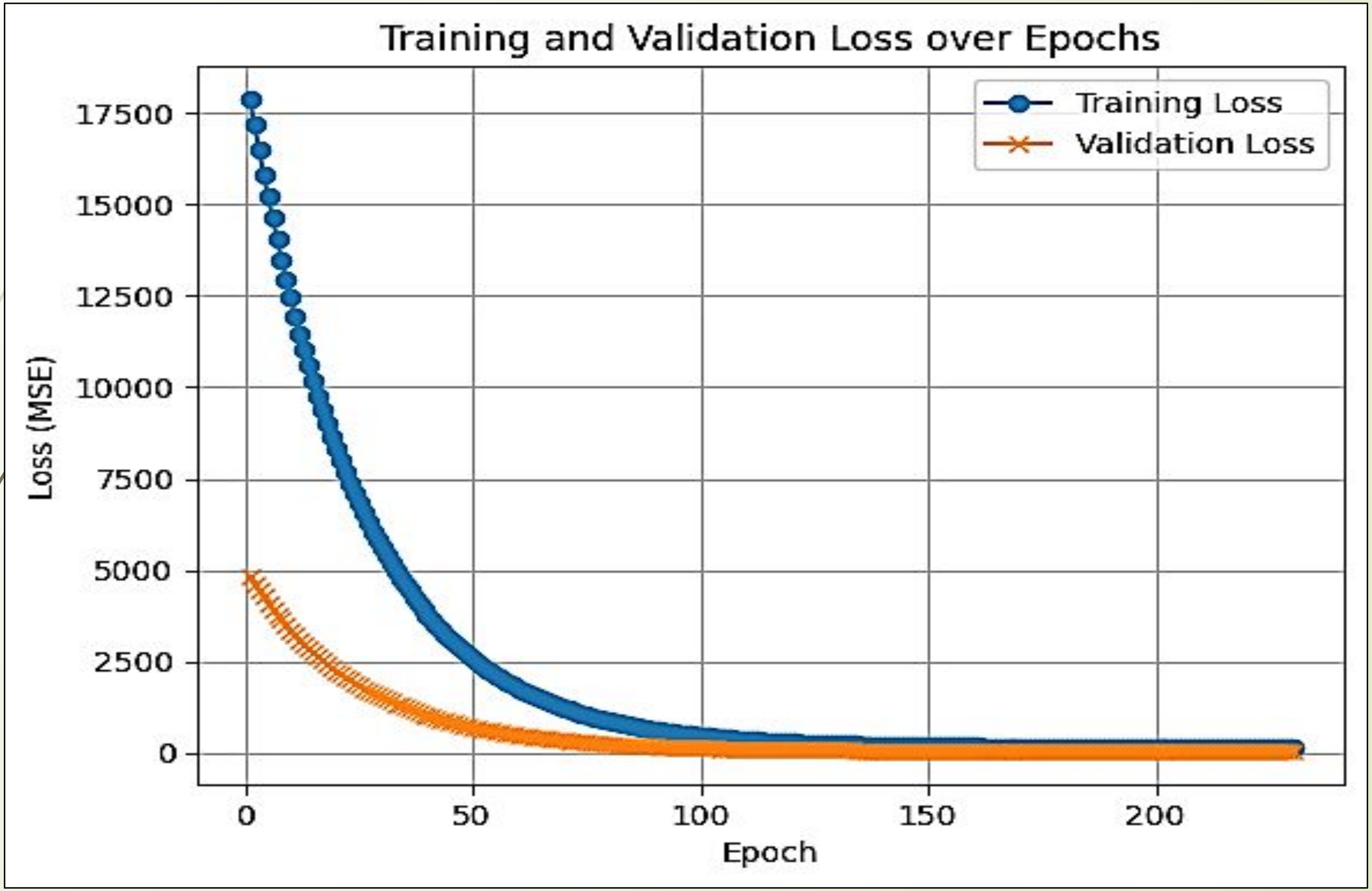
m is the mini-batch size

$\nabla J_i(\theta)$ is the gradient of the cost function J with respect to θ computed on a mini-batch of size m

3.1Batch size sensitivity Analysis



3.2 Convergence Analysis – Loss function vs Epochs



4. Analysis on Momentum Gradient Descent

Addition of a momentum term to the update rule

It accumulates the gradient values over time

Dampens the oscillations in the cost function - leading to faster convergence

Useful when the **cost function has a lot of noise or curvature** - chances to get stuck in local minima

Mathematical Formulation:

$$v_t = \gamma v_{t-1} + \alpha \cdot \nabla J(\theta_{t-1})$$
$$\theta_t = \theta_{t-1} - v_t$$

where:

θ_t is the parameter vector at iteration t

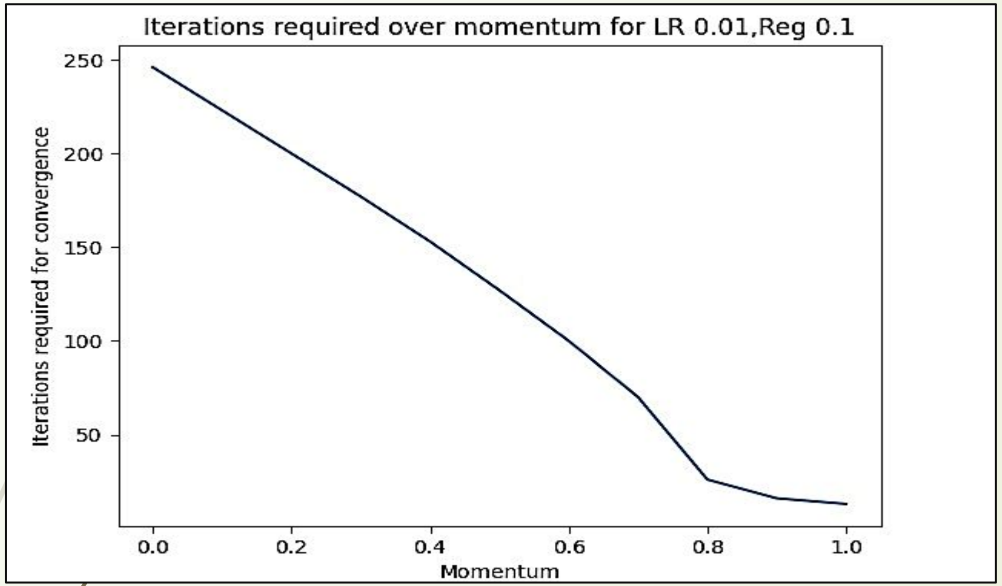
α is the learning rate

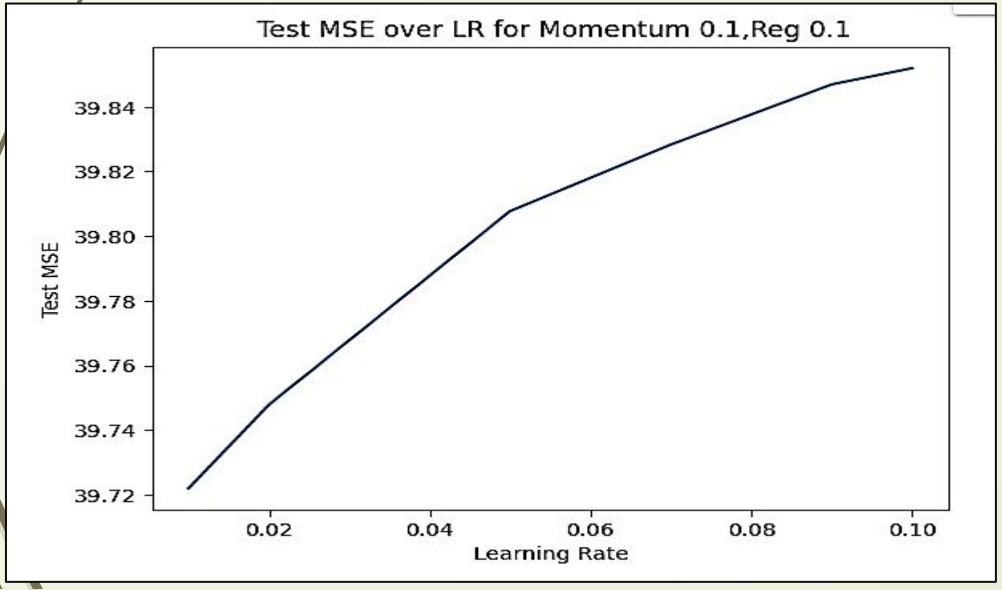
$\nabla J(\theta_{t-1})$ is the gradient of the cost function J with respect to θ evaluated at θ_{t-1}

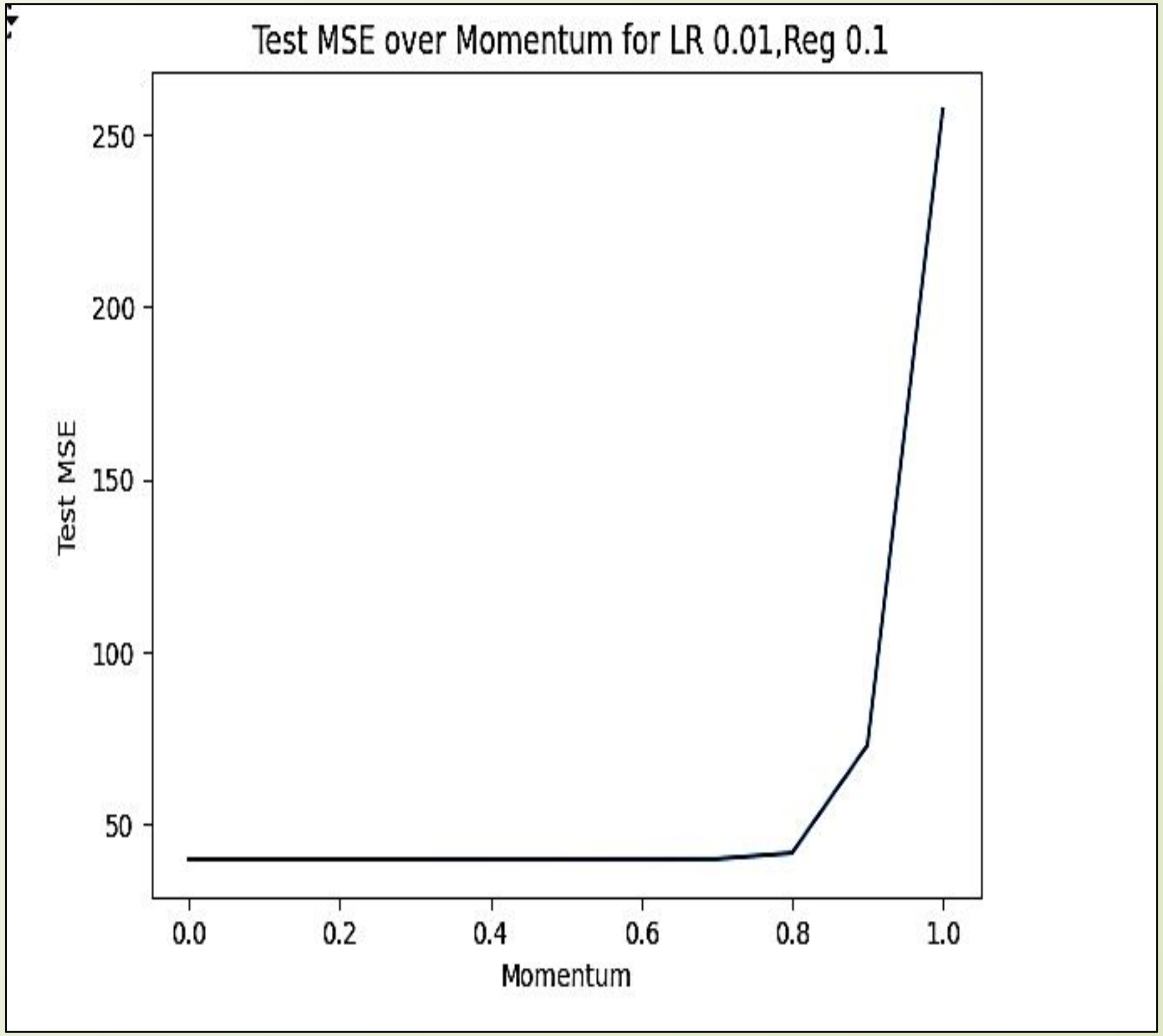
v_t is the velocity vector at iteration t

γ is the momentum coefficient, which controls the contribution of the previous velocity vector to the current velocity vector

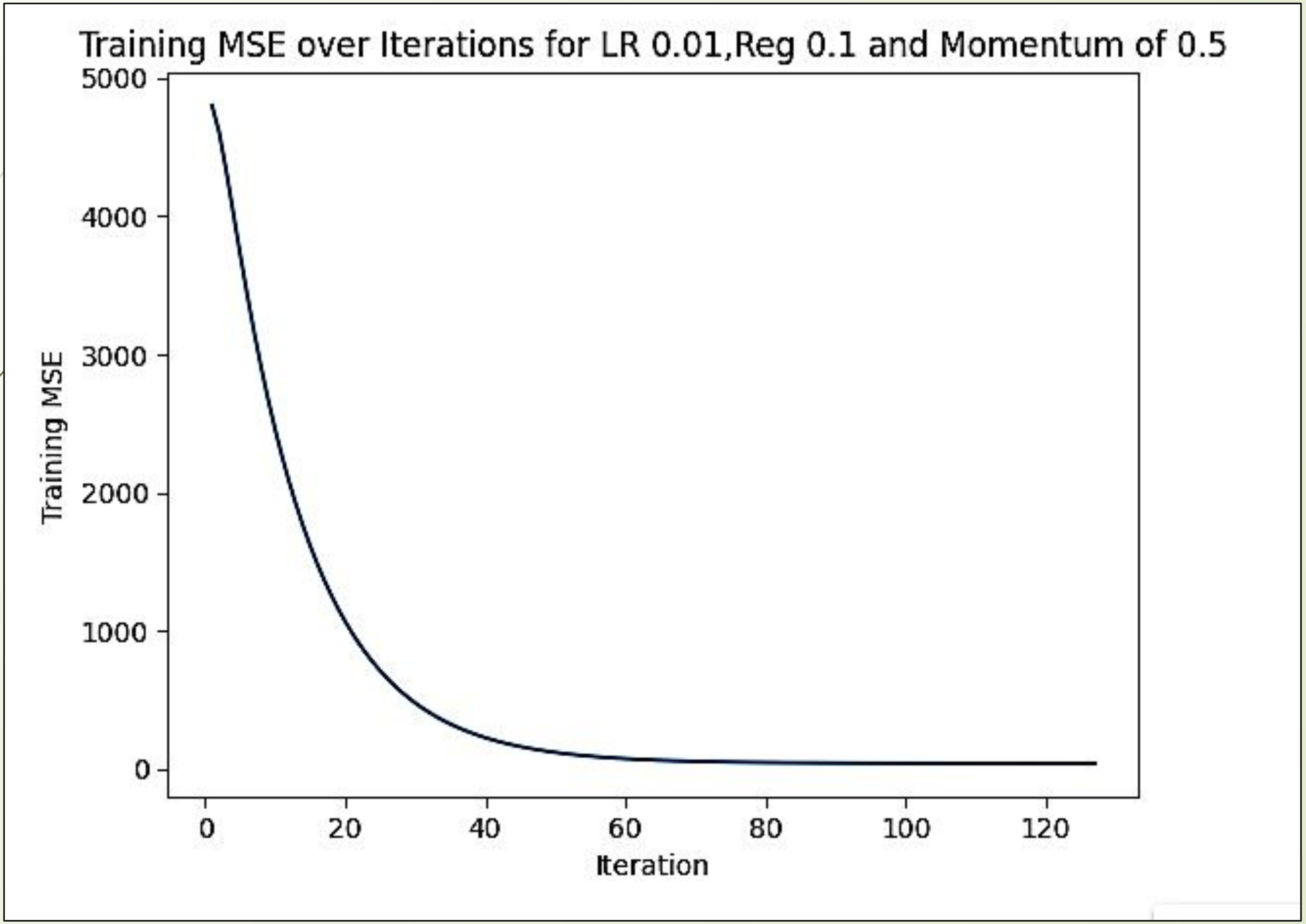
4.1 Sensitivity Analysis







4.2 Convergence Analysis – Epochs vs Loss function



5. Analysis on Nesterov accelerated gradient

An **extension of momentum gradient descent** - takes into account the future gradient values when computing the momentum term.

Reduced overshooting - lead to faster convergence than momentum gradient descent

Mathematical Formulation:

$$v_t = \gamma v_{t-1} + \alpha \cdot \nabla J(\theta_{t-1} - \gamma v_{t-1})$$
$$\theta_t = \theta_{t-1} - v_t$$

where:

θ_t is the parameter vector at iteration t

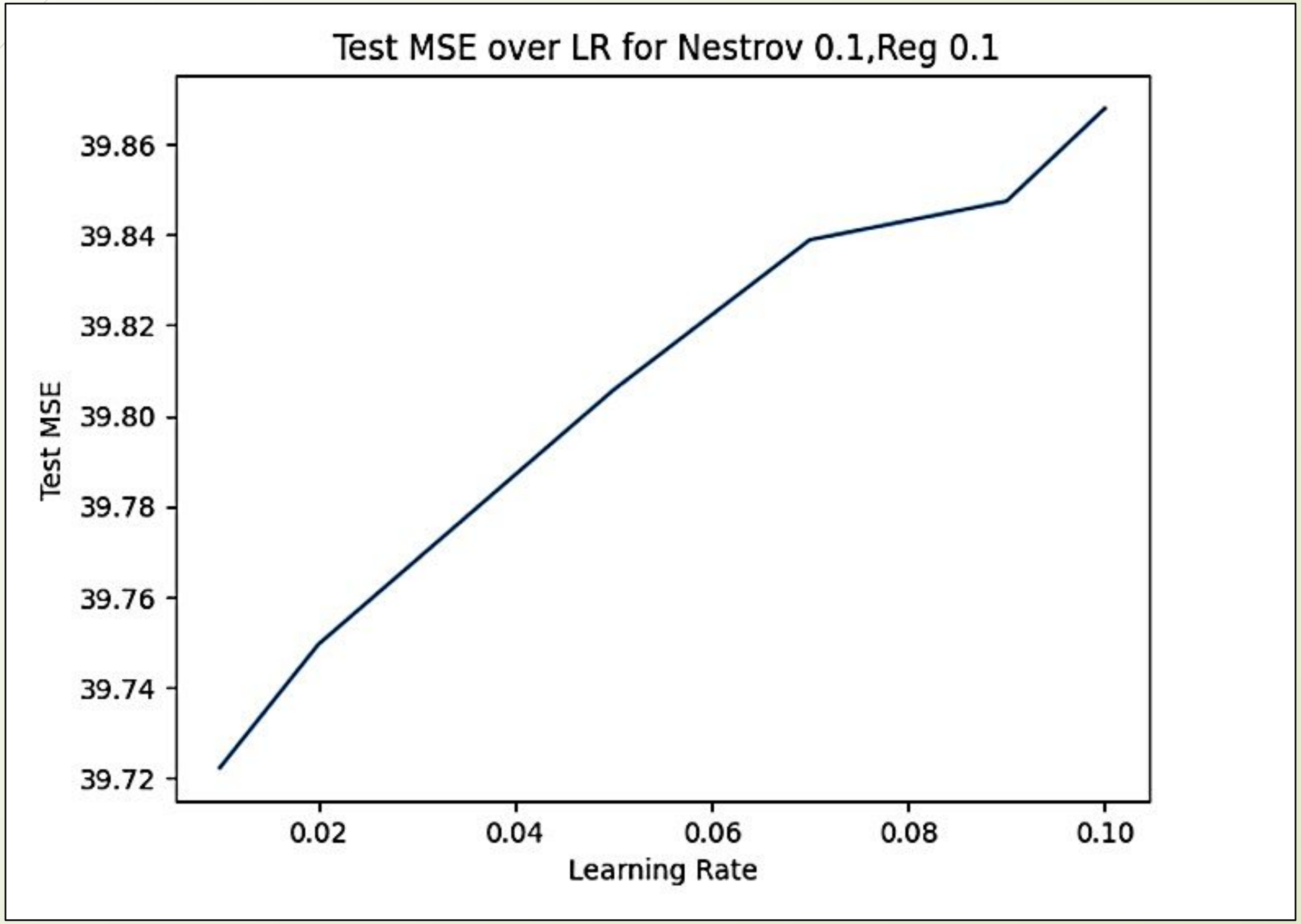
α is the learning rate

$\nabla J(\theta_{t-1} - \gamma v_{t-1})$ is the gradient of the cost function J with respect to θ evaluated at $\theta_{t-1} - \gamma v_{t-1}$

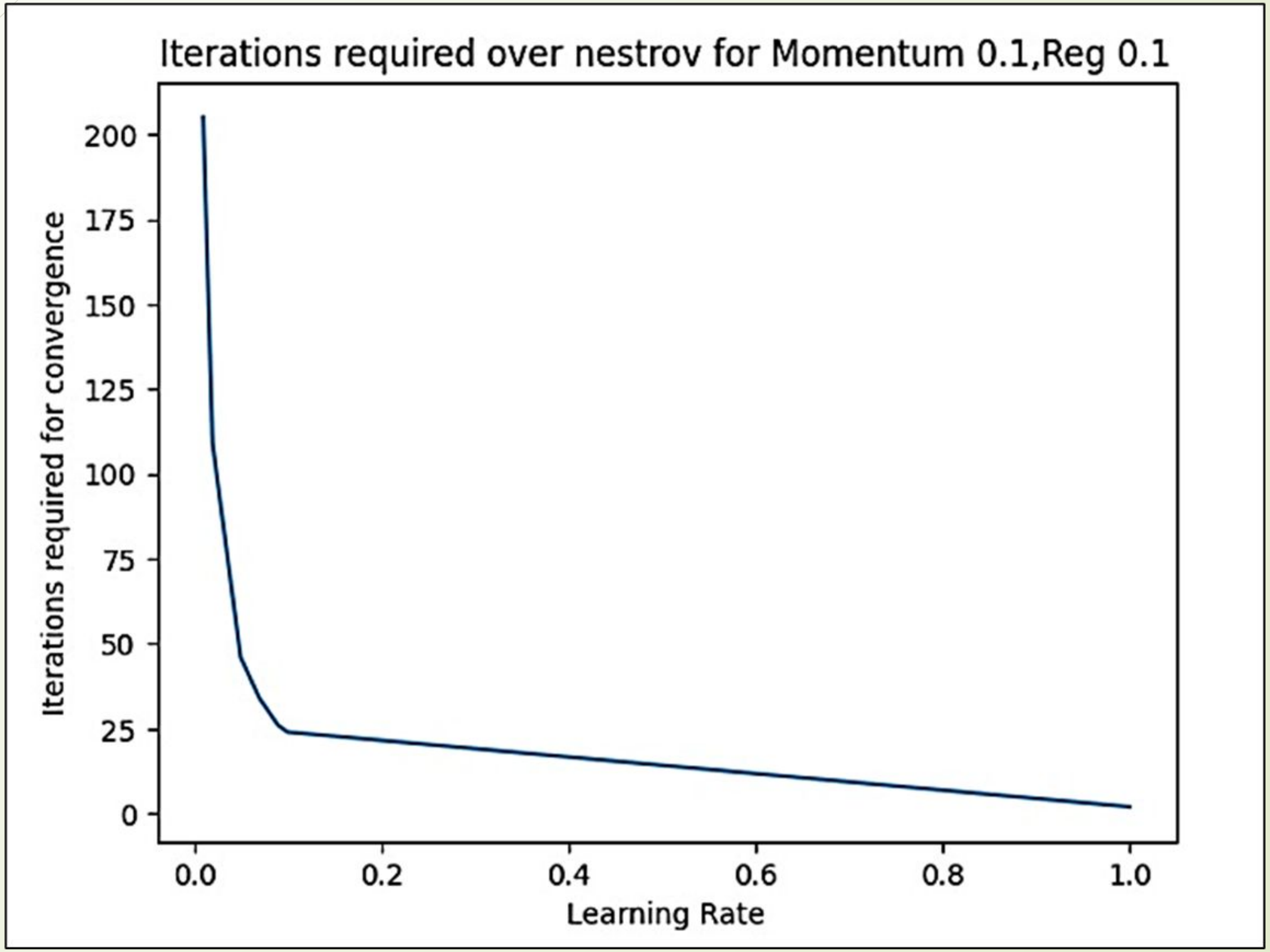
v_t is the velocity vector at iteration t

γ is the momentum coefficient, which controls the contribution of the previous velocity vector to the current velocity vector

5.1 Learning rate sensitivity analysis



5.2 Convergence analysis



Comparison of different gradient descent

Variant	Advantages	Disadvantages
Batch gradient descent	Guaranteed convergence to global optimum	Computationally expensive for large datasets, slow convergence
Stochastic gradient descent	Faster convergence, more efficient for large datasets	High variance, may not converge to global optimum
Mini-batch gradient descent	Balanced convergence speed and computational cost, efficient for large datasets	Choice of mini-batch size can be a challenge
Momentum gradient descent	Faster convergence, less likely to get stuck in local minima	May overshoot and oscillate around the optimum

B	C	D	E	F
For all Learning rate =0.01,Reg rate=0.1		Test MSE	Iterations required for convergence	
	Algorithm			
	Batch Grad	39.7247	246	
	Lip-Batch Grad	39.7237	246	
	Stochatic Grad	47.93	401	
	Lip-Stochatic Grad	46.614	301	
	Mini Batch	39.709	251	
	Lip-Mini Batch	40.925084343	209	
	Momentum(0.5)	39.739	127	
	Nestrov(0.5)	39.764	86	

Additional

- Code Link