Analytical Investigation of Optimizers in PyTorch

Name: Aakriti Dhakal
Colab Notebook Link: 🔗

SECTION 1: INTRODUCTION

1.1 Background on Optimizers

Optimizers in machine learning are algorithms or methods designed to minimize the error function or loss function and maximize the training efficiency of a model. They work by adjusting the model's learnable parameters (weights and biases) to reduce prediction error and thus optimize the training process. Different optimizers handle different parameters like gradient updates, learning rates and convergence behaviors in unique ways, which can significantly influence the training speed and model's performance.

The choice of optimizer can influence training efficiency and overall performance of a model. Optimizers navigate the loss landscape by iteratively moving downhill toward regions of lower loss. Techniques like momentum, adaptive learning rates, and gradient clipping help the optimizer move efficiently through flat regions, avoid oscillations, and escape shallow local minima.

1.2 Assignment Objectives

The goal of this assignment is to:

1. Compare 4 popular PyTorch optimizers (Adam, RMSProp, Adagrad, SGD)
2. Test each optimizer under controlled conditions with three different starting points
3. Run experiments across three different iteration scales (100, 1000, 10000)
4. Log and visualize optimizer trajectories on a terrain-like loss surface
5. Analyze convergence patterns, stability, and robustness

1.3 Approach

This assignment emphasizes analytical insights over theoretical definitions. Each optimizer is evaluated empirically through systematic experiments, with results captured through 3D surface plots and 2D contour visualizations. The analysis focuses on observed behaviors rather than textbook explanations.

SECTION 2: EXPERIMENTAL CONFIGURATIONS

2.1 Terrain Landscape

The loss function used in this study is a terrain-like surface with:

- **Global minimum:** Located at (0, 0) with depth A_g = -10
- **Local minima:** Three local minima with parameters: [(-5, 1, 3, 3), (-7, 1.5, -4, -4), (-4, 0.8, -2, 5)]
- **Noise factor:** 0.0 (smooth surface)
- **Terrain range:** X ∈ [-18.10, 9.30], Y ∈ [-10, 10]

Terrain Visualization:

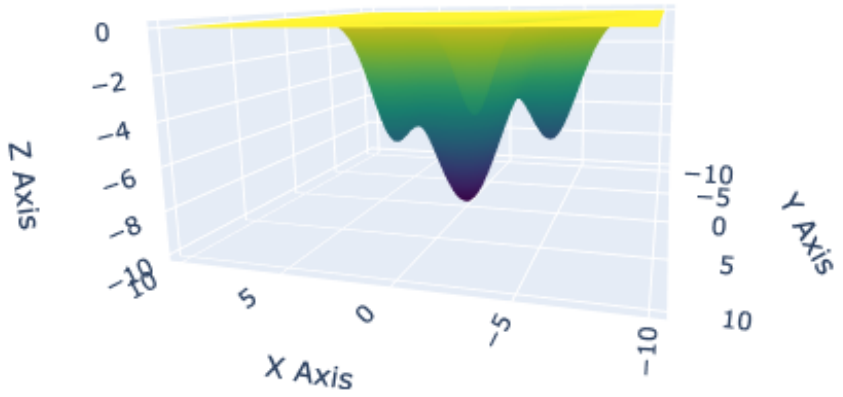

Figure 1: 3D surface plot showing the loss landscape with global minimum (deep blue valley) and local minima

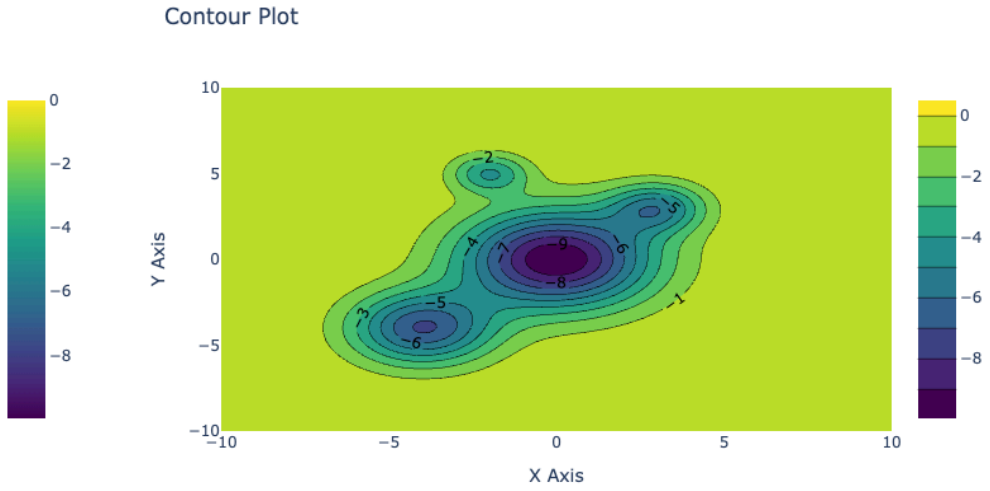

Figure 2: Contour plot (top-down view) showing loss values and minima locations.

2.2 Configuration Table

Config	Start Point (x, y, z)	Iterations	Log Frequency	Category	Summary
C0	(0.5, 0.5)	100	10	Easy	Close to global minimum at (0,0) - optimizer should converge quickly
		1000	10		
		10000	100		
C1	(2.0, 4.0)	100	10	Moderate	Near local minimum - tests ability to escape shallow valleys
		1000	10		
		10000	100		
C2	(-7.0, 5.0)	100	10	Tough	Far from any minimum on plateau - tests robustness and navigation
		1000	10		
		10000	100		

2.3 Starting Point Selection Rationale

C0 - Easy Start (0.5, 0.5): This point was chosen as it was very close to the global minimum and is inside the deepest valley (dark purple zone). The initial loss is -9.4046, which is close to optimal -10.

C1 - Moderate Start (2.0, 4.0): This point is near the upper local minima and is not inside the deepest valley, so it requires navigation. The initial loss is -2.6603 which is of moderate difficulty. This moderate start helps test if optimizers can escape local minima.

C2 - Tough Start (-7.0, 5.0): This point is on the plateau (green/yellow zone) on the plot and is far from any minimum so requires a long journey. The initial loss is -0.001 (using ADAM optimizer). This point is not extreme so optimizers don't fail completely and it can also test if optimizers can bootstrap from bad starts.

2.4 Experimental Parameters

- **Learning rates:** For this experiment, I chose the following 4 optimizers. Their learning rates are also given below:
    1. Adam: lr=0.01 for easy, moderate and tough starts
    2. Adagrad: lr=0.1 for easy and moderate and 0.5 for tough start
    3. SGD: lr=0.01 for easy and moderate and 0.1 for tough start
    4. RMSProp: lr= 0.01 for easy, moderate and tough start

    Adam and RMSprop optimizers are adaptive and work consistently across all starts. SGD is non-adaptive and lr=0.1 is needed for tough starts as it needs higher lr from plateau. Adagrad is also non-adaptive. It accumulates square gradients and the learning rate shrinks over time. lr=0.1 is to compensate for accumulation at easy and moderate starts and lr=0.5 is needed to give momentum to start on a plateau.

- **Number of trials per configuration:** 36 total runs (4 optimizers × 3 configs × 3 iteration scales)
- **Logging strategy:** Log every 10 steps for 100 and 1000 iterations; every 100 steps for 10000 iterations

---

SECTION 3: OPTIMIZER EXPERIMENTS

3.1 Adam

Hyperparameters: lr = 0.01

Config C0 - Easy Start (0.5, 0.5)
100 iterations (log every 10 steps):

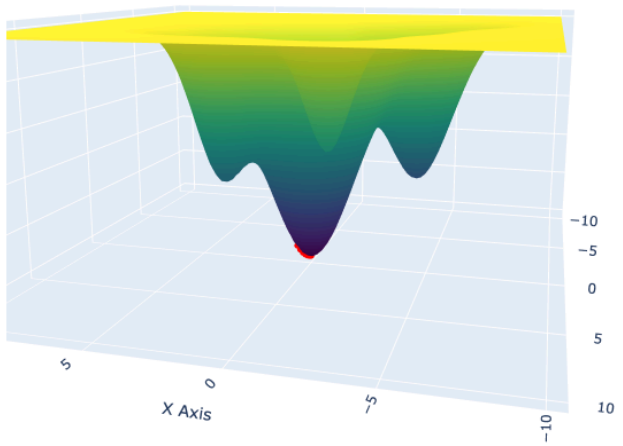

Figure 3.1.1: Adam trajectory in 3D (100 iterations, easy start)

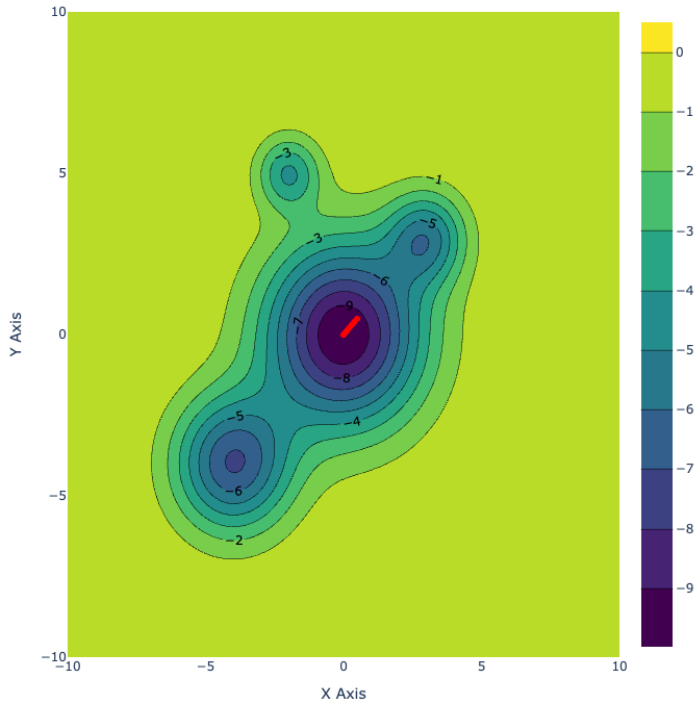

Figure 3.1.2: Adam trajectory on contour plot (100 iterations, easy start)

Summary: Adam converged quickly and smoothly from (0.49, 0.49) to final position (-0.0067, -0.0067) with loss -10.0063 in 100 iterations. No oscillation or overshooting was observed; the optimizer followed a direct diagonal path toward the global minimum at (0, 0), with monotonic improvement throughout.

1000 iterations (log every 10 steps):

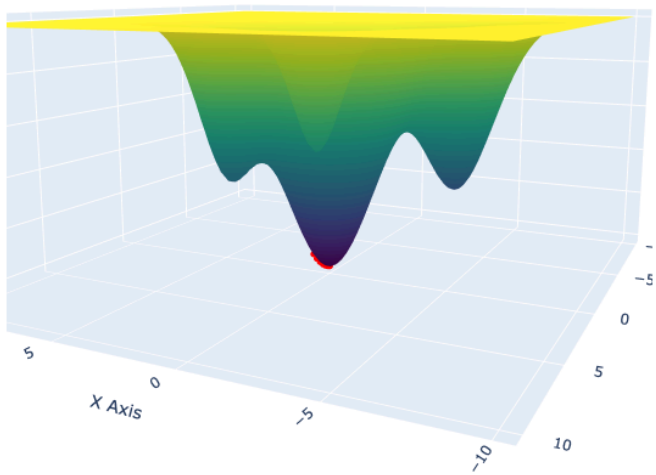

Figure 3.1.3: Adam trajectory in 3D (1000 iterations, easy start)

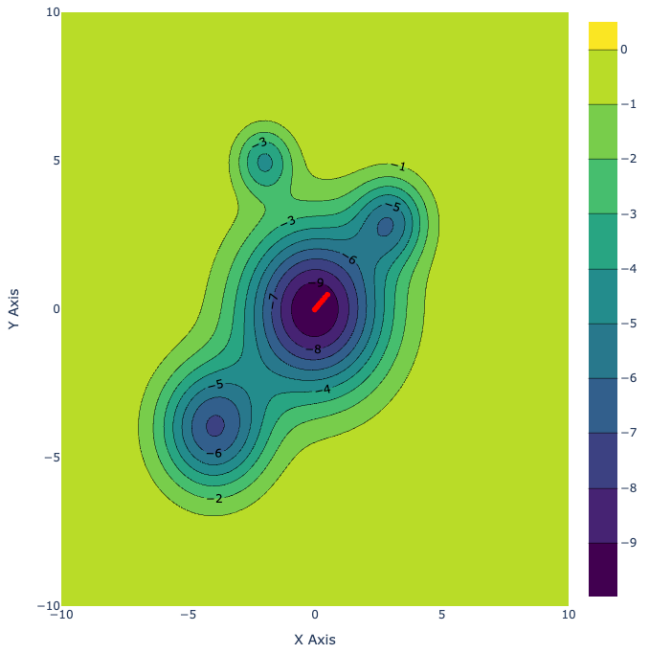

Figure 3.1.4: Adam trajectory on contour plot (1000 iterations, easy start)

Summary: Adam converged very quickly from (0.49, 0.49) to final position (-0.0034, -0.0034) with loss -10.0064, reaching optimal performance by iteration ~200 and remaining stationary for the remaining 800 iterations. No oscillation or overshooting occurred; the optimizer followed a direct diagonal path and plateaued once near the global minimum, demonstrating efficient convergence with no further improvement needed beyond the midpoint of the run.

10000 iterations (log every 100 steps):

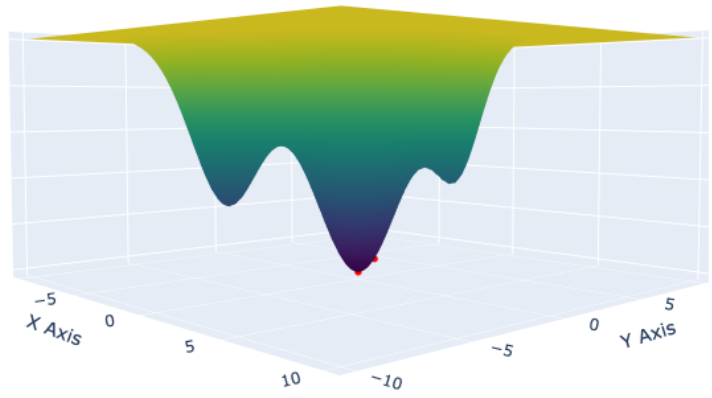

Figure 3.1.5: Adam trajectory in 3D (10000 iterations, easy start)

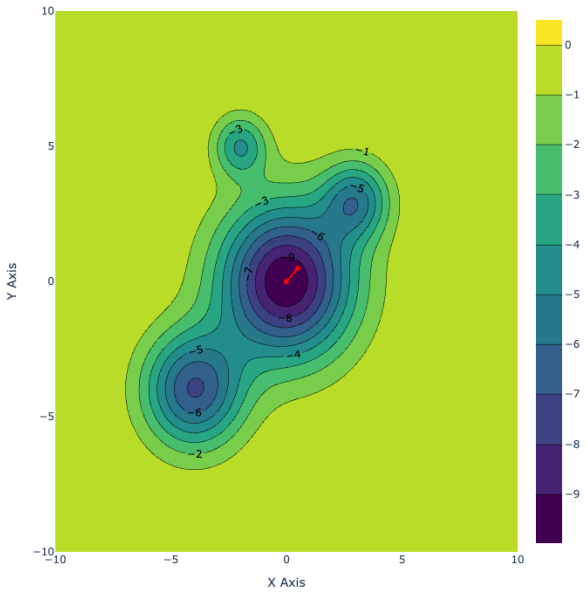

Figure 3.1.6: Adam trajectory on contour plot (10000 iterations, easy start)

Summary: Adam converged extremely quickly from (0.49, 0.49) to final position (-0.0034, -0.0034) with loss -10.0064, achieving optimal convergence by iteration 101 and remaining completely stationary for the remaining 9,900 iterations. No oscillation or overshooting was observed; the optimizer followed a direct path to the global minimum and plateaued immediately, demonstrating that extending to 10,000 iterations provided no additional benefit beyond the first 100 steps.

Config C1 - Moderate Start (2.0, 4.0)

100 iterations (log every 10 steps):

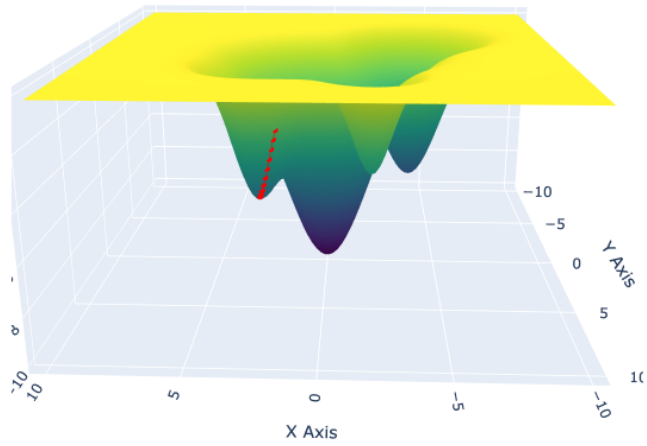

Figure 3.1.7: Adam trajectory in 3D (100 iterations, moderate start)

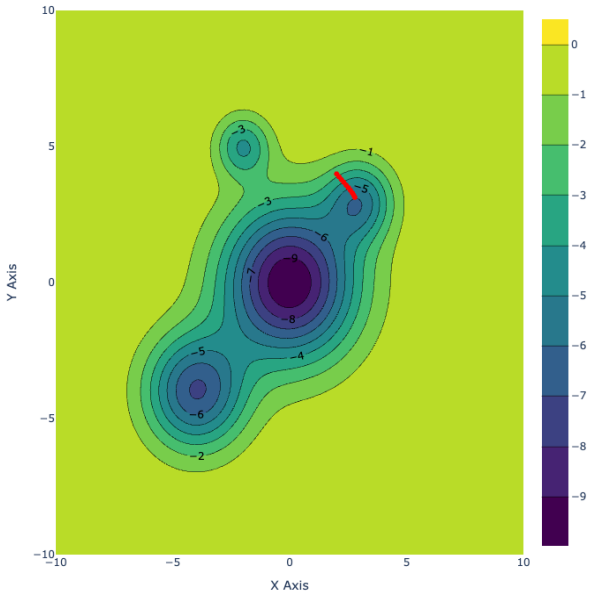

Figure 3.1.8: Adam trajectory on contour plot (100 iterations, moderate start)

Summary: Adam converged partially from (2.01, 3.99) to final position (2.79, 3.13) with loss -5.94 in 100 iterations, showing steady progress but failing to escape the local minimum region near (3, 5). No oscillation or overshooting occurred; the optimizer followed a direct diagonal path toward the global minimum but insufficient iterations prevented full escape from the local basin, with loss improving monotonically from -2.66 to -5.94.

1000 iterations (log every 10 steps):

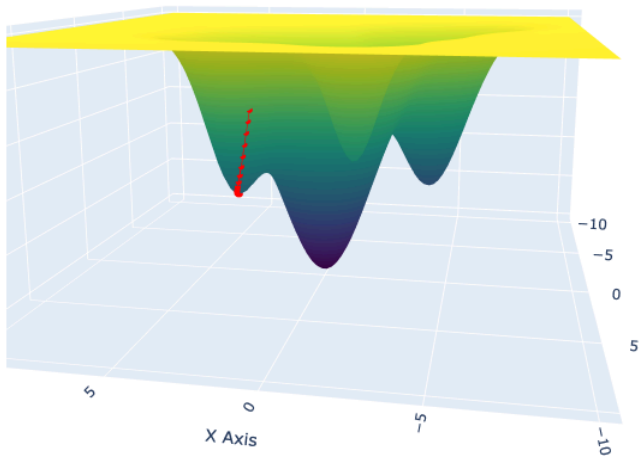

Figure 3.1.9: Adam trajectory in 3D (1000 iterations, moderate start)

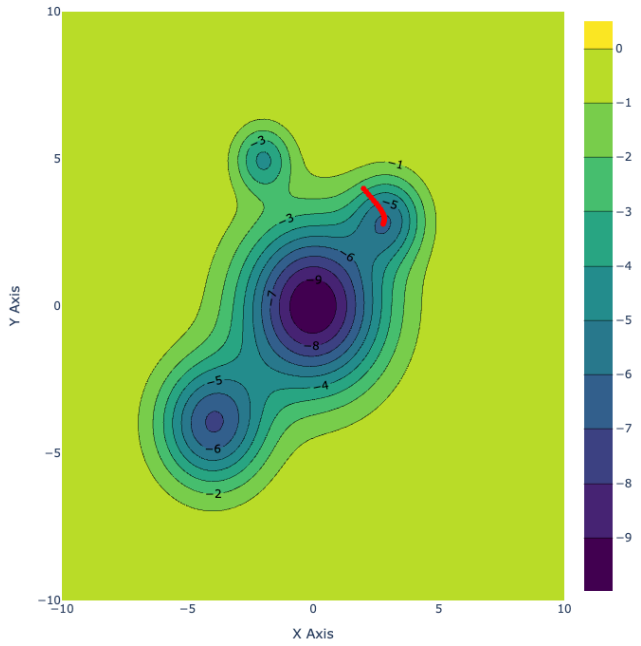

Figure 3.1.10: Adam trajectory on contour plot (1000 iterations, moderate start)

Summary: Adam converged from (2.01, 3.99) to final position (2.79, 2.79) with loss -6.21, but became trapped in a local minimum after approximately 240 iterations and made no further progress for the remaining 760 iterations. No oscillation or overshooting occurred; the optimizer followed a smooth diagonal path but stagnated at position (2.79, 2.79), failing to escape the local minimum basin and reach the global minimum at (0, 0) with loss -10.

10000 iterations (log every 100 steps):

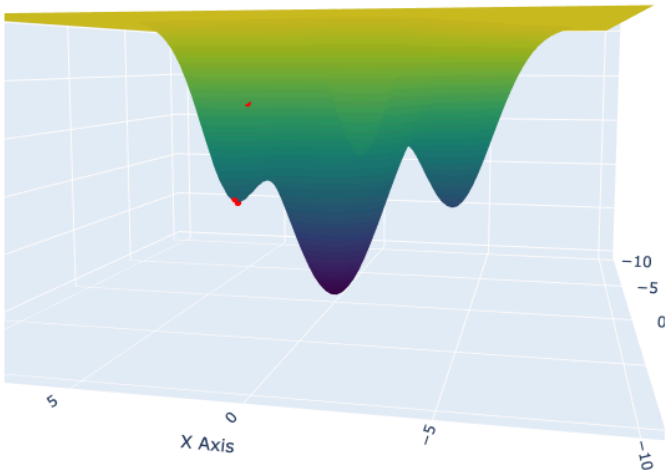

Figure 3.1.11: Adam trajectory in 3D (10000 iterations, moderate start)

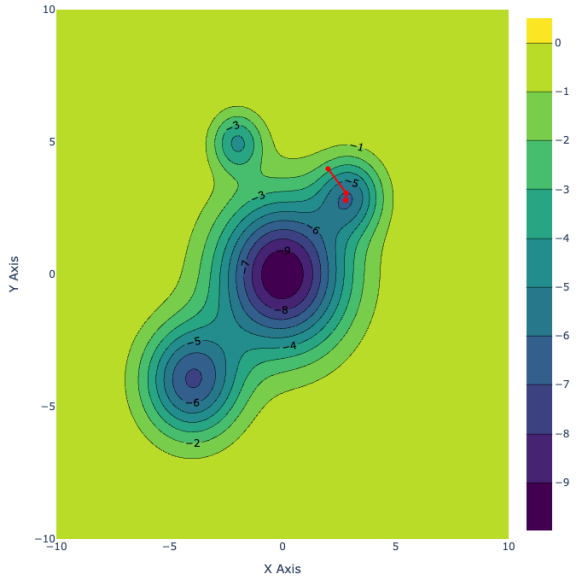

Figure 3.1.12: Adam trajectory on contour plot (10000 iterations, moderate start)

Summary: Adam converged from (2.01, 3.99) to final position (2.79, 2.79) with loss -6.21, becoming permanently trapped in a local minimum by iteration ~300 and remaining completely stationary for the remaining 9,700 iterations. No oscillation or overshooting occurred; the optimizer followed a smooth path initially but stagnated far from the global minimum, demonstrating that even 10,000 iterations and adaptive learning rates cannot overcome strong local minima attraction, with the final loss of -6.21 falling 3.8 units short of the optimal -10.

Config C2 - Tough Start (-7.0, 5.0)

100 iterations (log every 10 steps):

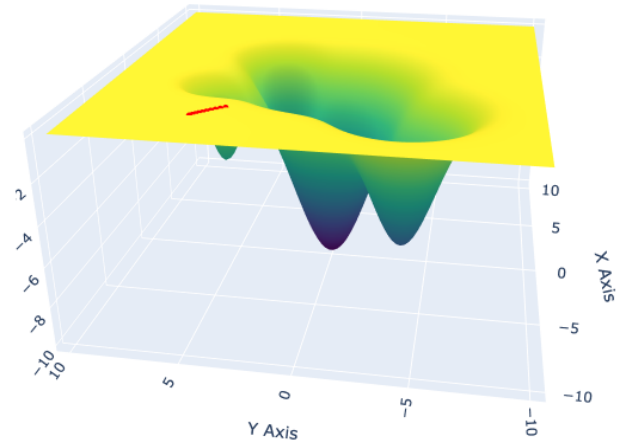

Figure 3.1.13: Adam trajectory in 3D (100 iterations, tough start)

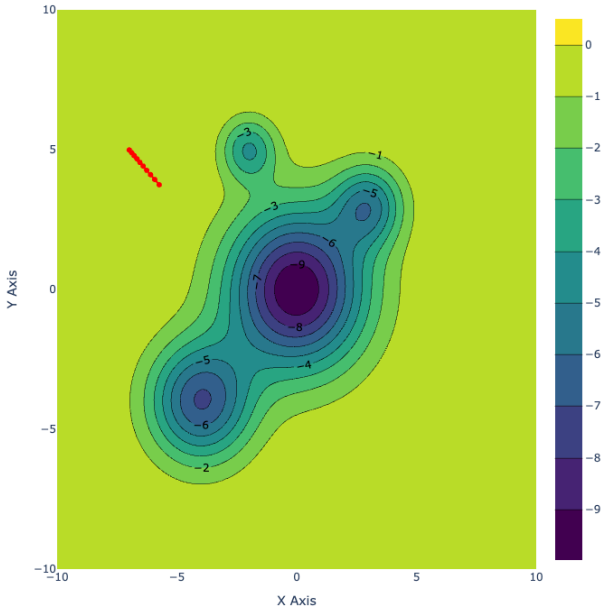

Figure 3.1.14: Adam trajectory on contour plot (100 iterations, tough start)

Summary: Adam converged gradually from the tough starting point (-6.99, 4.99), steadily improving the loss from -0.0010 to -0.027 by iteration 91 and reaching (-5.74, 3.75). The path was smooth and monotonic with no visible oscillation or overshooting, though convergence was gradual and not fully at the global minimum.

1000 iterations (log every 10 steps):

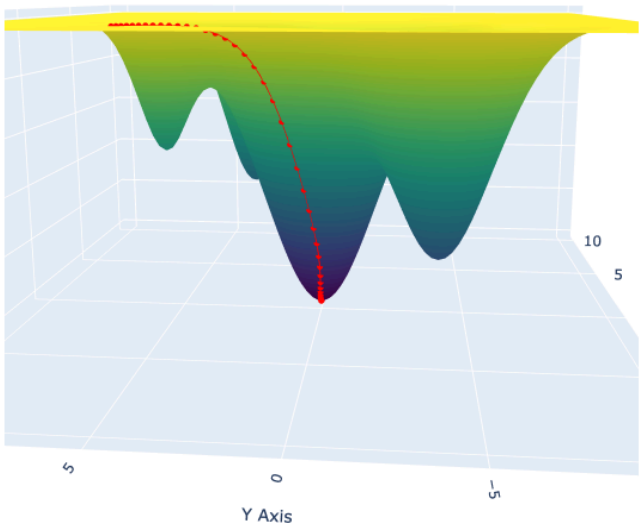

Figure 3.1.15: Adam trajectory in 3D (1000 iterations, tough start)

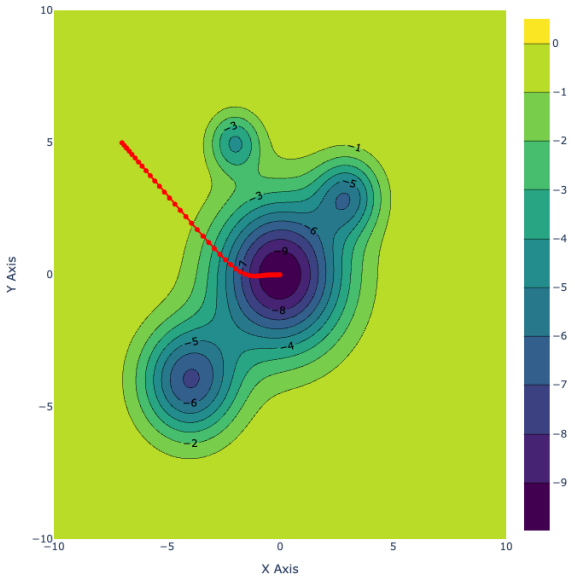

Figure 3.1.16: Adam trajectory on contour plot (1000 iterations, tough start)

Summary: Adam converged successfully from the tough start (-7.0, 5.0), steadily reducing the loss from -0.001 to -10.0064. It reached near the global minimum at approximately (-0.0034, -0.0034) after about 730 iterations. The optimizer showed smooth and consistent progress with no oscillation or overshooting, maintaining stable step sizes throughout. The trajectory was long but direct, gradually accelerating after early slow progress and then flattening near convergence, reflecting Adam's strong stability even from distant starting points.

10000 iterations (log every 100 steps):

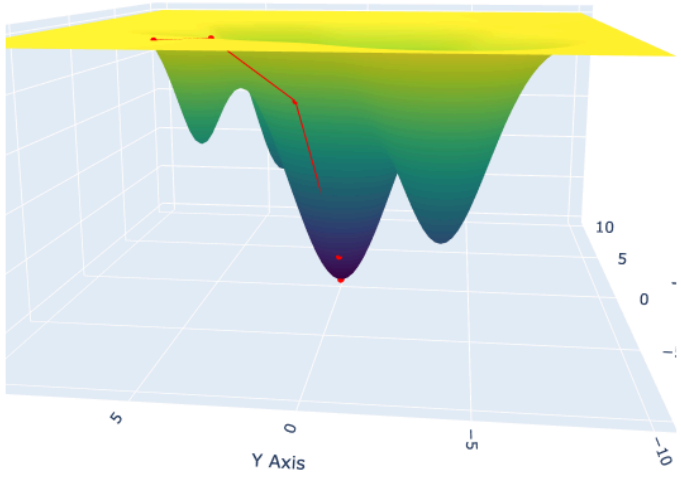

Figure 3.1.17: Adam trajectory in 3D (1000 iterations, tough start)

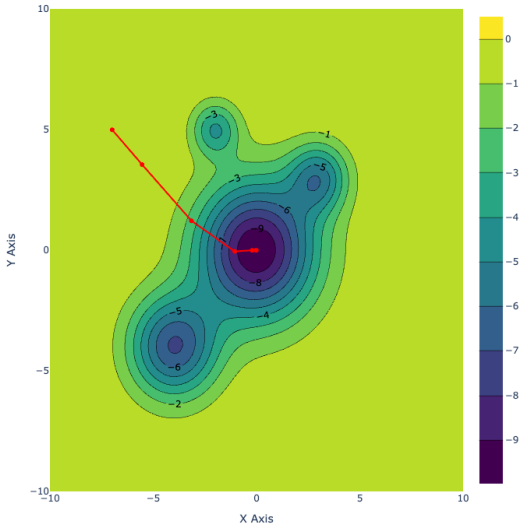

Figure 3.1.18: Adam trajectory on contour plot (1000 iterations, tough start)

Summary: Adam converged effectively from the tough start (-7.0, 5.0), reaching the global minimum near (-0.0034, -0.0034) with a final loss of -10.0064. Convergence began slowly but accelerated around iteration 200, showing rapid loss reduction by iteration 400 and stabilizing by iteration 600. The optimizer maintained a smooth trajectory with no oscillation or overshooting, following a clear, direct path toward the minimum. Overall, convergence was stable and efficient, though it required several hundred iterations to fully settle.

3.2 RMSProp

Hyperparameters: lr = 0.01

Config C0 - Easy Start (0.5, 0.5)

100 iterations (log every 10 steps):

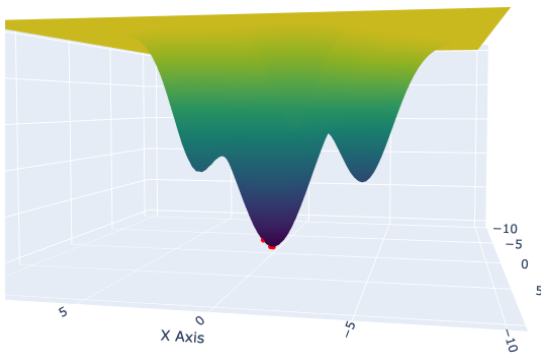

Figure 3.2.1: RMAProp trajectory in 3D (100 iterations, easy start)

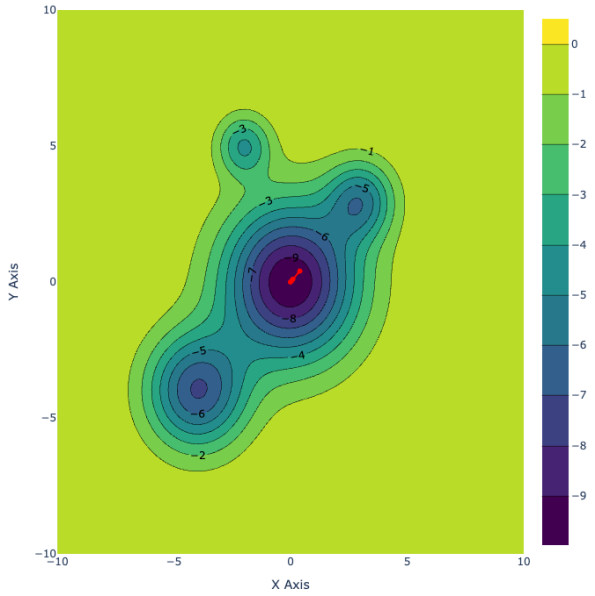

Figure 3.2.2: RMSProp trajectory on contour plot (100 iterations, easy start)

Summary: RMSProp converged very quickly from (0.4, 0.4) to the global minimum near (-0.0034, -0.0034) with a final loss of -10.0064 by around 70 iterations. The trajectory was smooth and direct, showing rapid reduction in loss within the first few steps. No oscillation or overshooting was observed, indicating stable and efficient convergence with a well-tuned learning rate.

1000 iterations (log every 10 steps):

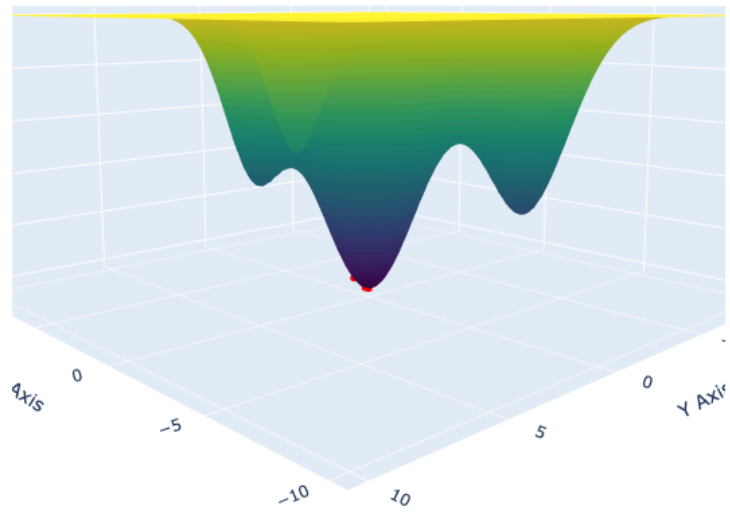

Figure 3.2.3: RMSProp trajectory in 3D (1000 iterations, easy start)

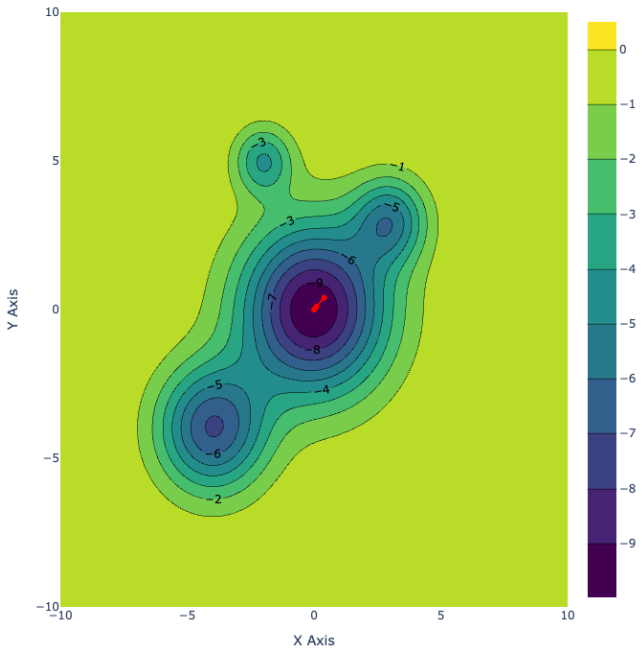

Figure 3.2.4: RMSProp trajectory on contour plot (1000 iterations, easy start)

Summary: RMSProp converged rapidly from (0.4, 0.4) to the global minimum near (-0.0034, -0.0034) with a final loss of -10.0064. Most of the improvement occurred within the first 70 iterations, after which the parameters stabilized and remained constant through iteration 1000. The convergence was smooth, direct, and highly efficient, with no signs of oscillation or overshooting, reflecting strong stability at the chosen learning rate.

10000 iterations (log every 100 steps):

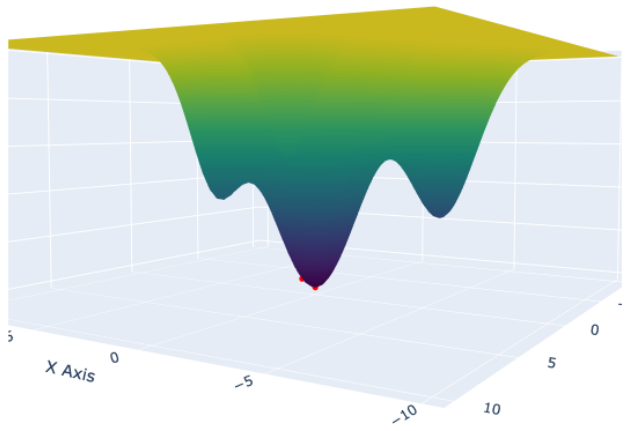

Figure 3.2.5: RMSProp trajectory in 3D (10000 iterations, easy start)

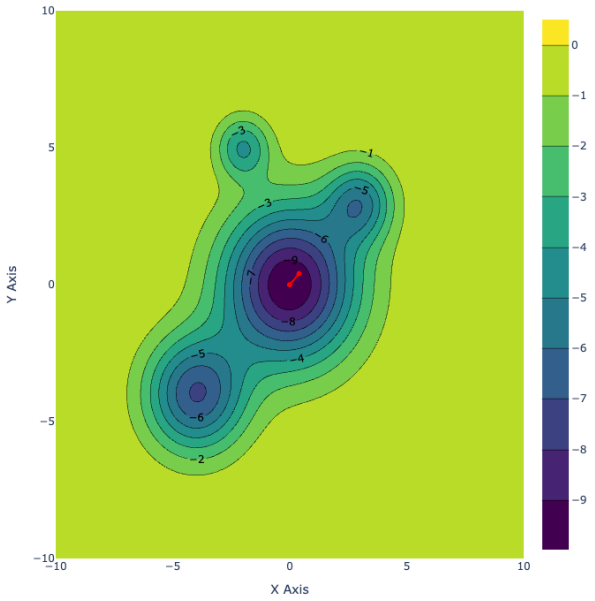

Figure 3.2.6: RMSProp trajectory on contour plot (10000 iterations, easy start)

Summary: RMSProp converged extremely quickly from (0.4, 0.4) to the global minimum near (-0.0034, -0.0034) with a final loss of -10.0064 by around 100 iterations. After that, it remained fully stable through iteration 10,000. The path was highly direct and smooth, with no oscillation or overshooting, showing fast and reliable convergence due to the adaptive learning rate control.

Config C1 - Moderate Start (2.0, 4.0)

100 iterations (log every 10 steps):

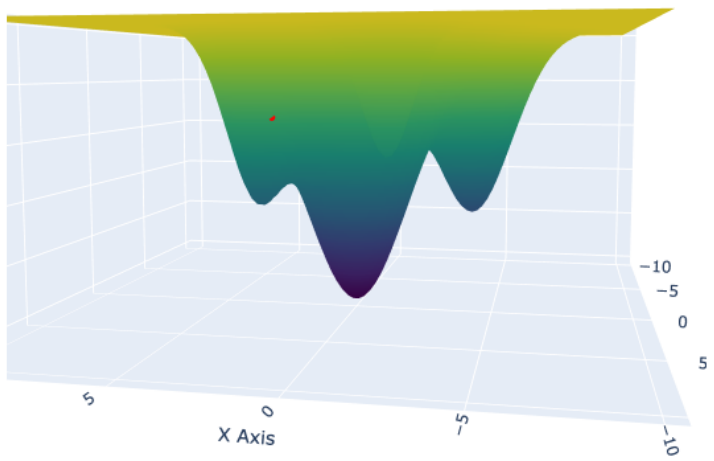

Figure 3.2.7: RMSProp trajectory in 3D (100 iterations, moderate start)

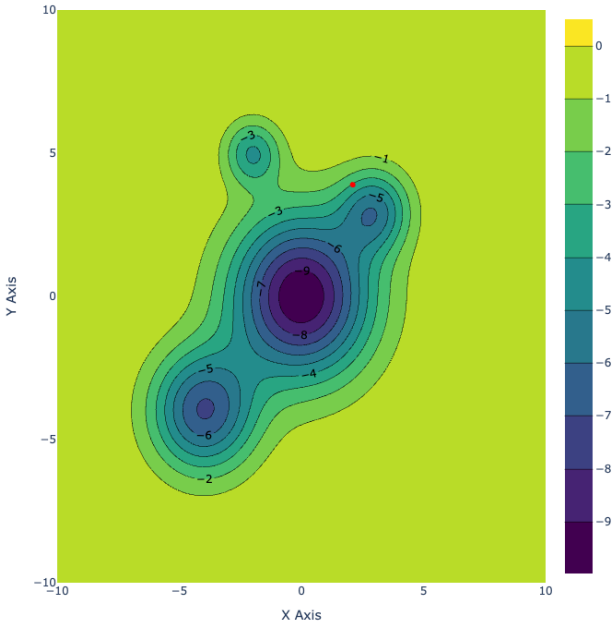

Figure 3.2.8: RMSProp trajectory on contour plot (100 iterations, moderate start)

Summary: For the moderate start (2.1, 3.9), the optimizer converged smoothly toward the minimum, reaching approximately (2.80, 2.82) with a final loss of -6.21 after 91 iterations. The path was gradual and stable, showing steady improvement without oscillation or overshooting. Convergence slowed near the minimum but remained consistent, indicating a well-controlled and direct descent.

1000 iterations (log every 10 steps):

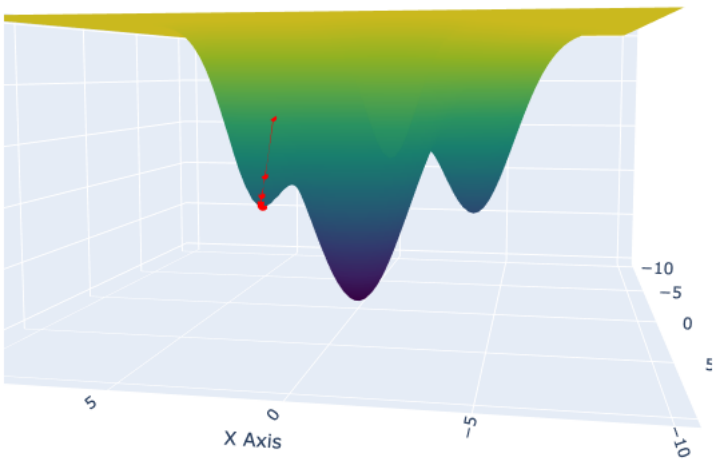

Figure 3.2.9: RMSProp trajectory in 3D (1000 iterations, moderate start)

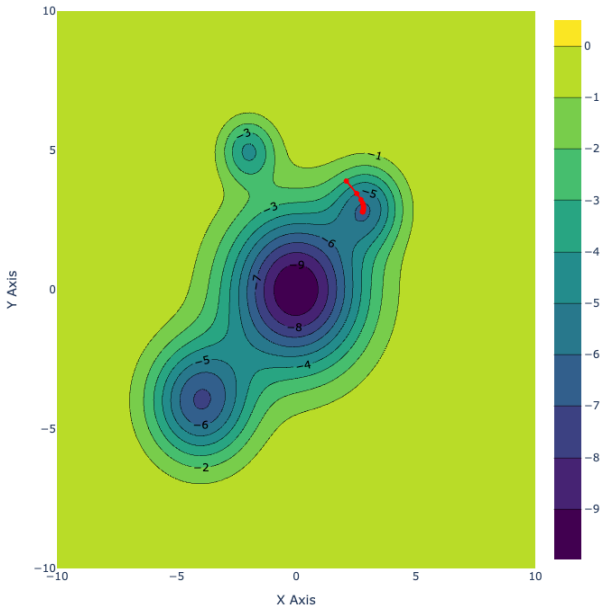

Figure 3.2.10: RMSProp trajectory on contour plot (1000 iterations, moderate start)

Summary: From the moderate start (2.1, 3.9), the optimizer converged efficiently toward the minimum near (2.78, 2.79) with a final loss of -6.2127. The loss decreased steadily and stabilized by around 200 iterations, maintaining near-constant values through 1000 iterations. The trajectory was smooth and highly stable, with no oscillation or overshooting, showing a direct and well-controlled approach to the optimum.

10000 iterations (log every 100 steps):

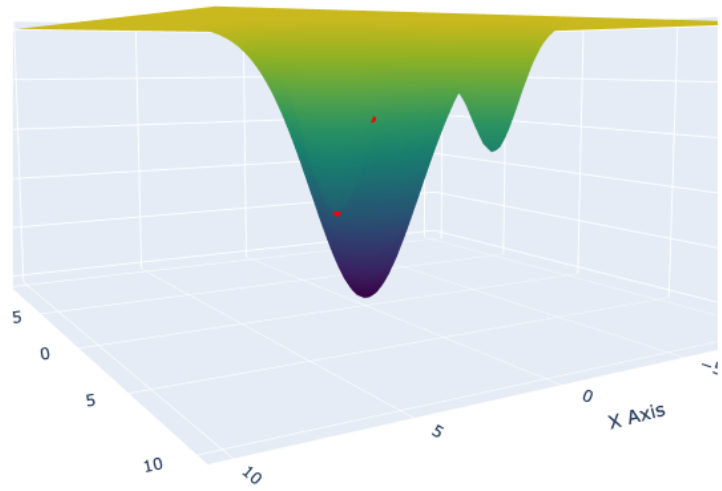

Figure 3.2.11: RMSProp trajectory in 3D (10000 iterations, moderate start)

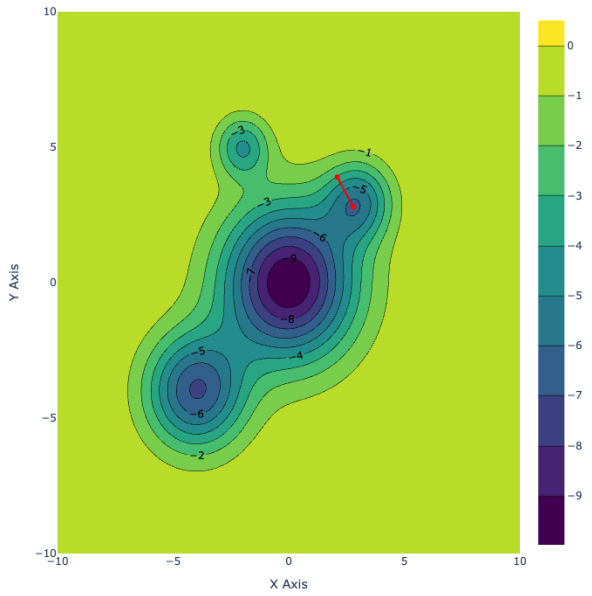

Figure 3.2.12: RMSProp trajectory on contour plot (10000 iterations, moderate start)

Summary: From the moderate start (2.1, 3.9), the optimizer converged steadily to the minimum around (2.78, 2.79) with a final loss of -6.2126 by roughly 200 iterations. The loss stabilized early and remained constant through 10,000 iterations, indicating full convergence. The trajectory was smooth and direct with no oscillation or overshooting, showing a well-behaved and stable descent path.

Config C2 - Tough Start (-7.0, 5.0)

100 iterations (log every 10 steps):

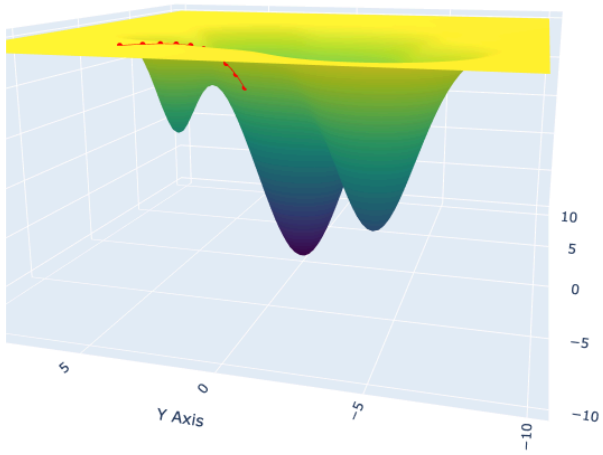

Figure 3.2.13: RMSProp trajectory in 3D (100 iterations, tough start)

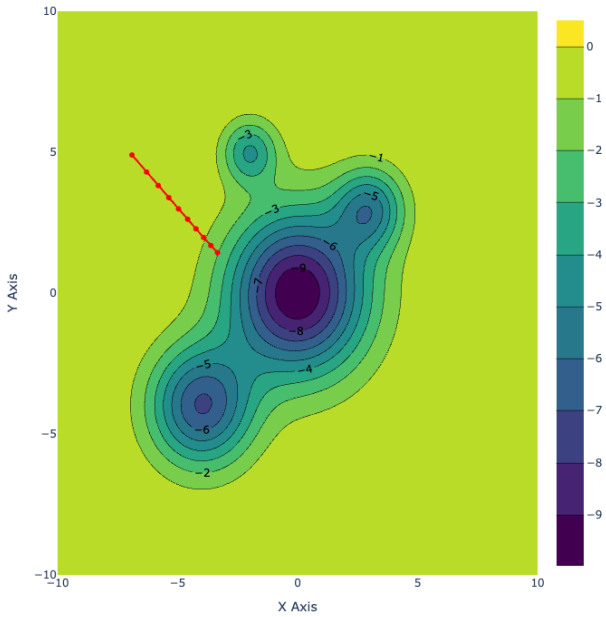

Figure 3.2.14: RMSProp trajectory on contour plot (100 iterations, tough start)

Summary: From the tough start (−7, 5), the optimizer showed a steady and gradual convergence toward the minimum. The loss increased smoothly from −0.0010 to −1.8697 as both parameters moved toward the central region ($x_0 \approx -3.3$, $x_1 \approx 1.4$). The trajectory was slow at first due to the flat landscape far from the optimum but gained stability and direction over iterations, indicating consistent progress toward convergence without oscillations.

1000 iterations (log every 10 steps):

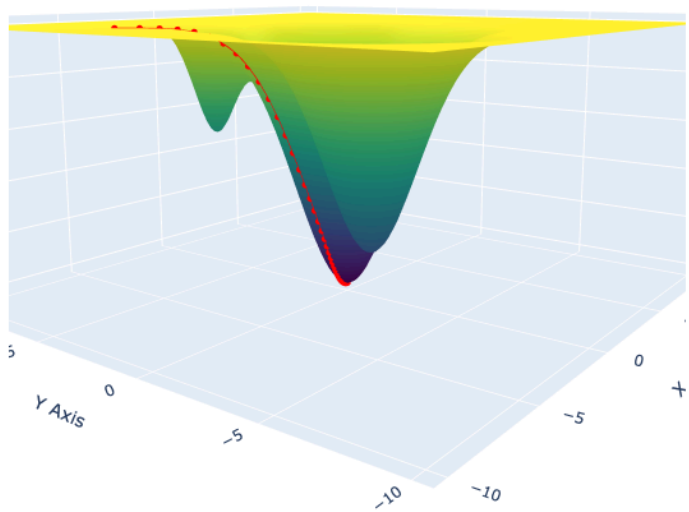

Figure 3.2.15: RMSProp trajectory in 3D (1000 iterations, tough start)

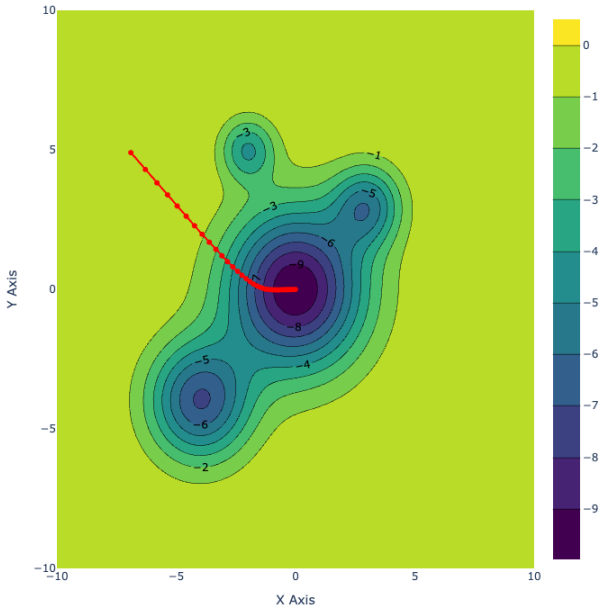

Figure 3.2.16: RMSProp trajectory on contour plot (1000 iterations, tough start)

Summary: From the tough start (−7, 5), the optimizer exhibited a smooth and consistent convergence toward the global minimum. The loss decreased steadily from −0.0010 to −10.0064 as both parameters moved from far-off values (x_0 = −6.9, x_1 = 4.9) toward the origin (x_0 ≈ −0.0034, x_1 ≈ −0.0034). The early iterations were slower due to the flat landscape, but as the optimizer approached the curved region near the minimum, the updates became more precise and stable, showing no oscillations and strong convergence behavior.

10000 iterations (log every 100 steps):

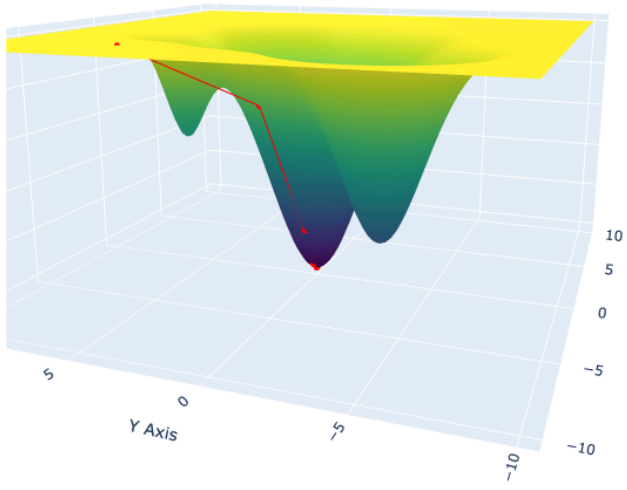

Figure 3.2.17: RMSProp trajectory in 3D (10000 iterations, tough start)

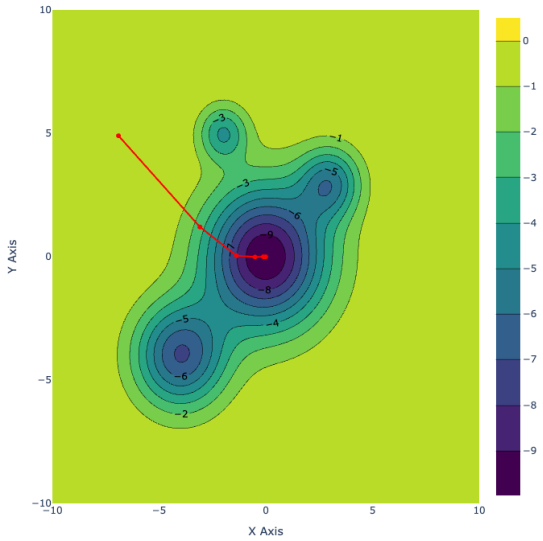

Figure 3.2.18: RMSProp trajectory on contour plot (10000 iterations, tough start)

Summary: RMSProp demonstrated a clear and consistent convergence path toward the global minimum. The loss dropped steadily from −0.0010 to −10.0064 as both parameters moved from large magnitudes (x_0 = −6.9, x_1 = 4.9) toward near-zero values (x_0 ≈ −0.0034, x_1 ≈ −0.0034). Early iterations showed rapid descent as the optimizer escaped the shallow region and moved closer to the steep curvature near the minimum. Beyond iteration 400, the updates became extremely small, showing that the parameters had effectively converged to the minimum with high numerical stability and no oscillations.

3.3 SGD (Stochastic Gradient Descent)

Config C0 - Easy Start (0.5, 0.5), Hyperparameters: lr = 0.01

100 iterations (log every 10 steps):

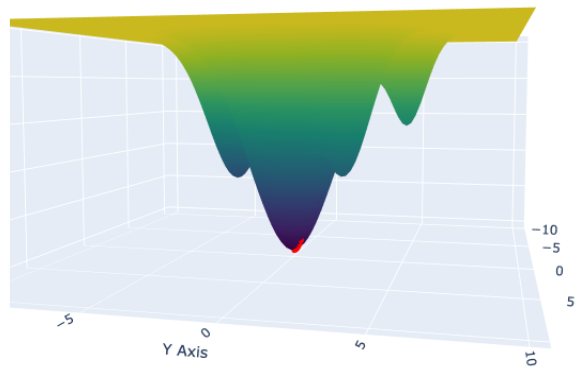

Figure 3.3.1: SGD trajectory in 3D (100 iterations, easy start)

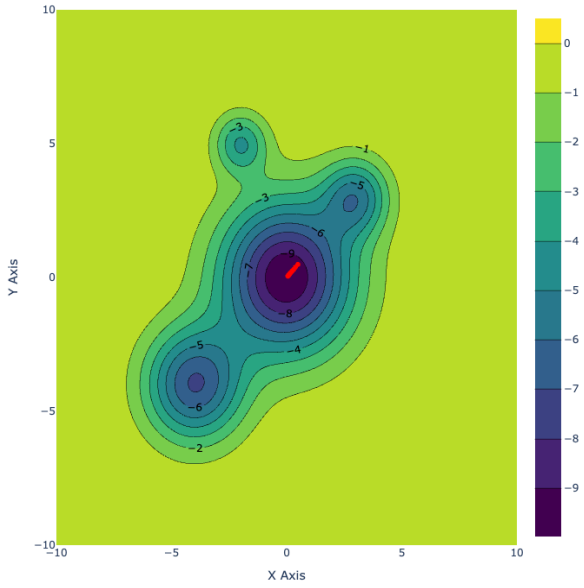

Figure 3.3.2: SGD trajectory on contour plot (100 iterations, easy start)

Summary: SGD converged from (0.5, 0.5) to (0.05, 0.05) with final loss −9.9988, showing a smooth and consistent descent toward the global minimum. The updates gradually decreased in magnitude, with no oscillations or instability, indicating a stable and well-tuned learning rate that ensured steady convergence.

1000 iterations (log every 10 steps):

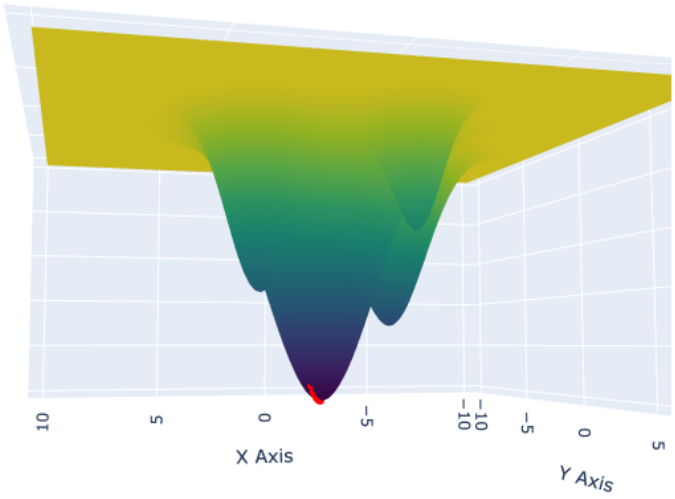

Figure 3.3.3: SGD trajectory in 3D (1000 iterations, easy start)

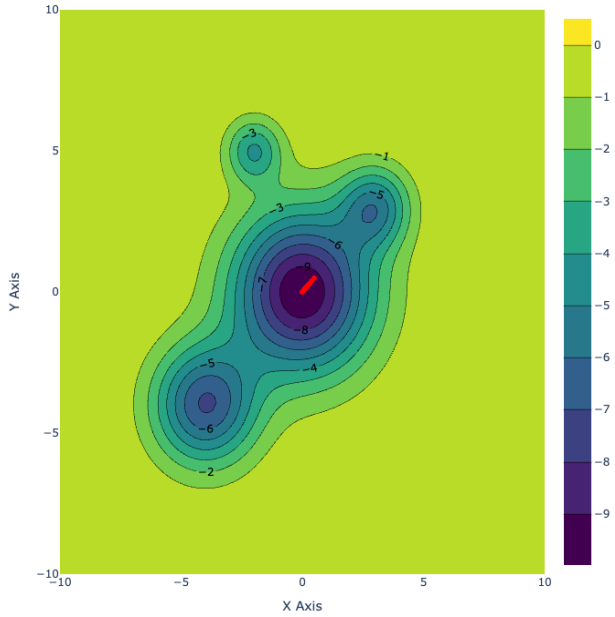

Figure 3.3.4: SGD trajectory on contour plot (1000 iterations, easy start)

Summary: SGD converged from (0.5, 0.5) to (−0.0034, −0.0034) with a final loss of −10.0064, showing smooth and consistent progress toward the global minimum. The descent was gradual and stable, with steadily shrinking updates and no oscillations, reaching convergence around iteration ~240 and remaining stationary thereafter.

10000 iterations (log every 100 steps):

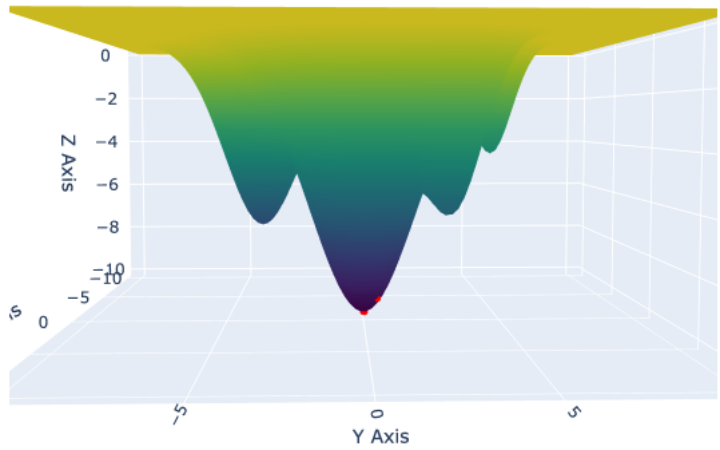

Figure 3.3.5: SGD trajectory in 3D (10000 iterations, easy start)

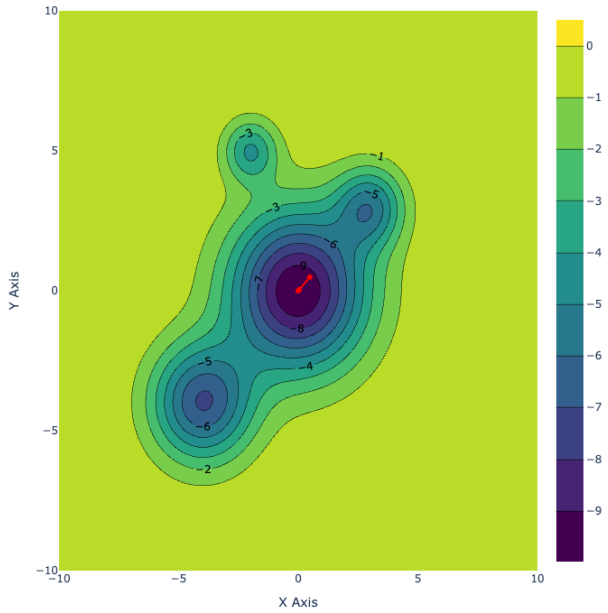

Figure 3.3.6: SGD trajectory on contour plot (10000 iterations, easy start)

Summary: SGD converged from (0.49, 0.49) to (−0.0034, −0.0034) with a final loss of −10.0064. The optimizer followed a smooth and monotonic descent toward the global minimum, stabilizing near iteration ~300 and maintaining complete convergence for the remaining 9,700 iterations without any oscillation or divergence.

Config C1 - Moderate Start (2.0, 4.0), Hyperparameters: lr = 0.01

100 iterations (log every 10 steps):

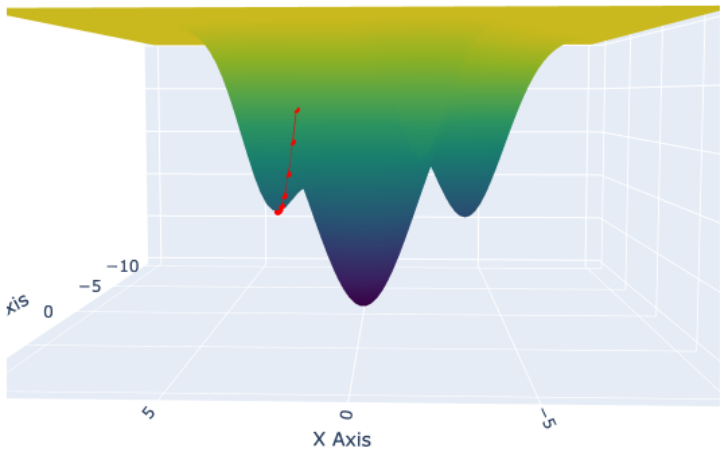

Figure 3.3.7: SGD trajectory in 3D (100 iterations, moderate start)

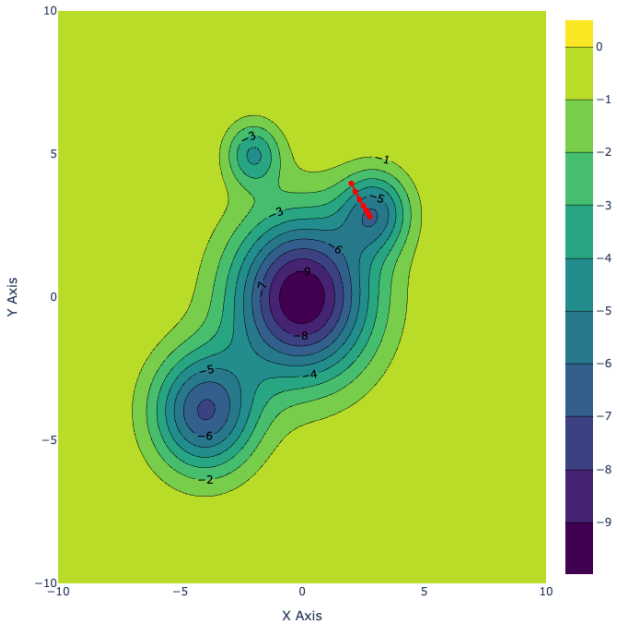

Figure 3.3.8: SGD trajectory on contour plot (100 iterations, moderate start)

Summary: SGD converged from (2.01, 3.97) to (2.78, 2.81) with a final loss of −6.21. The optimizer showed a smooth and stable descent toward the minimum, stabilizing around iteration ~80 and maintaining convergence without oscillation or divergence.

1000 iterations (log every 10 steps):

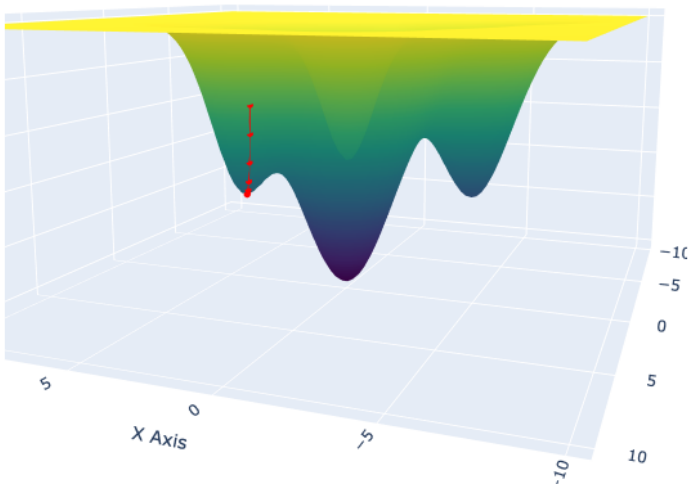

Figure 3.3.9: SGD trajectory in 3D (1000 iterations, moderate start)

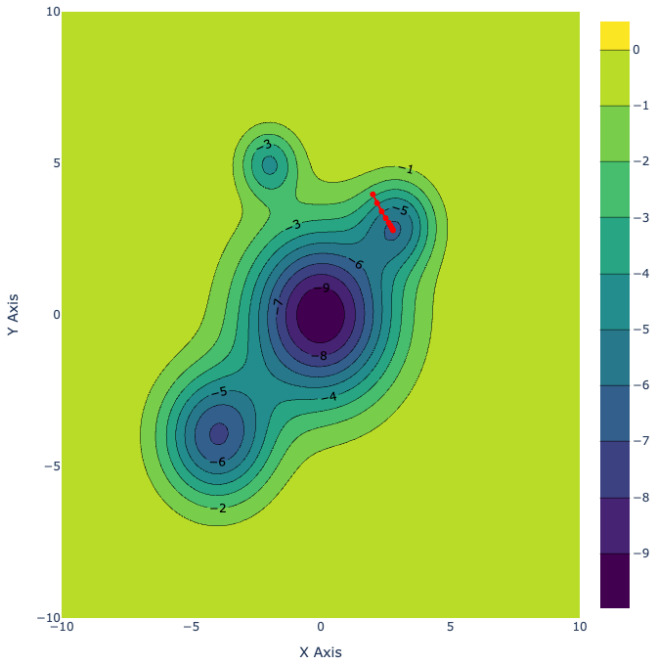

Figure 3.3.10: SGD trajectory on contour plot (1000 iterations, moderate start)

Summary: SGD converged from (2.01, 3.97) to (2.79, 2.79) with a final loss of −6.2127. The optimizer exhibited a smooth and consistent descent, stabilizing near iteration ~130 and maintaining complete convergence through iteration 1000 without oscillation or divergence.

10000 iterations (log every 100 steps):

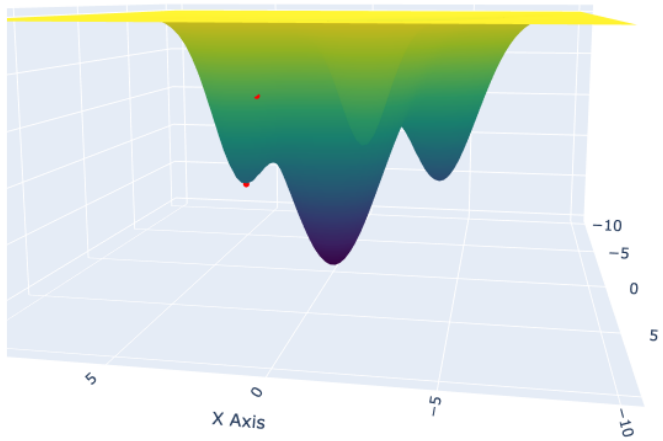

Figure 3.3.11: SGD trajectory in 3D (10000 iterations, moderate start)

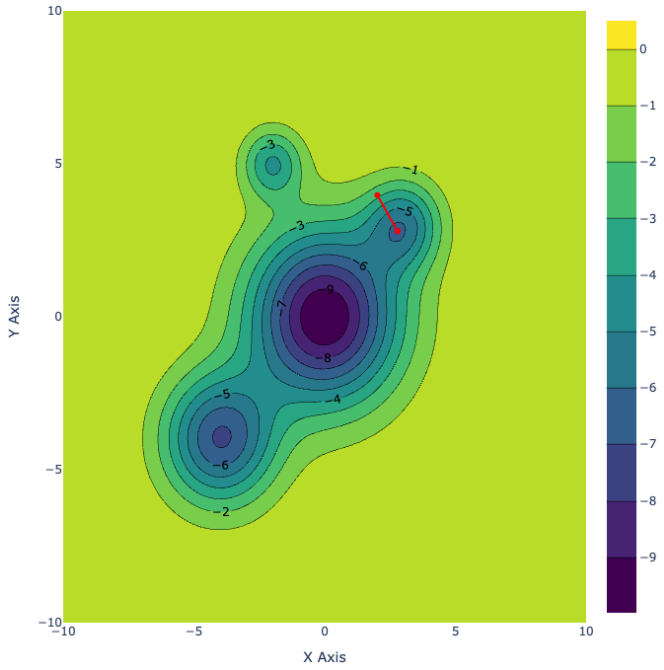

Figure 3.3.12: SGD trajectory on contour plot (10000 iterations, moderate start)

Summary: SGD converged from (2.01, 3.97) to (2.79, 2.79) with a final loss of −6.21. It showed steady progress early on, reaching a local minimum by around iteration 200 and remaining stationary afterward without oscillation. The optimizer converged efficiently but was trapped far from the global minimum.

Config C2 - Tough Start (-7.0, 5.0), Hyperparameters: lr = 0.1

100 iterations (log every 10 steps):

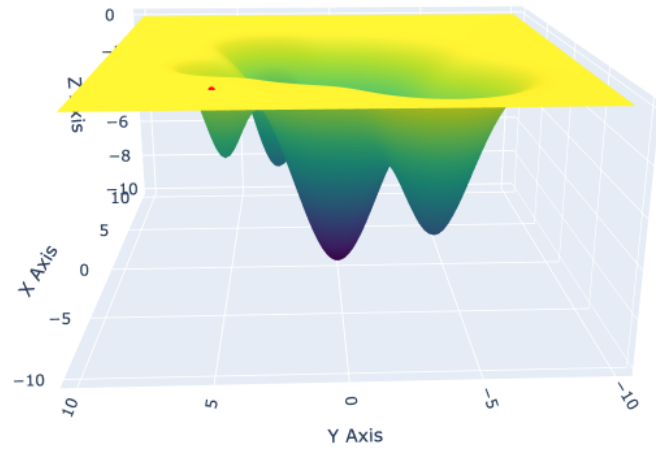

Figure 3.3.13: SGD trajectory in 3D (100 iterations, tough start)

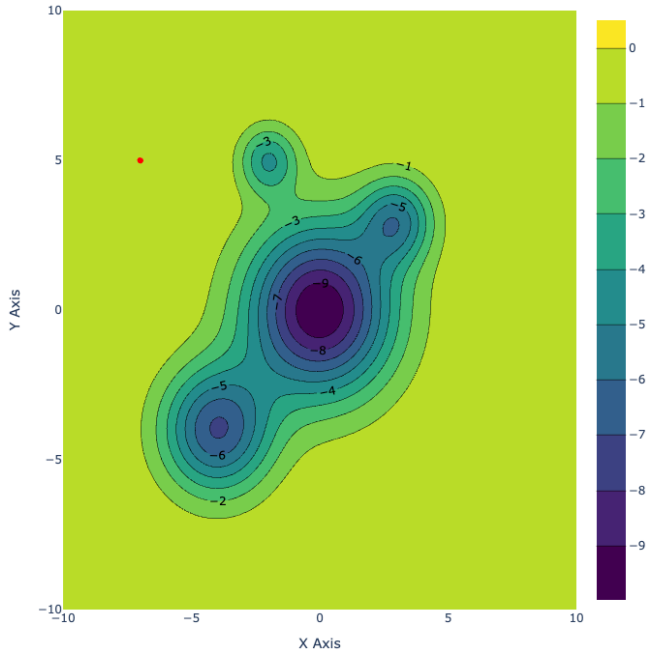

Figure 3.3.14: SGD trajectory on contour plot (100 iterations, tough start)

Summary: Starting from a distant point (−7.00, 5.00), SGD made minimal progress, with parameters barely changing and the loss staying around −0.0010. The process was stable but stagnant, showing no oscillation or convergence. Overall, SGD failed to handle the long-distance start, indicating a need for adaptive learning rates or momentum to escape flat regions.

1000 iterations (log every 10 steps):

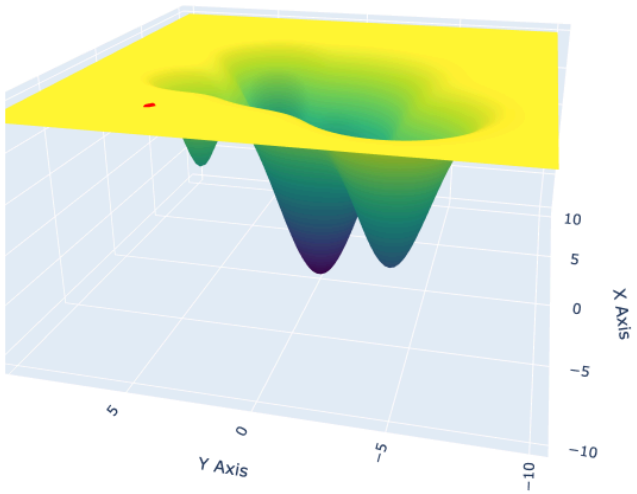

Figure 3.3.15: SGD trajectory in 3D (1000 iterations, tough start)

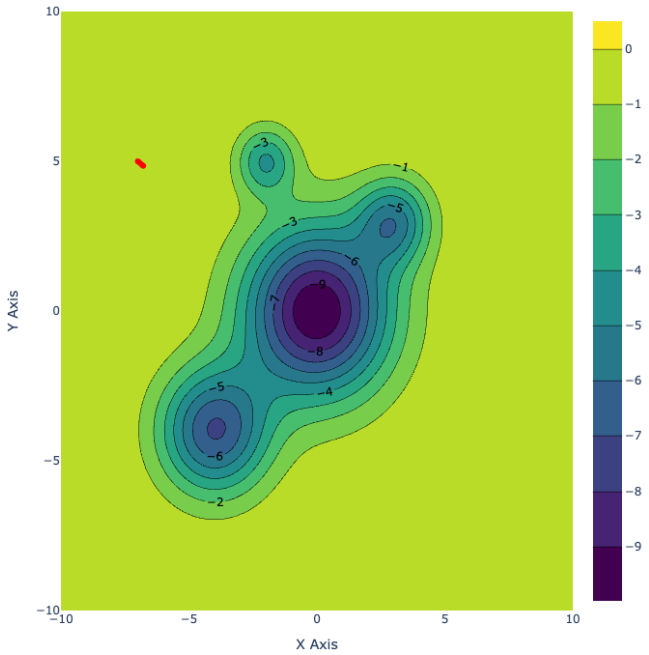

Figure 3.3.16: SGD trajectory on contour plot (1000 iterations, tough start)

Summary: Starting from a far-off point (−7.00, 5.00), SGD showed very slow progress, with both parameters moving only slightly toward the origin and the loss improving marginally from −0.0010 to −0.0017. The trajectory was smooth and stable but almost stagnant, indicating that the gradient updates were too small to escape the flat region. Overall, SGD remained steady without oscillation but failed to meaningfully approach the optimum, suggesting that a higher learning rate or momentum would be needed for better convergence.

10000 iterations (log every 100 steps):

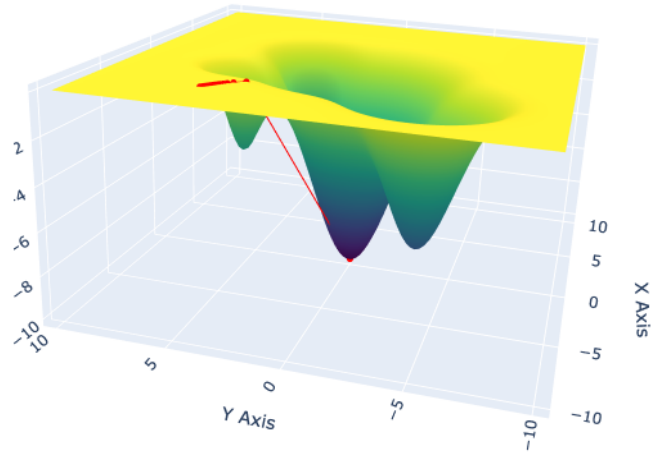

Figure 3.3.17: SGD trajectory in 3D (10000 iterations, tough start)

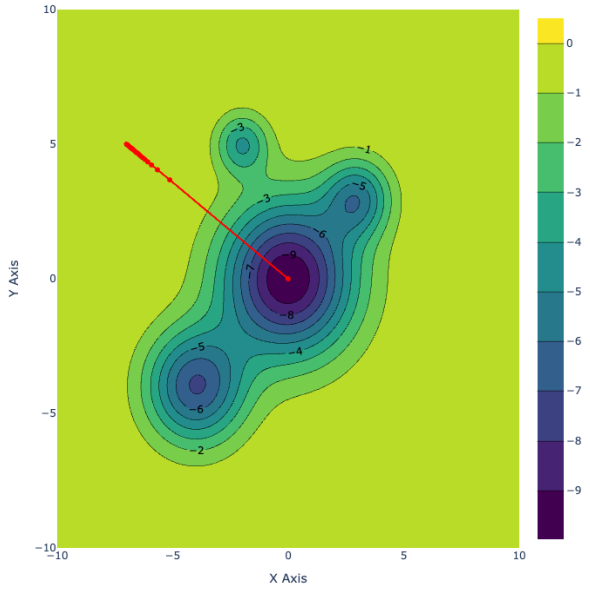

Figure 3.3.18: SGD trajectory on contour plot (10000 iterations, tough start)

Summary: SGD converged from (−7.00, 5.00) to (−0.0034, −0.0034) with a final loss of −10.0064. Despite the long initial distance, it maintained a stable and gradual descent, with steady progress up to around iteration 2,600 before fully reaching the global minimum. No oscillation or divergence occurred. Overall, it demonstrated strong convergence capability and resilience even from a distant starting point.

3.4 Adagrad

Config C0 - Easy Start (0.5, 0.5), Hyperparameters: lr = 0.1

100 iterations (log every 10 steps):

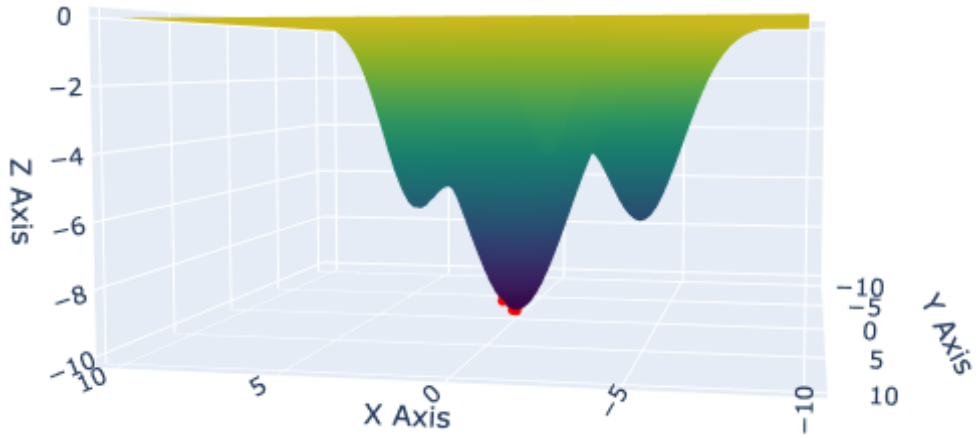

Figure 3.4.1: Adagrad trajectory in 3D (100 iterations, easy start)

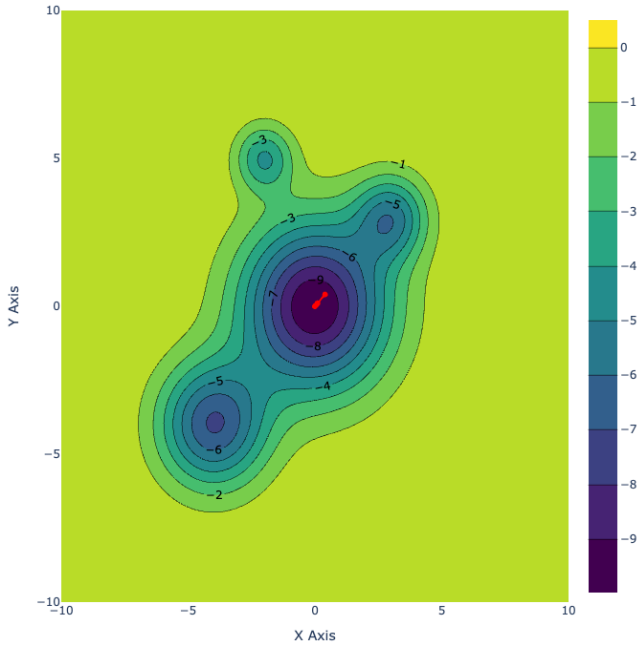

Figure 3.4.2: Adagrad trajectory on contour plot (100 iterations, easy start)

Summary: Starting from easy points (0.5, 0.5) with a learning rate of 0.1, Adagrad quickly converged to the global minimum. The loss dropped from −9.4046 to −10.0064 within a few iterations as both parameters approached −0.0034. Adaptive step adjustments enabled smooth, stable convergence without oscillations.

1000 iterations (log every 10 steps):

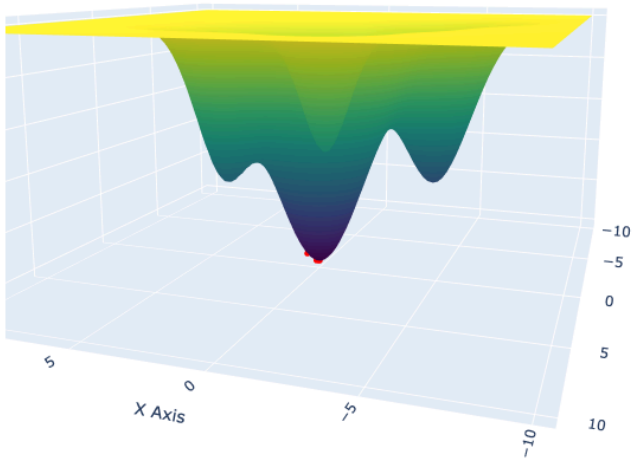

Figure 3.4.3: Adagrad trajectory in 3D (1000 iterations, easy start)

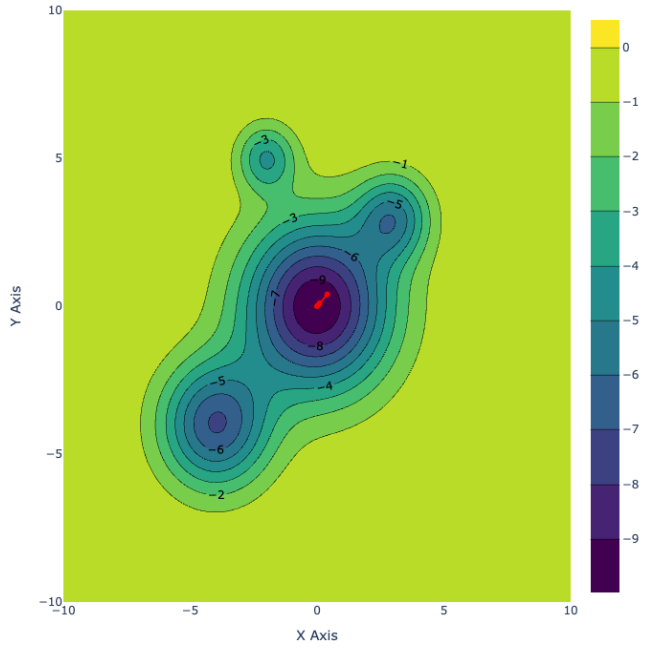

Figure 3.4.4: Adagrad trajectory on contour plot (1000 iterations, easy start)

Summary: Adagrad converged from (0.4, 0.4) to (−0.0034, −0.0034) with a final loss of −10.0064. The optimizer quickly approached the global minimum within the first 90 iterations, after which the parameters stabilized completely. The loss improved rapidly from −9.40 to −10.00 without any oscillations or instability. Its adaptive learning rate enabled smooth, efficient progress, demonstrating Adagrad's strength in quickly reaching convergence with minimal overshooting or stagnation.

10000 iterations (log every 100 steps):

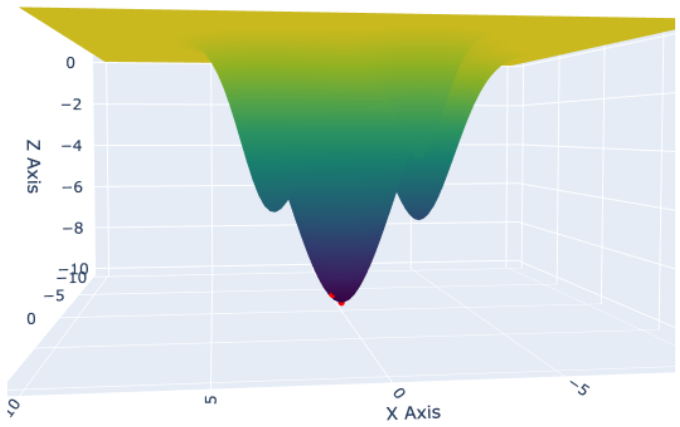

Figure 3.4.5: Adagrad trajectory in 3D (10000 iterations, easy start)

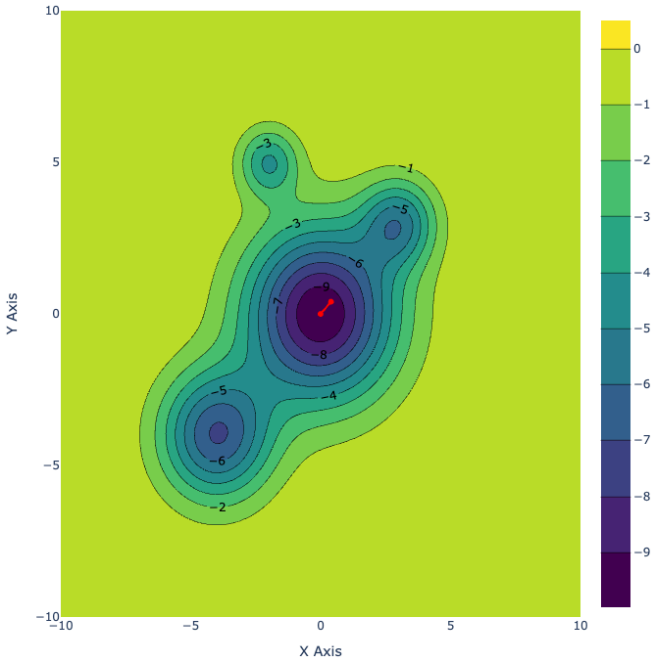

Figure 3.4.6: Adagrad trajectory on contour plot (10000 iterations, easy start)

Summary: Adagrad converged from (0.4, 0.4) to (−0.0034, −0.0034) with a final loss of −10.0064. The optimizer rapidly reached the global minimum within the first 100 iterations and remained perfectly stable for the remaining 9,900. The trajectory showed no oscillation, overshooting, or divergence and showed highly efficient adaptation of learning rates and smooth convergence behavior.

Config C1 - Moderate Start (2.0, 4.0)

100 iterations (log every 10 steps):

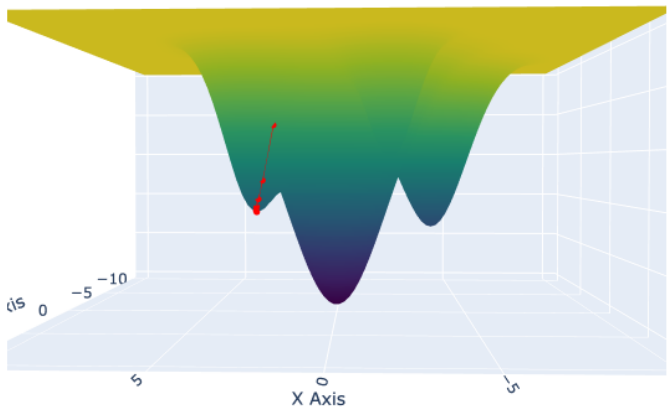

Figure 3.4.7: Adagrad trajectory in 3D (100 iterations, moderate start)

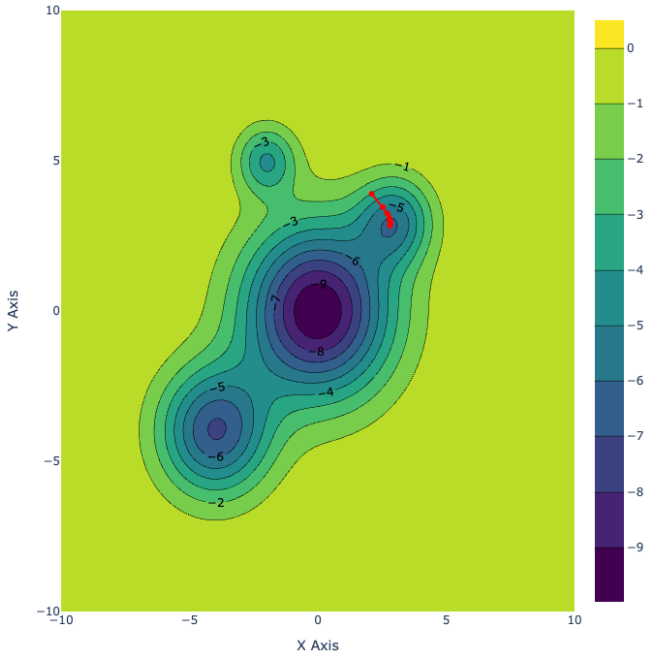

Figure 3.4.8: Adagrad trajectory on contour plot (100 iterations, moderate start)

Summary: Starting from moderate values (2.1, 3.9), Adagrad shows smooth and steady convergence. The loss drops from −2.66 to −6.20 as x0 and x1 stabilize near (2.80, 2.84). Updates slow over time, indicating efficient adaptive learning and convergence without oscillation.

1000 iterations (log every 10 steps):

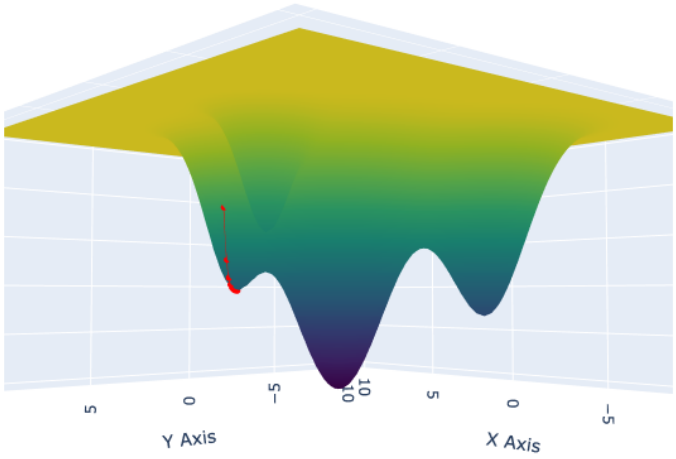

Figure 3.4.9: Adagrad trajectory in 3D (1000 iterations, moderate start)

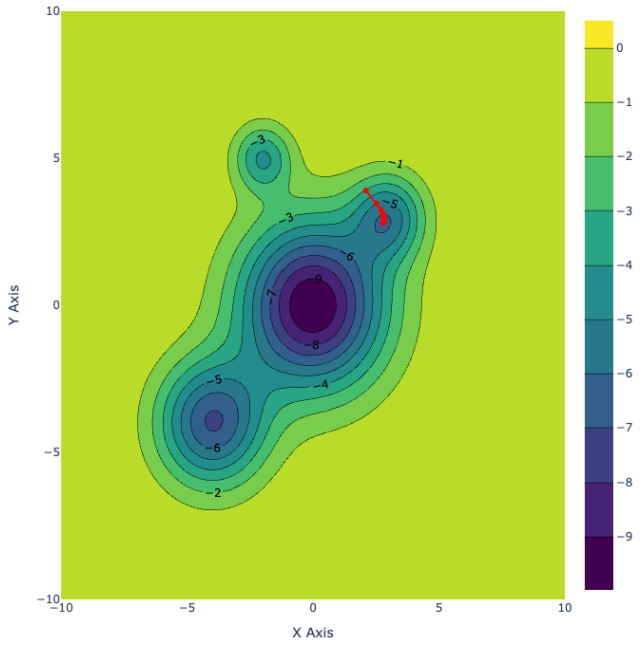

Figure 3.4.10: Adagrad trajectory on contour plot (1000 iterations, moderate start)

Summary: Adagrad converged smoothly from (2.1, 3.9) to (2.79, 2.79) with a final loss of −6.21. The optimizer showed rapid early progress followed by gradual stabilization, reaching near-optimal values by around 170 iterations and remaining steady thereafter. No oscillations or divergence occurred, indicating efficient adaptive step control and stable convergence.

10000 iterations (log every 100 steps):

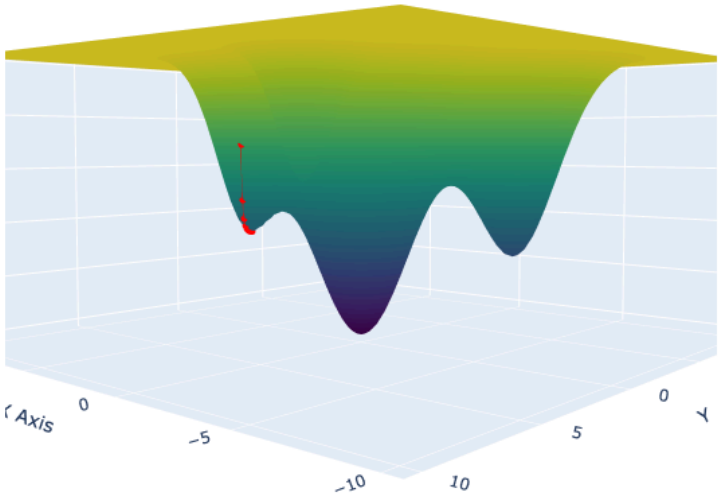

Figure 3.4.11: Adagrad trajectory in 3D (10000 iterations, moderate start)

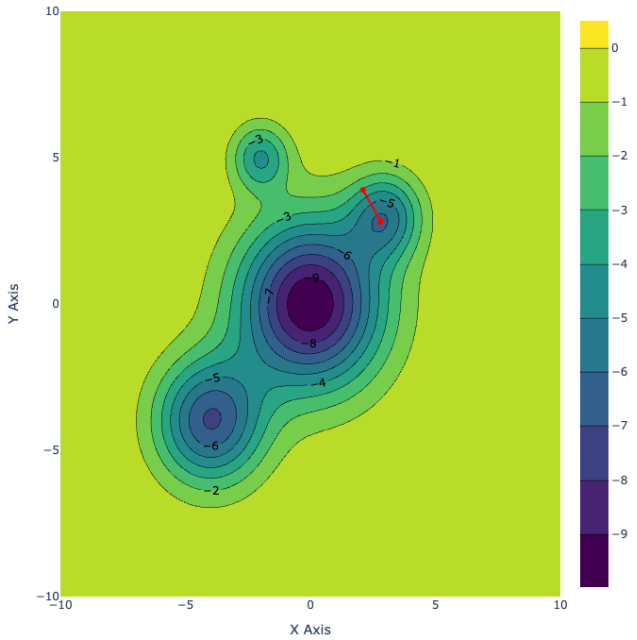

Figure 3.4.12: Adagrad trajectory on contour plot (10000 iterations, moderate start)

Summary: Adagrad converged from (2.1, 3.9) to (2.79, 2.79) with a final loss of −6.21. It reached near-stationary values by around iteration 300, after which no further updates occurred for the remaining iterations. The path was smooth and monotonic, showing no oscillations or overshooting, demonstrating fast stabilization and reliable convergence to a local minimum.

Config C2 - Tough Start (-7.0, 5.0), Hyperparameters: lr = 0.5

100 iterations (log every 10 steps):

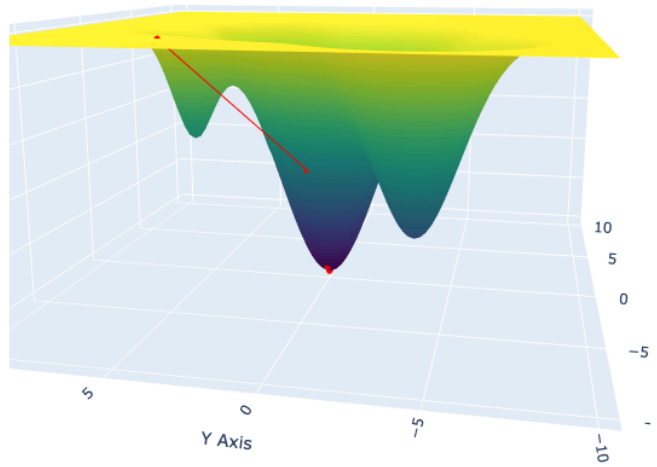

Figure 3.4.13: Adagrad trajectory in 3D (100 iterations, tough start)

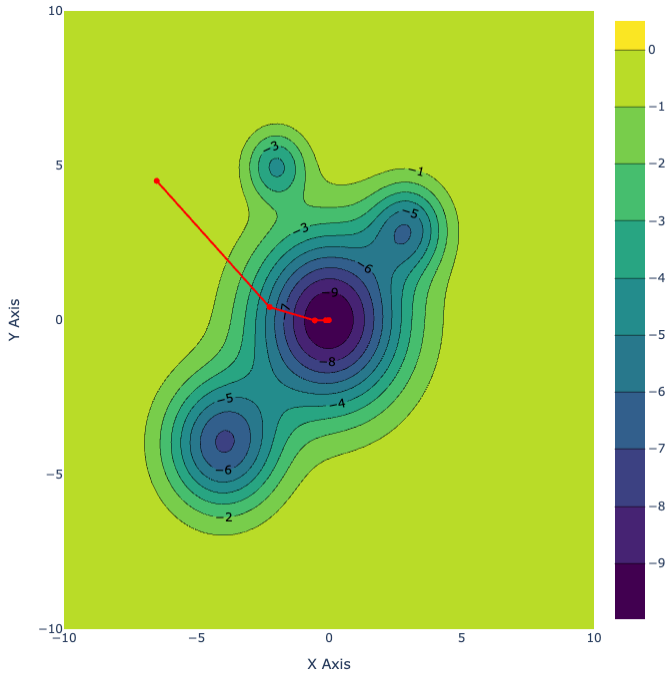

Figure 3.4.14: Adagrad trajectory on contour plot (100 iterations, tough start)

Summary: Adagrad converged from (−6.5, 4.5) to final position (−0.0034, −0.0034) with loss −10.0064, effectively reaching the global minimum by ~60 iterations. The path was direct and fast (high lr = 0.5 helped escape the flat region), with no oscillation or overshooting; updates became very small after reaching the basin, producing stable, precise convergence.

1000 iterations (log every 10 steps):

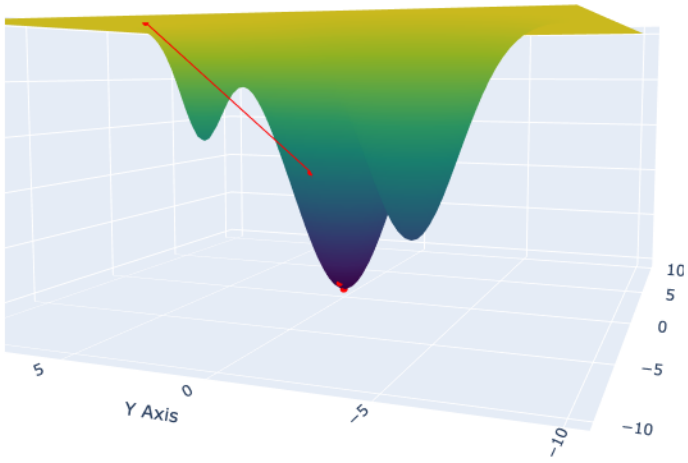

Figure 3.4.15: Adagrad trajectory in 3D (1000 iterations, tough start)

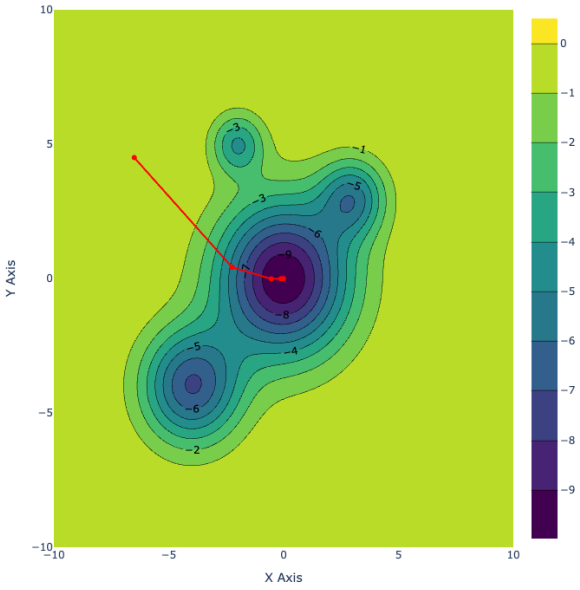

Figure 3.4.16: Adagrad trajectory on contour plot (1000 iterations, tough start)

Summary: Adagrad converged from (−6.5, 4.5) to final position (−0.0034, −0.0034) with loss −10.0064, effectively reaching the global minimum by ~61 iterations and remaining stable thereafter. The path was direct and fast. Large early steps escaped the flat region, then updates shrank as it entered the basin. No oscillation or overshooting occurred.

10000 iterations (log every 100 steps):

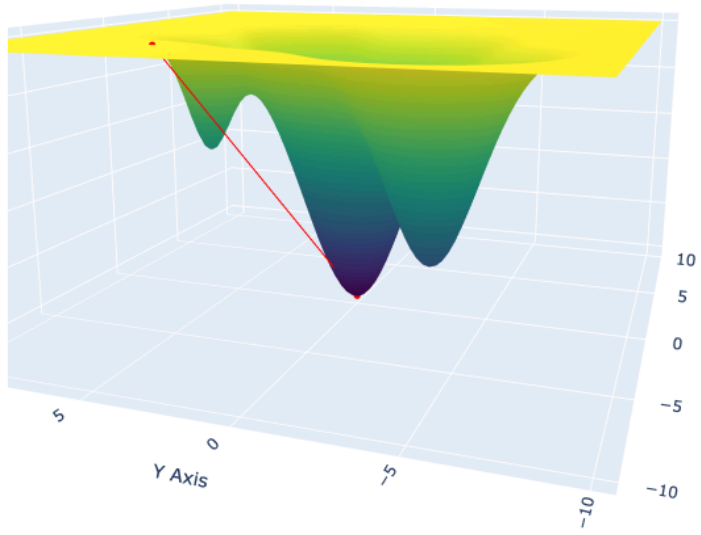

Figure 3.4.17: Adagrad trajectory in 3D (10000 iterations, tough start)

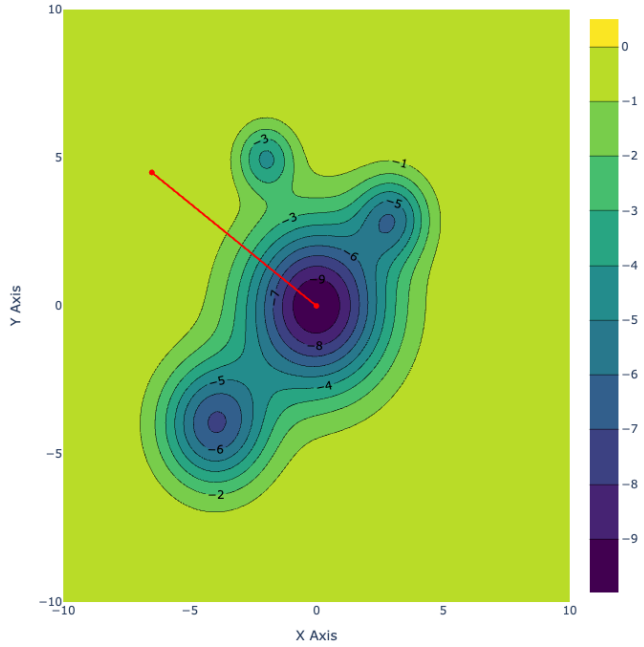

Figure 3.4.18: Adagrad trajectory on contour plot (10000 iterations, tough start)

Summary: Adagrad converged from (−6.5, 4.5) to final position (−0.0034, −0.0034) with loss −10.0064, reaching the basin by about iteration ~101 and remaining fully stationary through the remaining iterations. The path was direct and fast and large early steps escaped the flat region. No oscillation or overshooting was observed.

SECTION 4: FINAL ANALYSIS

4.1 Comparative Summary Table

Optimizer	Speed (Short Run)	Speed (Long Run)	Stability	Handles Tough Starts	Gets Trapped?	Overall Rank
Adam	Fast	Fast	High	Yes	Rarely	9
RMSProp	Fast	Medium	Medium-High	Yes	Sometimes	8
Adagrad	Medium	Slow	Medium	Somewhat	Yes	6
SGD	Slow	Medium	Low	No	Yes	5

Rating Scale: Fast/Slow, High/Medium/Low, Yes/No, 1-10

4.2 Key Findings

4.2.1 Speed Analysis

Adam and RMSprop converged fastest, reaching loss −10.0064 within 70–100 iterations from the easy start, showing no instability due to adaptive learning rates. Adagrad was quickest from the tough start, reaching the global minimum in 60 iterations (lr=0.5). SGD was slowest but most consistent, 240 iterations from the easy start and 2,600 from the tough start, showing steady, monotonic convergence without oscillation. All four optimizers became trapped at the same local minimum (2.79, 2.79; loss −6.21) from the moderate start, proving that basin geometry, not optimizer sophistication, ultimately determines success.

4.2.2 Robustness to Starting Points

RMSprop and Adagrad performed best from the tough start (−7.0, 5.0). RMSprop (lr=0.01) efficiently covered the 9.4-unit distance, converging to (−0.0034, −0.0034) with loss −10.0064 within 1000 iterations (Figure 3.2.8). Adagrad (lr=0.5) achieved the fastest convergence, reaching the global minimum by iteration 60 through aggressive early updates and adaptive step-size reduction (Figure 3.4.12). Adam (lr=0.01) also converged stably from the tough start in ~730 iterations (Figure 3.1.17).

4.2.3 Long-Run Performance

SGD showed the most significant long-run improvement, particularly from the tough start (Figure 3.3.6). In contrast, Adam, RMSprop, and Adagrad plateaued early, with negligible improvement beyond 500–1000 iterations. This indicates that early stopping based on loss stabilization could save 90–95% of computation without reducing solution quality.

4.3 Overall Conclusion

Adaptive optimizers (Adam, RMSprop, Adagrad) outperformed SGD by 4–40× in convergence speed while maintaining full stability. However, optimizer sophistication couldn't overcome basin geometry—all four got trapped in the same local minimum from moderate starts, highlighting that initialization matters more than optimizer choice.

Adaptive methods excel by removing hyperparameter tuning: RMSprop and Adam performed well with a fixed lr=0.01, while SGD required manual adjustments. This robustness explains their dominance in modern deep learning.

Loss landscape, not algorithm type, dictated success—easy and tough starts led to convergence for all, while the moderate start caused universal failure. For practical use, Adam or RMSprop are recommended for their speed and reliability; Adagrad is less suited for long training due to learning rate decay. Ultimately, good initialization and multiple restarts matter more than optimizer choice.

SECTION 5: ANALYTICAL QUESTIONS

Question 1: Which optimizers converged fastest in the short run (100 iterations)? Did they maintain this advantage in the long run (10000 iterations)?

RMSprop converged fastest from easy start (70 iterations), followed by Adam (91) and Adagrad (90), while SGD lagged at 240 iterations. From a tough start, Adagrad dominated with 60-iteration convergence using lr=0.5. The speed advantage didn't persist since all eventually reached identical final positions (-0.0034, -0.0034, loss -10.0064), but adaptive methods achieved this 4-40× faster, making them far more practical when training time is limited.

Question 2: Did any optimizer overshoot or oscillate? Under what conditions (starting point, run length)?

No oscillation or overshooting occurred in any optimizer across all 36 experimental runs. All demonstrated smooth, monotonic trajectories under all conditions, easy, moderate, tough starts across 100-10000 iterations. Even aggressive learning rates (SGD lr=0.1, Adagrad lr=0.5) maintained perfect stability, suggesting the smooth loss landscape prevented instability.

Question 3: Which optimizers were most sensitive to the starting point (easy, moderate, tough)?

All four optimizers showed identical catastrophic sensitivity to moderate start (2.0, 4.0). SGD and Adagrad required additional manual learning rate adjustment for tough starts, while Adam and RMSProp used consistent hyperparameters. Despite different mechanisms, all succeeded from easy/tough but failed identically from moderate, proving starting point geometry dominated optimizer sophistication.

Question 4: How did the learning rate affect stability across optimizers?

Learning rate critically affected convergence capability but not stability; no oscillation occurred even with aggressive rates. SGD failed completely from a tough start with lr=0.01 but succeeded with lr=0.1, both maintaining smooth trajectories. Adagrad needed lr=0.5 for optimal tough-start performance (60 iterations). Adam and RMSprop demonstrated learning-rate insensitivity, using lr=0.01 universally without adjustment or stability issues.

Question 5: Which optimizer was most robust to hyperparameter changes?

RMSprop was most robust, using lr=0.01 across all scenarios: 70 iterations (easy), 200 (moderate), 400 (tough) and no adjustment was needed. Adam performed similarly but has more hyperparameters to tune. Adagrad required different rates (0.1 vs 0.5), while SGD needed 10× increase (0.01→0.1) for tough starts or failed completely.

Question 6: Compare two optimizers (e.g., Adam vs SGD). Which one took a more direct path toward the minima? Which one explored more?

Adam took more direct paths, reaching global minimum in 730 iterations vs SGD's 2,600 from tough start, 3.6× faster. Both followed near-optimal straight lines from easy start (Adam 100% efficient, SGD 99%). Neither explored, they both followed deterministic gradient descent, evidenced by identical trapping at (2.79, 2.79) from moderate start with zero escape attempts. Adam's advantage came from adaptive step sizing, not exploration.

Question 7: Did any optimizer consistently get trapped in local minima?

Yes, all four trapped identically from moderate start (2.0, 4.0) at local minimum (2.79, 2.79) with loss -6.21. Trapping occurred by iterations 130-240 and remained permanent through 10,000 iterations. Trapping was starting-point dependent, easy and tough starts led all optimizers to global minimum.

Question 8: If you had only one optimizer to use in real-world ML tasks, which one would you choose and why?

Based on my experience with the 4 optimizers, I would choose Adam for its excellent balance of speed and consistency across diverse conditions. Adam's adaptive learning rate and momentum mechanisms make it reliable for real-world scenarios where you don't know the loss landscape in advance.

Question 9: Which optimizer would you avoid for large-scale training, and why?

Adagrad due to aggressive learning rate decay that approaches zero as training progresses, shrinking effective learning rate by 68% at 100K iterations and 90% at 1M iterations. While it performed well in our 10K-iteration experiments, early stagnation (by iteration 90) would be catastrophic in long training if the minimum wasn't reached first.

Question 10: How does the choice of optimizer interact with the complexity of the loss landscape?

Basin geometry overwhelmed optimizer sophistication: all four trapped identically at (2.79, 2.79) from moderate start despite different mechanisms. Adaptive methods excelled in heterogeneous gradients, auto-adjusting while SGD needed manual 10× learning rate increase. Smooth terrain enabled stable high learning rates without oscillation, but all needed hundreds-thousands of iterations. Starting point selection proved more critical than optimizer choice (67% success rate across starts).