Foundations of Momentum Methods

Fatim Majumder

March 25, 2025

Department of Mathematics and Computer Science

Motivation: Challenges of Gradient Descent in Elongated Contours

Exploring Geometric Properties

To understand the effectiveness of the momentum method, let's examine gradient descent on a challenging objective function. Consider $f(\mathbf{x}) = x_1^2 + 2x_2^2$, which forms a moderately distorted ellipsoid. We enhance this distortion by stretching it along the x_1 axis:

$$f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2. \tag{1}$$

This function has its minimum at $(0, 0)$ and is extremely flat in the x_1 direction. We will observe the behavior of gradient descent on this function using a learning rate of 0.4.

Visualizing Gradient Descent on a Distorted Objective

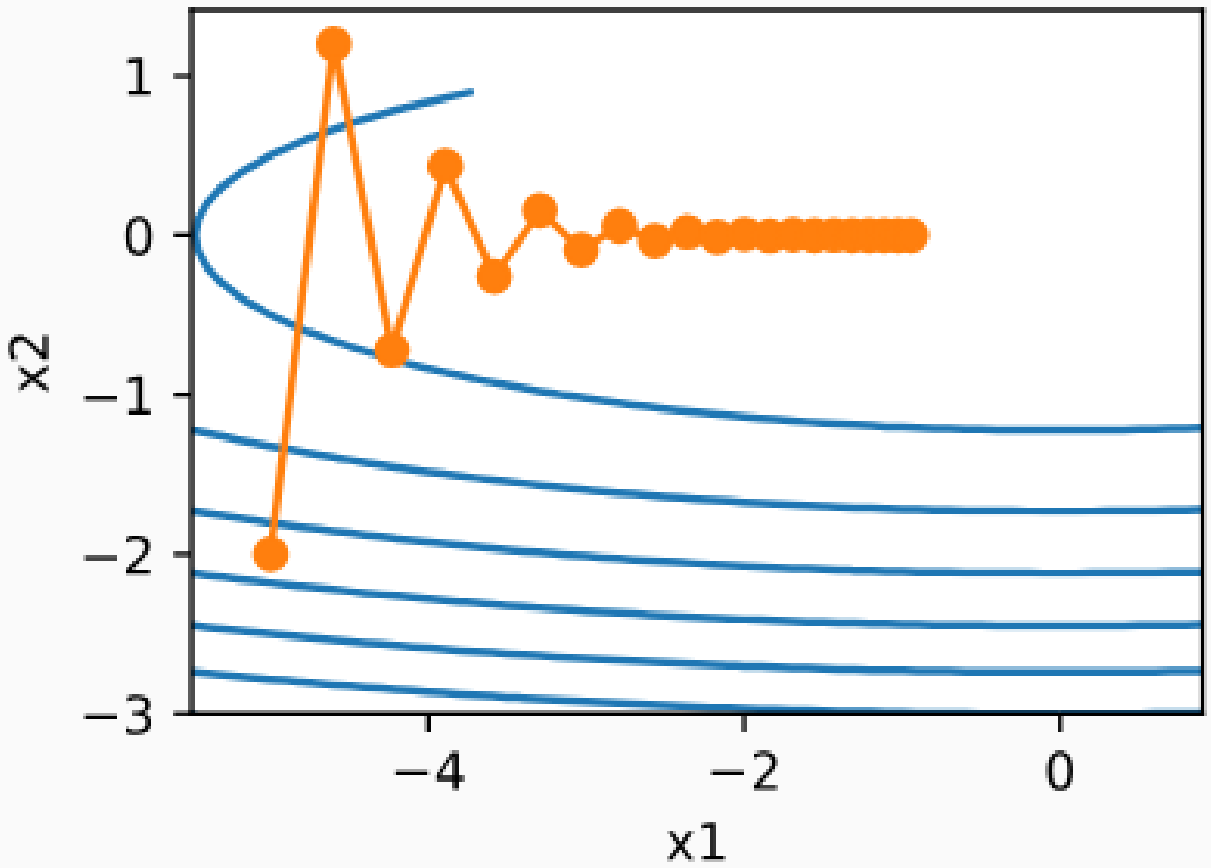


Figure 1: Gradient descent trajectory on a distorted ellipsoid objective.

Challenges of Slow Convergence

Addressing the challenge of slow convergence in gradient descent requires a delicate balance in the learning rate, especially in scenarios with differing gradient steepness:

- **Steepness Disparity:** The gradient in the x_2 direction is much steeper compared to the x_1 direction.
- **Small Learning Rate:** A smaller learning rate can prevent divergence in the steeper x_2 direction but results in slow progress along x_1.
- **Large Learning Rate:** Increasing the learning rate may speed up convergence in x_1, but it introduces the risk of instability in the x_2 direction.
- **Visual Demonstration:** The following figure demonstrates the effects of changing the learning rate from 0.4 to 0.6, highlighting these challenges.

Impact of Adjusted Learning Rate

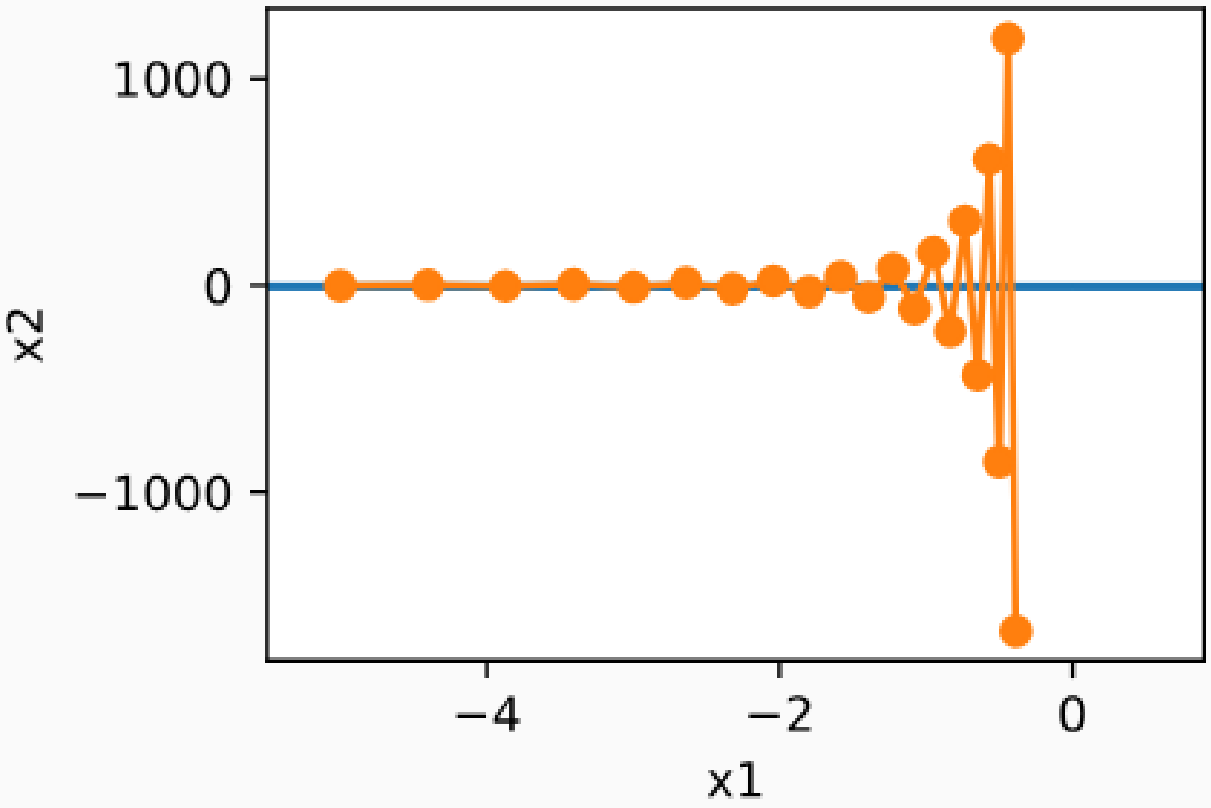


Figure 2: Effect of a higher learning rate on convergence paths.

Understanding the Issue with Elongated Contours

Gradient descent faces significant challenges in certain scenarios, particularly when dealing with cost functions having minima in long, narrow valleys. Consider the following:

- **Cost Function Landscape:** These valleys are typically represented by two-dimensional quadratic functions. They have elongated elliptical contours.
- **Inefficient Gradient Steps:** A standard step in gradient descent (solid red) often proves ineffective in these landscapes.
- **Perpendicular Gradients:** The gradient at any point w^k is perpendicular to the contour line at that point.
- **Deviation from Optimal Path:** Consequently, the gradient direction significantly deviates from the most direct path to the minimum. The optimal path, shown in dashed black, leads straight to the valley's center.

Visualizing Gradient Descent in Elongated Contours

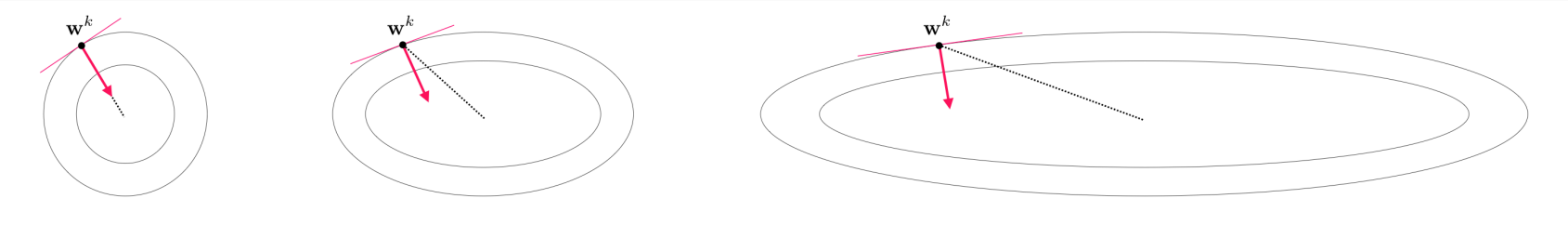


Figure 3: Sequential illustrations from left to right show that in more elongated contours of a two-dimensional quadratic, the gradient descent direction (in red) increasingly diverges from the optimal path (dashed black) that connects w^k to the minimum at the center.

Theoretical Background

Historical Foundations of Momentum in Optimization

- **Classical Mechanics Roots:** Momentum in optimization traces back to principles in classical mechanics, with the analogy to optimization algorithms emerging in the late 20th century.
- **Boris Polyak's Contribution:** In 1964, Boris Polyak introduced the Heavy-Ball method. This method was a pioneering effort to integrate the concept of momentum into the field of optimization.
- **Inspiration from Physics:** The Heavy-Ball method drew inspiration from the physical principle of momentum, where an object in motion continues its trajectory unless acted upon by external forces.
- **Optimization and Physics:** Similar to its physical counterpart, the momentum term in optimization aims to accelerate convergence and navigate complex landscapes.

Momentum in Optimization: A Differential Equations Perspective

- **Differential Equations Context:** Momentum-based methods in optimization are framed in the context of differential equations, a foundation that dates back to the foundational work in dynamical systems theory.
- **Historical Mathematical Foundations:** These methods share parallels with the mathematical formulations of Isaac Newton and Leonhard Euler, key figures of the 17th and 18th centuries.
- **Representation in Modern Optimization:**
  - The behavior of momentum in optimization algorithms is described using differential equations akin to those in dynamical systems.
  - *System Dynamics:* $x_{t+1} = x_t + \Delta t \cdot v_t$
  - *Momentum Dynamics:* $v_{t+1} = v_t + \Delta t \cdot (-\nabla f(x_t))$

The Heavy-Ball Method: A Landmark in Momentum-Based Optimization

- **Origins and Significance:**
  - Developed by Boris Polyak in 1964, the Heavy-Ball method stands as a pivotal innovation in momentum-based optimization.
- **Methodology:**
  - It innovatively combines the current gradient and the previous update vector, enhancing the optimization process's direction and momentum.
- **Mathematical Formulation:**
  - The method is mathematically expressed as:

$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1}) \tag{2}$$

- **Impact in Machine Learning:**
  - The Heavy-Ball method marks a critical junction where mathematical theory and practical machine learning optimization techniques converge.

Evolution of Gradient Descent with Momentum

- **Historical Progression:**
  - The concept of integrating momentum into gradient descent marks a significant development in optimization techniques since the 1960s.
- **Rise in Machine Learning:**
  - These methods gained prominence with the introduction of backpropagation in the 1980s, a pivotal moment in machine learning advanced by Rumelhart, Hinton, and Williams.
- **Momentum Update Formula:**
  - The momentum update is a refinement of traditional gradient descent, elegantly combining current and past gradients to improve convergence:

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_t, \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \eta \mathbf{v}_t. \end{aligned} \tag{3}$$

Momentum Methods in Gradient Descent

Momentum-Adjusted Gradient Descent Formulation

- **Momentum Term Integration:**
  - The momentum term augments the gradient descent update by factoring in the direction from the previous step, providing a more informed update path.
- **Updated Formula:**
  - The formula incorporating momentum is given by:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla f(\mathbf{w}^k) + \beta(\mathbf{w}^k - \mathbf{w}^{k-1}), \tag{4}$$

- **Parameters:**
  - Here, α represents the learning rate, and β (where $0 < \beta < 1$) is the momentum factor.
- **Significance of Tuning:**
  - Correct tuning of β is essential for enhancing the convergence efficiency of the method.

Equivalent Formulation of Momentum Update

- **Introduction of an Auxiliary Variable:**
  - The momentum update is reformulated by introducing an auxiliary variable, $\mathbf{z}^{k+1}$, to capture the momentum aspect.
- **Momentum Update Equation:**
  - The update of the auxiliary variable $\mathbf{z}^{k+1}$ is given by:

$$\mathbf{z}^{k+1} = \beta \mathbf{z}^k + \nabla f(\mathbf{w}^k), \tag{5}$$

- **Parameter Update Equation:**
  - Subsequently, the parameter update is expressed as:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \mathbf{z}^{k+1}, \tag{6}$$

- **Interpretation of the Update Mechanism:**
  - This formulation, separating the position update ($\mathbf{w}^{k+1}$) and the momentum component ($\mathbf{z}^{k+1}$), aligns with principles of motion in physics and offers clarity in understanding the momentum update process in optimization.

Visualizing the Effectiveness of Momentum

Momentum plays a crucial role in optimizing gradient descent steps. It corrects their direction to be more aligned with the most efficient path to the objective's minimum:

- **Correcting Step Direction:** Gradient descent steps, originally perpendicular to the cost function's level curves, are adjusted by momentum.
- **Directing Towards the Minimum:** Momentum redirects these steps towards the objective's minimum, rather than following the indirect path of standard gradient descent.
- **Visual Comparison:** Figure 2 visually compares the paths of standard gradient descent (in red) and gradient descent with momentum (in blue). It highlights how momentum leads to a more direct and efficient route to the minimum.

Comparative Illustration of Gradient Descent with Momentum

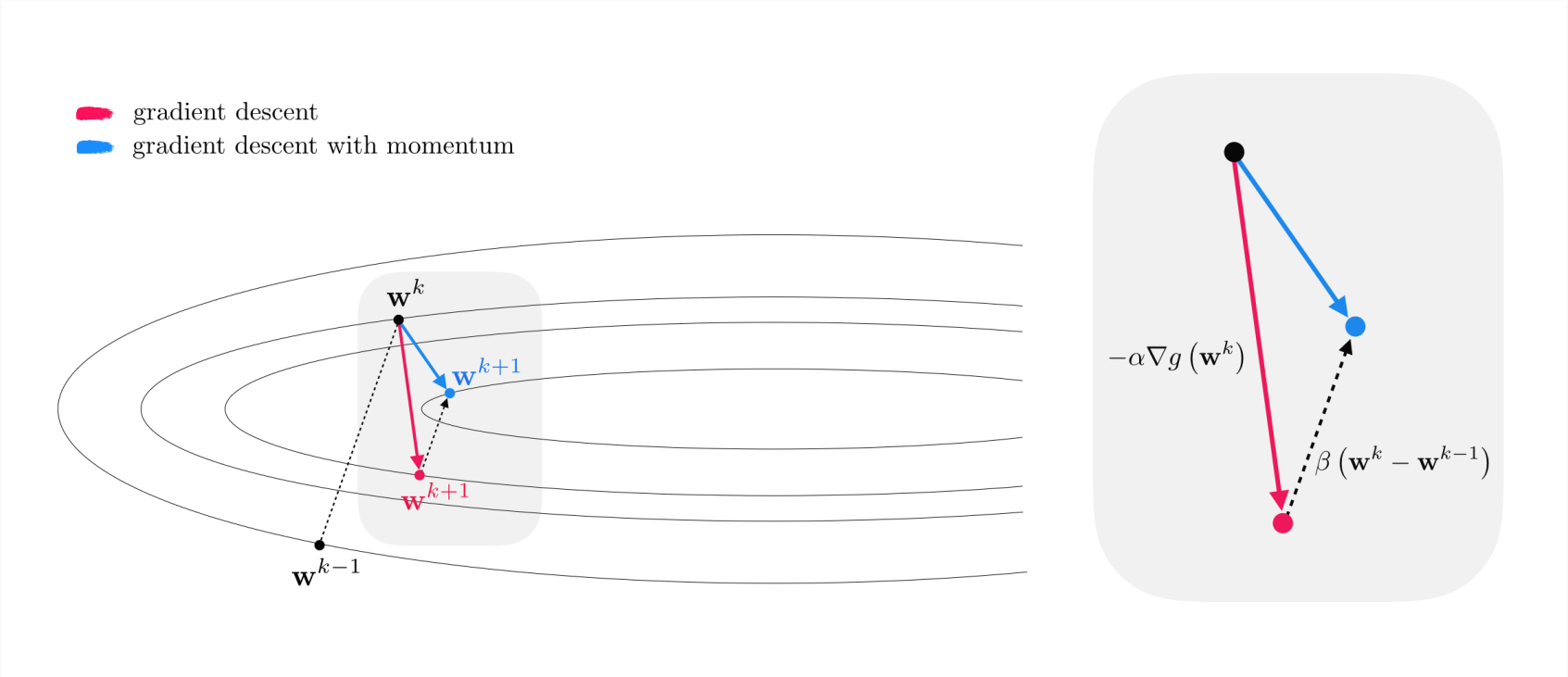


Figure 4: A comparative illustration of the paths taken by standard gradient descent (red) and gradient descent enhanced with momentum (blue), showing the latter's more direct and smoother convergence towards the minimum.

Gradient Descent with Momentum

Algorithm 1 Gradient Descent with Momentum

1: **Input:** Objective function f, initial point w_0, learning rate α, momentum coefficient β.
2: **Output:** Optimized parameters w.
3: **Initialize:** Set $w \leftarrow w_0$, $v \leftarrow 0$, iteration counter $k \leftarrow 0$.
4: **while** stopping criteria are not met **do**
5: $\quad k \leftarrow k + 1$.
6: $\quad$ Compute gradient: $g \leftarrow \nabla f(w)$.
7: $\quad$ Update velocity: $v \leftarrow \beta \cdot v + g$.
8: $\quad$ Update parameters: $w \leftarrow w - \alpha \cdot v$.
9: **end while**
10: **Return** w.

Ideal Example: Using Momentum in Simple Quadratic Functions

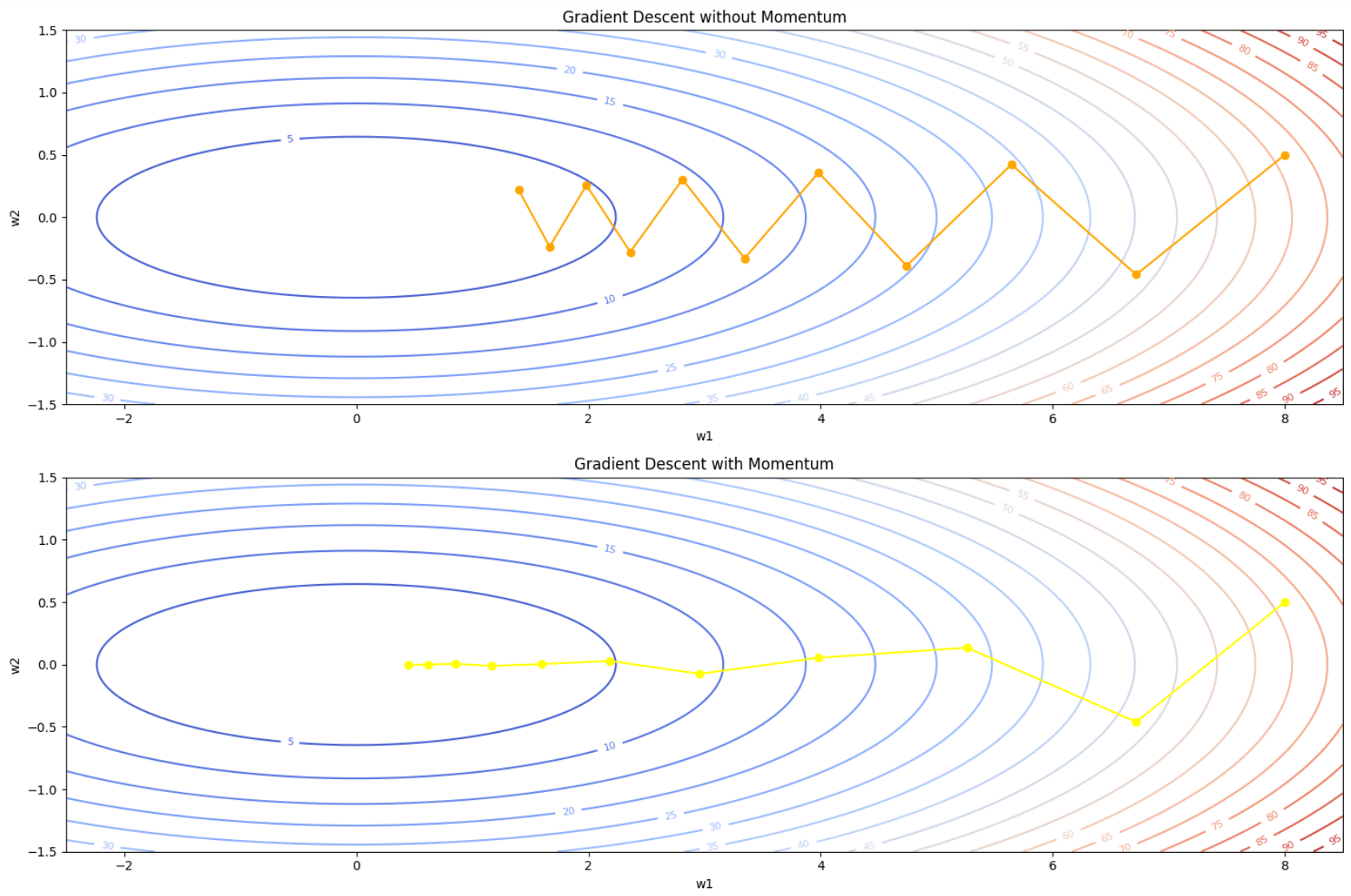
Consider a simple quadratic cost function in 2-dimensions:

$$g(\mathbf{w}) = a + \mathbf{b}^\top \mathbf{w} + \mathbf{w}^\top \mathbf{C}\mathbf{w}$$

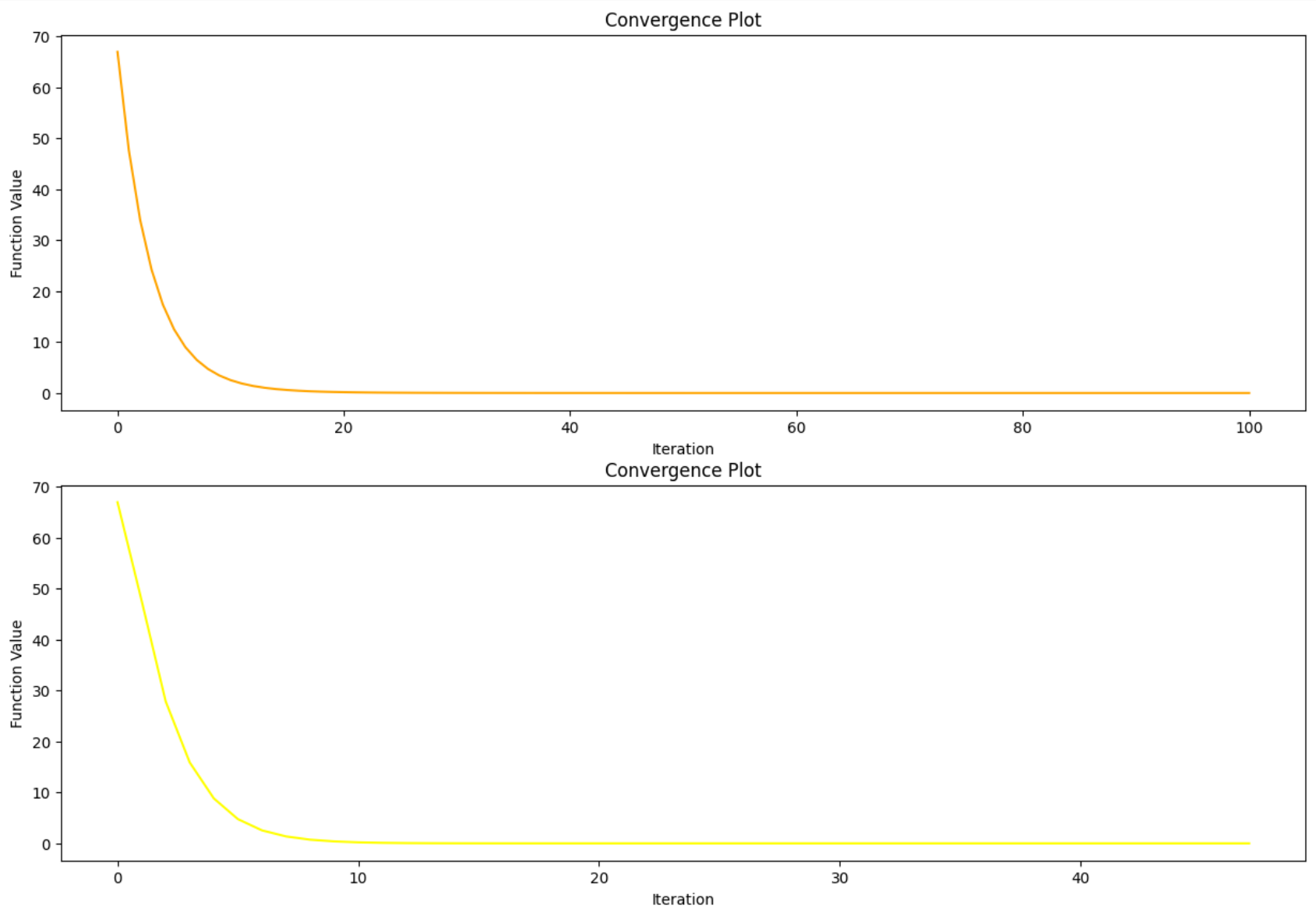
where a is a scalar, $\mathbf{b}$ a 2×1 vector, and $\mathbf{C}$ a 2×2 matrix.

Setting $a = 0$, $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, and $\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 12 \end{bmatrix}$, we run gradient descent with $\beta = 0.3$ and without $(\beta = 0)$ momentum for a maximum of 10 iterations with the step length parameter set to $\alpha = 0.08$ in both cases.

Ideal Example: Gradient Descent Contour Path



Ideal Example: Gradient Descent Convergence Plots



Extra: Nesterov Momentum in Gradient Descent

Refining Momentum with Nesterov's Approach

- **Enhancing Standard Momentum:**
  - Nesterov momentum refines the standard momentum method by integrating the momentum term within the gradient computation.
- **Nesterov Momentum Update Formula:**
  - The update formula is:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \alpha \nabla g(\mathbf{w}^k + \beta(\mathbf{w}^k - \mathbf{w}^{k-1})) + \beta(\mathbf{w}^k - \mathbf{w}^{k-1}) \quad (7)$$

- **Defining an Intermediate Step:**
  - With $\mathbf{w}^{\tilde{k}} = \mathbf{w}^k + \beta(\mathbf{w}^k - \mathbf{w}^{k-1})$, the approach allows for computing the gradient step at $\mathbf{w}^{\tilde{k}}$, leading to a more anticipatory update.
- **Modified Update Equation:**
  - This results in the modified update equation:

$$\mathbf{w}^{k+1} = \mathbf{w}^{\tilde{k}} - \alpha \nabla g(\mathbf{w}^{\tilde{k}}) \quad (8)$$

Visualizing the Effect of Nesterov Momentum

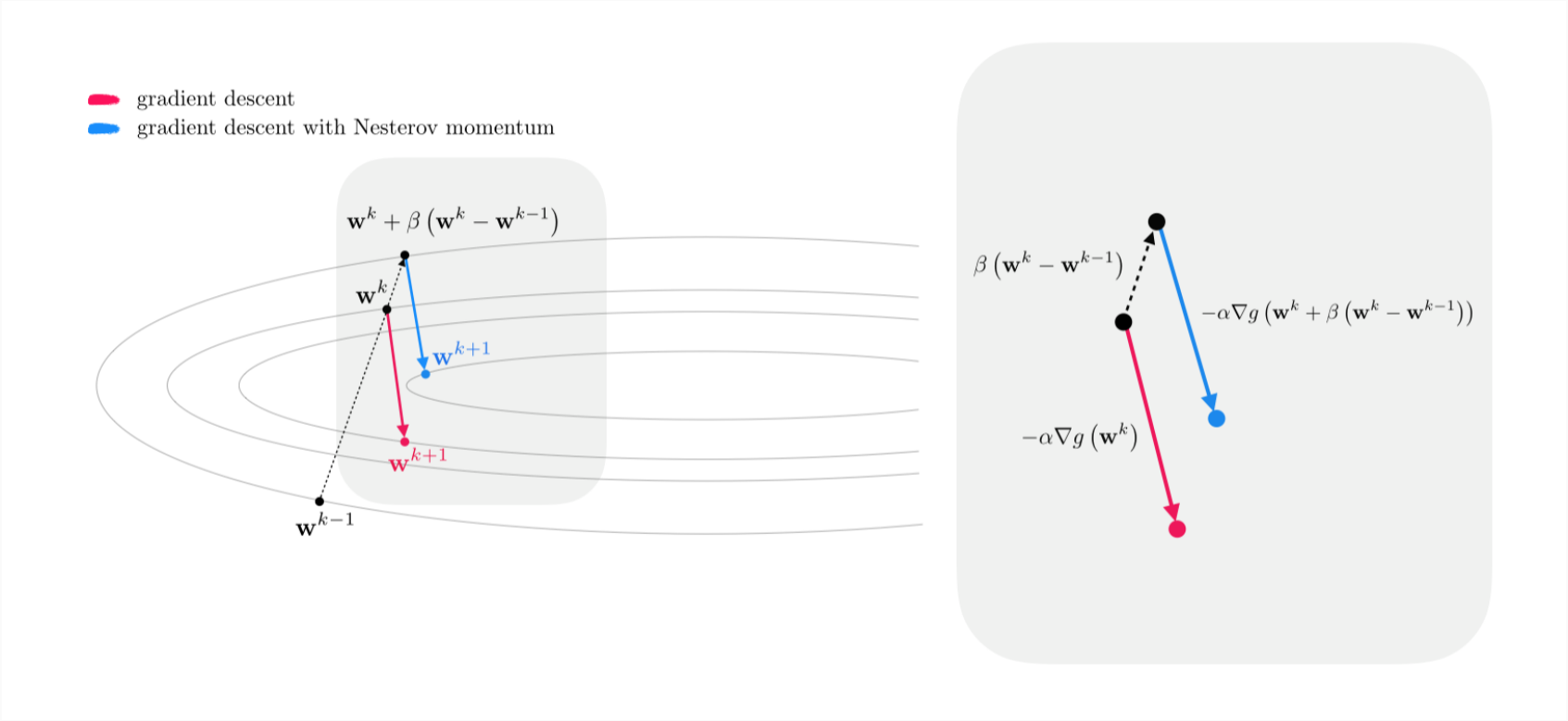


Figure 5: Comparative visualization of gradient descent with standard (red) and Nesterov (blue) momentum, showing Nesterov's improved anticipation in narrow valleys.

Effective Sample Weighting: Application of Momentum in SGD

The Significance of Minibatch SGD in Deep Learning

In deep learning, Minibatch Stochastic Gradient Descent (SGD) is crucial for handling large datasets and complex models, as it accelerates computation and reduces variance through gradient averaging.

$$\mathbf{g}_t = \partial_{\mathbf{w}} \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} f(\mathbf{x}_i, \mathbf{w}_{t-1}) = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \mathbf{g}_{i,t-1}. \tag{9}$$

Here, $\mathbf{g}_{ii}$ denotes the gradient with respect to weights $\mathbf{w}$ at data point $\mathbf{x}_i$. In deep learning, our objective extends to further enhancing variance reduction beyond minibatch SGD for more stable training.

Concept of Leaky Averages in Deep Learning

In deep learning, introducing "leaky average" for gradient computation helps in stabilizing training:

$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1} \tag{10}$$

Here, β is a parameter in the interval (0, 1). This method is crucial for combining current and previous gradients in deep neural networks, where momentum, $\mathbf{v}$, accumulates past gradients akin to a rolling ball gaining force.

Expanding Leaky Averages in Deep Learning

Applying expanded leaky averages in deep learning:

$$\mathbf{v}_t = \beta^2 \mathbf{v}_{t-2} + \beta \mathbf{g}_{t-1,t-2} + \mathbf{g}_{t,t-1} = \ldots = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}. \tag{11}$$

A higher β in deep learning implies long-term averaging, altering the gradient direction from steepest descent for a single instance to an average of past gradients, aiding in smoother convergence.

Leveraging Momentum in Deep Learning's Stochastic Gradient Descent

Incorporating momentum in deep learning's SGD, including minibatch SGD, involves careful velocity updates. With $\mathbf{v}_0$ initialized to zero, the gradient accumulation is described by:

$$\mathbf{v}_t = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau}. \tag{12}$$

For large τ, the series converges to $\frac{1}{1-\beta}$, effectively scaling the step size and promoting a more stable descent in deep neural networks. Subsequent illustrations will depict the change in effective sample weight for varying β values.

Impact of Momentum Parameter Adjustment

Adjusting the momentum parameter β has a significant impact on deep learning models. A reduced β, like 0.25, can lead to slower yet more stable convergence compared to no momentum.

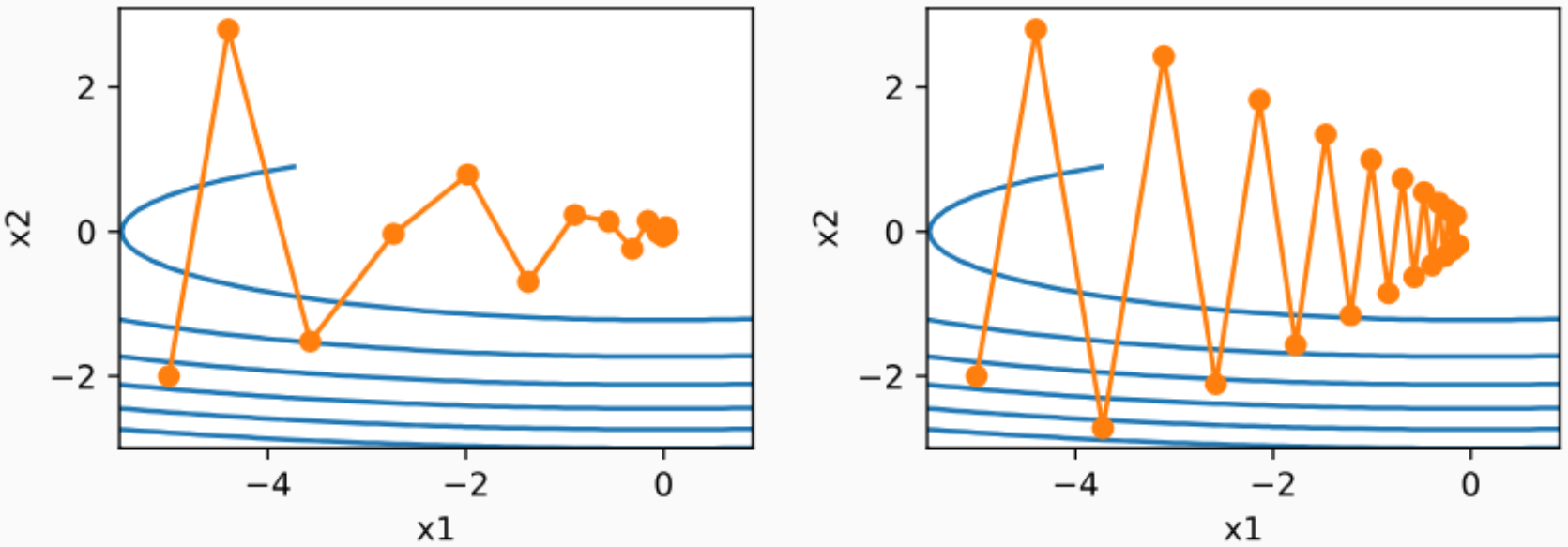


Figure 6: Comparing gradient descent trajectories on a distorted ellipsoid objective with varying momentum parameters, illustrating effects on convergence speed and stability.

Implementations: Illustrating Pitfalls of Momentum

The Rosenbrock Function

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{13}$$

$$\nabla f = \begin{bmatrix} -2(1 - x) - 400x(y - x^2) \\ 200(y - x^2) \end{bmatrix} \tag{14}$$

Specific Challenges for Momentum Methods:

- **Narrow Valley Structure:** The Rosenbrock function features a narrow, elongated valley leading to the global minimum. This structure causes momentum methods to experience oscillations as they tend to overshoot the valley due to accumulated momentum.
- **Risk of Overshooting:** The function's flat regions combined with steep slopes pose a risk of overshooting, especially when momentum accumulates in directions not aligned with the valley's orientation.

Visualization of Optimization Path

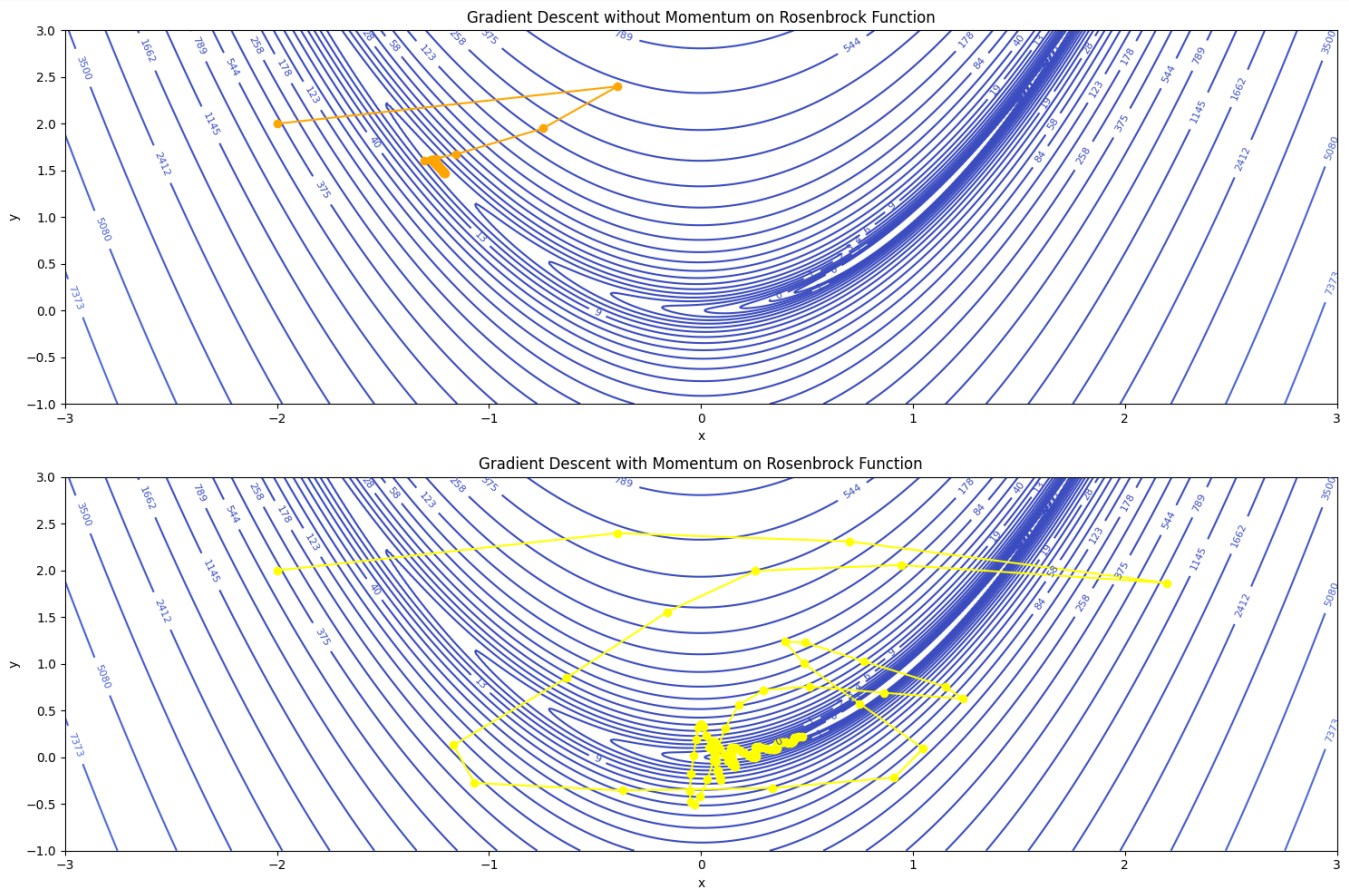


Figure 7: Contour path visualization in Rosenbrock function optimization

Analysis of Convergence Behavior

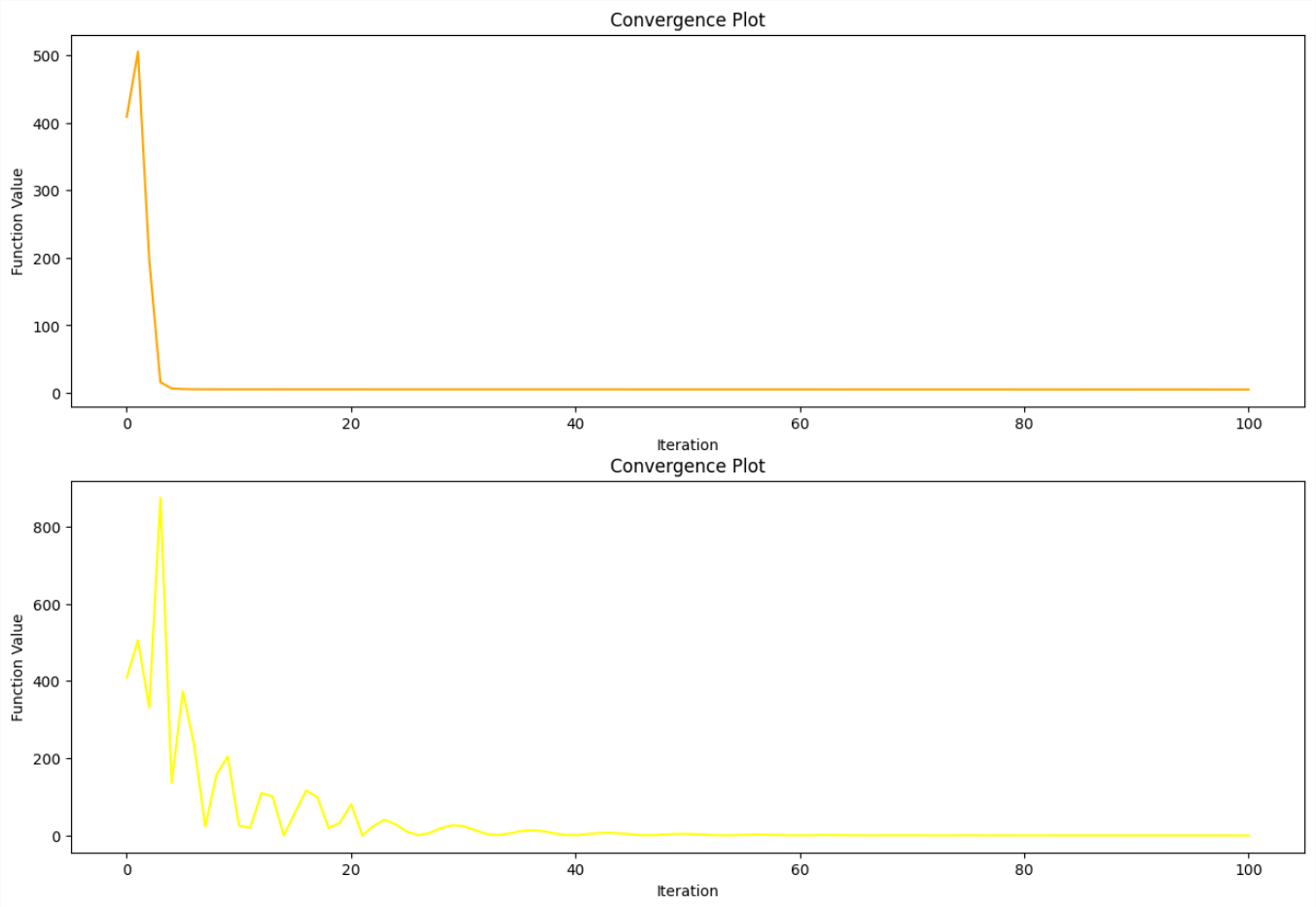


Figure 8: Convergence plot for momentum-based optimization

Challenges of Using Momentum in Non-Convex Optimization

- **Oscillations Due to Momentum Memory:** In regions with high curvature or narrow valleys, momentum retains a memory of previous gradients. This memory can cause oscillatory behavior, making it difficult to settle in minima, especially those located in tight or sharp regions of the landscape.
- **Risk of Overshooting Minima:** Accumulated momentum can cause the optimizer to overshoot the minima. This is particularly problematic in steep areas of the landscape where the change in gradient direction is abrupt and significant.
- **Path Suboptimality and Slow Convergence:** Momentum can potentially lead the optimizer down suboptimal paths, diverging from the most direct route to a minimum. This often results in increased iterations required for convergence, especially in landscapes with complex topologies.

Convergence Analysis for Momentum Methods

Theorem on Convergence in λ-Strongly Convex Scenarios

Theorem: Let $h(\mathbf{x})$ be a λ-strongly convex function. Then, under appropriate choices of parameters β and η, the sequence $\{\mathbf{x}_t\}$ generated by the momentum method converges to the minimizer $\mathbf{x}^*$.

Proof: To prove this, we analyze the dynamics of the momentum method under the strong convexity condition, which ensures a unique minimizer and a lower bound on the curvature of $h(\mathbf{x})$.

Analyzing Momentum Method Dynamics

- **Strong Convexity Definition:** A function $h(\mathbf{x})$ is λ-strongly convex if $\forall \mathbf{x}, \mathbf{y}$,

$$h(\mathbf{y}) \geq h(\mathbf{x}) + \nabla h(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\lambda}{2} \|\mathbf{y} - \mathbf{x}\|^2. \tag{15}$$

- **Momentum Update Dynamics:**

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \mathbf{v}_t, \quad \mathbf{v}_t = \beta \mathbf{v}_{t-1} + \nabla h(\mathbf{x}_t). \tag{16}$$

- **Convergence Analysis:** The convergence analysis involves showing that $\{\mathbf{x}_t\}$ approaches $\mathbf{x}^*$ as $t \to \infty$.

Impact of Parameters on Convergence

- **Convergence Criteria:** The rate of convergence depends on the balance between β and η, and the strong convexity parameter λ.
- **Mathematical Criteria for Convergence:** For convergence, it is required that $0 < \eta < \frac{2}{\lambda(1+\beta)}$.
- **Conclusion:** Given these conditions, the sequence $\{\mathbf{x}_t\}$ generated by the momentum method converges to $\mathbf{x}^*$.

Further Remarks: Adam Optimization Algorithm

Adam Optimization Algorithm

Algorithm 2 Adam Algorithm

1: **Initialize:** Set initial parameters θ_0, learning rate α, momentum coefficients β_1, β_2, initialize $m_0 = 0$, $v_0 = 0$, and timestep $t = 0$.
2: **while** stopping criteria are not met **do**
3: $t \leftarrow t + 1$
4: Compute gradient $g_t = \nabla_\theta L(\theta_{t-1})$ at the current parameters.
5: Update biased first moment estimate: $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
6: Update biased second raw moment estimate: $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
7: Correct bias in first moment: $\hat{m}_t = m_t / (1 - \beta_1^t)$
8: Correct bias in second moment: $\hat{v}_t = v_t / (1 - \beta_2^t)$
9: Update parameters: $\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
10: **end while**

About Adam Optimization

Adam (Adaptive Moment Estimation) combines ideas from momentum and RMSprop. Key features include:

- **Moment Estimation:** Maintains two moving averages for gradients - a first moment (mean) and a second moment (uncentered variance).
- **Bias Correction:** Corrects bias in moving averages due to their initialization at the origin.
- **Adaptive Learning Rate:** Adjusts the learning rate for each parameter based on the first and second moments.
- **Efficiency and Efficacy:** Performs well in practice, especially with large datasets or parameters.

References

References

- Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics, 4*(5), 1-17.
- Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Doklady AN USSR, 269*, 543-547.
- Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *International conference on machine learning*, 1139-1147.
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.