Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 25, 2024

1. Optimization
2. The Loss Surface
3. Newton's Method
4. Gradient Descent
5. Momentum
6. References

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum

6 References

Optimization Problem

- **Goal**: Find the value of x where $f(x)$ is at a **minimum** or **maximum**.
- In neural networks, we aim to minimize **prediction error** by finding the optimal weights w^*:

$$w^* = \arg\min_w J(w)$$

- Simply put: determine the **direction to step** that will quickly **reduce loss**.

Figure 1: Which way is downhill? (CS231n, Stanford)

Convexity and Optimization

- **Convex Functions**:
  - A function is **convex** if any line segment between points on the curve lies **above or on** the curve.
  - Convex functions are easier to optimize, as they have a single **global minimum**.
  - **Gradient Descent** is guaranteed to reach the global minimum in convex functions.

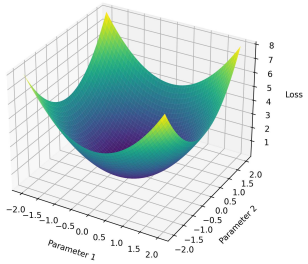

Figure 2: Example of convex function (bowl shape)

Non-Convex Functions and Challenges

- **Non-Convex Functions**:
  - Characterized by multiple **local minima** and **saddle points**.
  - **Global Minimum**: Overall lowest point.
  - **Local Minimum**: Lower than nearby points, but not the lowest overall.
  - **Saddle Points**: Regions where the gradient is close to zero but can increase or decrease in other directions.
- Finding the **global minimum** is more complex in non-convex functions.

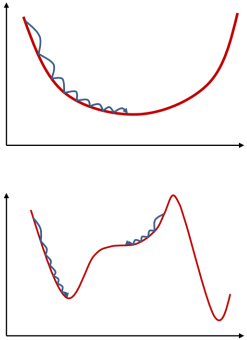

Figure 3: Convex (top) vs. Non-Convex (bottom) functions (CMU, 11-785)

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum

6 References

Loss Surface Definition

- The **loss surface** shows how error changes based on network weights.
- For neural networks, the loss surface is typically **non-convex** due to multiple layers, nonlinear activations, and complex parameter interactions, resulting in **multiple local minima** and **saddle points**.
- In large networks, most local minima yield similar error values close to the **global minimum**; this is less true in smaller networks.

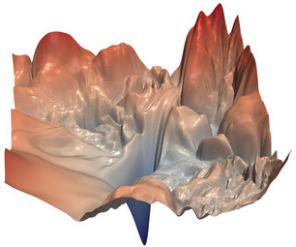

Figure 4: Loss surface of ResNet56. Source: `https://github.com/tomgoldstein/loss-landscape`

Loss Optimization

- **Goal**: How can we optimize a non-convex loss function effectively?
- **Newton's Method**:
  - When optimizing, we look for critical points where the derivative $f'(x) = 0$, which may indicate minima, maxima, or saddle points.
  - Newton's Method uses the second derivative (Hessian) to adjust step sizes, which can lead to faster convergence compared to Gradient Descent.
- **Gradient Descent**:
  - This method identifies the steepest descent direction to guide the optimization process.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum

6 References

Overview of Newton's Optimization Method

- **Origin**:
  - Developed from classical physics and calculus, Newton's method aims to optimize functions by leveraging **curvature** information.
  - Named after Sir Isaac Newton, who introduced the idea of using **second derivatives** to predict the behavior of functions.
- **Objective**:
  - Directly find optimal weights by considering both gradients (first derivative) and curvature (second derivative) of the loss surface.

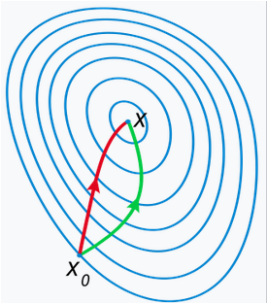

Figure 5: A comparison of gradient descent (green) and Newton's method (red) for minimizing a function. Source: Wikipedia

Newton's Optimization Method: Mathematical Formulation

- **Core Mechanism**:
  - Newton's method adjusts weight updates using the Hessian matrix, incorporating second-order derivative information to account for local curvature.
- **Update Rule**:

$$w_{t+1} = w_t - H^{-1} \nabla_w J(w_t)$$

where:

- H is the Hessian matrix of second-order partial derivatives, capturing curvature information.
- $\nabla_w J(w_t)$ is the gradient of the cost function J at time t.

Newton's Method: Advantages and Disadvantages

- Newton's method offers various benefits but also has limitations, especially in large-scale machine learning. Below is a summary:

Advantages	Disadvantages
Faster Convergence Quadratic convergence enables reaching minima faster in convex problems.	**Computationally Expensive** Requires Hessian calculation, making it costly in high-dimensional models.
Adaptive Step Sizes Curvature-based step adjustment avoids slow progress in shallow regions.	**Memory Intensive** Storing the Hessian matrix is memory-intensive for models with millions of parameters.
Reduced Oscillations Curvature information stabilizes paths in oscillatory regions.	**Convergence Challenges** May converge to saddle points in non-convex functions common in machine learning.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum

6 References

Gradient Descent Overview

- **Gradient Descent**: As mentioned earlier in this course, Gradient Descent is an iterative method to minimize error by updating weights in the direction of the **negative gradient**:

$$w_{t+1} = w_t - \eta \nabla J(w_t)$$

  where η is the **learning rate**.

- **Types of Gradient Descent**:
  - **Batch**: Full dataset for stable but slow updates.
  - **Stochastic (SGD)**: One data point for fast, noisy updates.
  - **Mini-Batch**: Small batches, balancing speed and stability.

Problems with Gradient Descent

- **High Variability (SGD)**: Quick in steep directions but slow in shallow ones, causing **jitter and slow progress**.
- **Local Minima and Saddle Points**: Risk of **sub-optimal solutions** or long convergence times in flat regions.
- **Noisy Updates**: Using individual points or mini-batches introduces noise, affecting stable convergence.

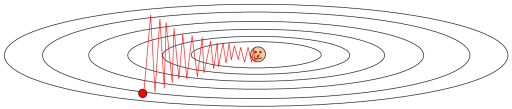

Figure 6: SGD Variability (CS231n, Stanford)

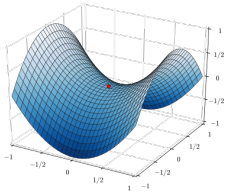

Figure 7: Saddle Point (`en.wikipedia.org`)

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum
- First Moment (Momentum)
- Second Moment (Variance)
- Adam: Adaptive Moment Estimation

6 References

Problem Definition and Solution

- **Objective**: Enhance the vanilla Gradient Descent algorithm to improve convergence and stability.
- **Challenges**:
  - Selecting an appropriate learning rate is crucial to avoid slow convergence and getting stuck in local minima.
- **Proposed Solution**:
  - Instead of testing multiple learning rates, which can be inefficient and time-consuming, incorporate **Momentum** to adaptively adjust the learning rate based on oscillations:
    - Increase steps in stable directions.
    - Decrease steps in oscillating directions.

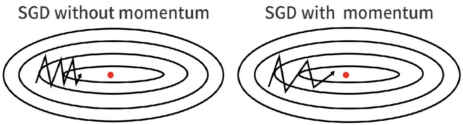

Figure 8: Momentum smooths oscillations and accelerates progress. Source: paperswithcode

Introduction to Momentum in Optimization

- **Origin of Momentum**:
  - Inspired by Newtonian physics, momentum in optimization uses the concept of velocity in motion, accumulating gradient history to smooth the learning trajectory, akin to an object moving based on past inertia.
  - Initially introduced to tackle challenges in gradient descent, where inconsistent gradients or noisy updates lead to erratic and slow convergence.
- **Purpose of Momentum**:
  - **Dampens Oscillations**: Utilizes prior gradients to minimize oscillations along steep or erratic regions, resulting in a smoother and more stable path.
  - **Speeds Up Convergence**: Particularly effective in narrow valleys or flat regions, where standard gradient descent may struggle or oscillate, causing slow progress.
- **Mechanism**:
  - Achieves improvements by adding a "velocity" term to gradient descent, facilitating consistent directional movement and balancing update sizes for faster and more reliable convergence.

How Momentum Accelerates Learning

- In momentum-based optimization, updates are influenced by a weighted combination of the current gradient and previous updates.
- **Why Use Momentum?**
  - **Reduces Oscillations**: Momentum softens oscillations along steeper directions while preserving progress in directions with smaller gradients, resulting in a balanced path.
  - **Increases Efficiency**: By "accelerating" updates in directions of consistent gradient descent, momentum minimizes unnecessary steps and decreases overall convergence time.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum
- First Moment (Momentum)
- Second Moment (Variance)
- Adam: Adaptive Moment Estimation

6 References

First Moment (Momentum)

- **Definition**: The first moment, m_t, represents a moving average of past gradients. It builds "velocity" that propels learning in a consistent direction.
- **Update Rule**:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla_w J(w_t)$$

$$w_{t+1} = w_t - \eta m_{t+1}$$

  where:
  - β_1: Decay rate, usually 0.9 or 0.99, which controls the weight of past gradients.
  - η: Learning rate.
- **Why Use First Momentum?**
  - Inspired by the idea of rolling momentum, it smooths and accelerates learning by sustaining direction from prior gradients.
  - This type of momentum is ideal for traversing narrow valleys or regions where standard gradient descent would oscillate.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum
- First Moment (Momentum)
- Second Moment (Variance)
- Adam: Adaptive Moment Estimation

6 References

Second Moment (Variance)

- **Definition**: The second moment, v_t, represents the moving average of squared gradients. It measures the gradient magnitude over time.
- **Update Rule**:

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)(\nabla_w J(w_t))^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_{t+1} + \epsilon}} \nabla_w J(w_t)$$

where:

  - β_2: Decay rate for variance (usually 0.99 or 0.999).
  - ϵ: Small constant to prevent division by zero.

- **Why Use Second Momentum?**
  - Adjusts step size based on gradient magnitude, preventing large steps when gradients are large and accelerating learning when they are small.
  - Useful in adaptive methods like Adam, where the combination of first and second moments allows stable and efficient convergence across various scenarios.

Moment Bias Correction

- **Problem**: When we start training, both m_t and v_t are initialized to zero, causing their estimates to be biased toward zero in the early steps, especially when gradients are small.
- **Solution**: We use bias-corrected versions of m_t and v_t to address this:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

- These corrections compensate for the bias by scaling m_t and v_t upward, especially in the early steps when t is small, ensuring more accurate estimates of the moments.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum
- First Moment (Momentum)
- Second Moment (Variance)
- Adam: Adaptive Moment Estimation

6 References

Introduction to Adam Optimizer

- **Origin and Purpose**:
  - Proposed in 2014 by Diederik Kingma and Jimmy Ba, Adam (Adaptive Moment Estimation) addresses key limitations in earlier optimization methods by combining aspects of **momentum** and **adaptive learning rates**.
  - Adam is designed to handle sparse gradients and noisy updates by adjusting the learning rate for each parameter based on historical gradients.
- **Core Idea**:
  - Adam optimizes by maintaining two moving averages — the **first moment (mean of gradients)** and the **second moment (variance of gradients)** — allowing it to adapt learning rates for each parameter individually.

Adam's Adaptive Learning Rate Mechanism

- **Why Adaptive Rates?**
  - Unlike traditional SGD, Adam adapts the learning rate for each parameter based on recent gradient magnitudes.
  - Large gradients lead to reduced update sizes, while smaller gradients allow larger updates, balancing convergence speed and stability.
- **Moment Tracking**
  - The **first moment** (m_t) tracks the mean of gradients to provide momentum.
  - The **second moment** (v_t) tracks squared gradients, enabling Adam to normalize updates and prevent sudden changes in direction.

Mathematical Formulation of Adam

- **Adam Update Rules**:
  - First moment estimate:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla_w J(w_t)$$

  - Second moment estimate:

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) (\nabla_w J(w_t))^2$$

  - Bias-corrected moments to address initialization bias:

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 - \beta_1^{t+1}}, \quad \hat{v}_{t+1} = \frac{v_{t+1}}{1 - \beta_2^{t+1}}$$

  - Update step for parameter w_t:

$$w_{t+1} = w_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon}$$

where β_1, β_2 are decay rates, and ϵ is a small constant to prevent division by zero.

1 Optimization

2 The Loss Surface

3 Newton's Method

4 Gradient Descent

5 Momentum

6 References

[1] F.-F. Li, J. Wu, and R. Gao, "Cs231n: Convolutional neural networks for visual recognition." *Lecture slides*, Apr. 2022.
Available: http://cs231n.stanford.edu/slides/2022.

[2] M. learning for signal processing group, "11-785 introduction to deep learning." *Lecture slides*, 2024.
Available: https://deeplearning.cs.cmu.edu/F24/document/slides.

[3] A. Amini, "6s191: Introduction to deep learning." *Lecture slides*, 2024.
Available: http://introtodeeplearning.com/.

[4] "Gradient descent explained."
`https://ml-explained.com/blog/gradient-descent-explained`, 2021.

[5] L. Jiang, "A visual explanation of gradient descent methods: Momentum, adagrad, rmsprop, adam." `https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmspr` 2021.

[6] S. Kuznetsov, "Gradient descent." https://blog.skz.dev/gradient-descent, 2021.

[7] T. Goldstein, "Loss landscape." https://github.com/tomgoldstein/loss-landscape, 2021.

[8] G. Sanderson, "Gradient descent, animated." https://www.youtube.com/watch?v=IHZwWFHWa-w, 2017.

[9] "Understanding optimization algorithms." https://laptrinhx.com/understanding-optimization-algorithms-3818430905/, 2021.

[10] "Sgd with momentum." https://paperswithcode.com/method/sgd-with-momentum, 2021.

[11] "Saddle point." https://en.wikipedia.org/wiki/Saddle_point, 2024.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] "Newton's method in optimization." https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization, 2024.

[14] GeeksforGeeks, "Optimization in neural networks and newton's method." https://www.geeksforgeeks.org/optimization-in-neural-networks-and-newtons-method/, 2024.

[15] GeeksforGeeks, "Optimization algorithms in machine learning." https://www.geeksforgeeks.org/optimization-algorithms-in-machine-learning/, 2024.

[16] D2L.ai, "Adam." https://d2l.ai/chapter_optimization/adam.html, 2024.

[17] D2L.ai, "Momentum." https://d2l.ai/chapter_optimization/momentum.html, 2024.