Non-Convex Optimization and Algorithms for Machine Learning

Roshan Avik Srivastava
roshanavik@gmail.com

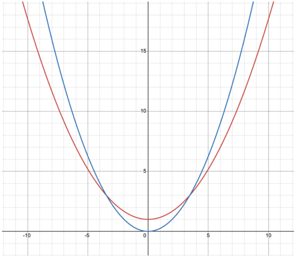
Euler Circle

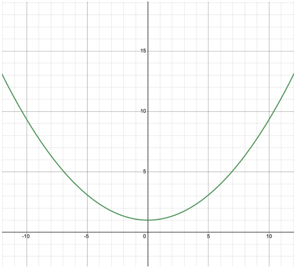
July 11, 2025

Table of Contents

The Loss Function

$$\min_{x \in \mathbb{R}^d} f(x)$$





Key:
Red Function: Original Function
Blue Function: Estimated Function
Green Function: Loss Function

Example Loss Function

True Function (Red): $f(x) = x^2$
Model Function (Blue): $\hat{f}(x) = wx + b$
Loss Function

$$L(w, b) = \sum_{i=1}^{n} \left(wx_i + b - x_i^2\right)^2$$

This can be rewritten in matrix form as:

$$L(w, b) = \left\| \mathbf{X} \begin{bmatrix} w \\ b \end{bmatrix} - \mathbf{y} \right\|_2^2$$

Where:

$$\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_n^2 \end{bmatrix}$$

Convex vs Non Convex

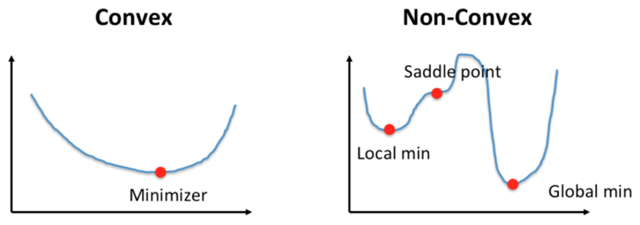
Definition. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *convex* if for every pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and every $\lambda \in [0, 1]$, we have

$$f(\lambda \mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}).$$

Strictly convex, if $\forall x, y, x \neq y, \forall \lambda \in (0, 1)$

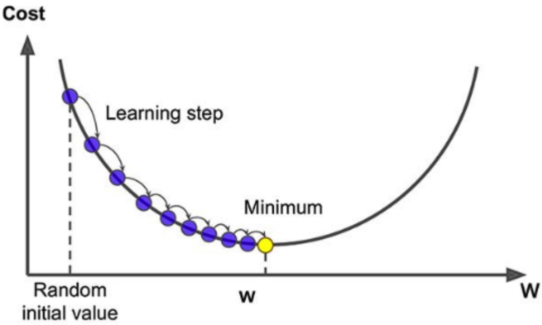
$$f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda) f(y).$$

For Convex functions, every local minimum is also a global minimum



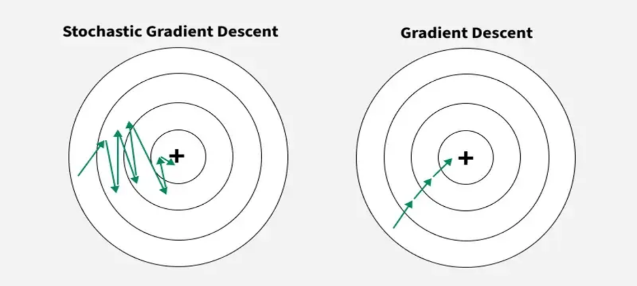
Gradient Descent

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta_n \nabla f(\mathbf{x}_n), \quad n \geq 0.$$



Stochastic Gradient Descent - Solutions for Non-Convex

$$x_{k+1} = x_k - \eta \nabla f_i(x_k)$$



GD vs SGD Tradeoff (Pros and Cons)

Pros

1. Much faster updates
2. Works well for large datasets
3. Includes an element of randomness, which can help escape bad minimas

Cons

1. Each step is “noisy”
2. May not always go downhill, but it often does

Importance of learning rate

VERY important in SGD

- If N is too big $\rightarrow$ Model jumps around randomly
- If N is too small $\rightarrow$ Learning is very slow

Common practice is to start with a larger learning rate and decrease over time.

Learning Rate Schedule helps with stabilization.

Learning Rate and Scheduling

The learning rate controls how big our steps are:

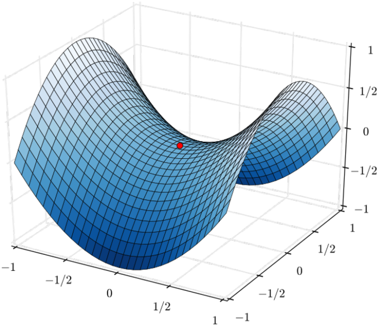
1. If it's too big: we overshoot the minimum.
2. If it's too small: we move too slowly and time complexity becomes too high.

Common Strategy: Learning Rate Scheduling

1. Start with a larger step size.
2. As you progress and get closer to the "minimum," reduce the step size to avoid overshooting.

Saddle Points and Escaping Them

$$x_{k+1} = x_k - \eta \nabla f(x_k) + \xi_k$$



With Random Noise:

. Escape any Saddle point quickly

. In Polynomial Time

Mini Batch SGD

$$x_{k+1} = x_k - \eta \cdot \frac{1}{B} \sum_{j=1}^{B} \nabla f_{i_j}(x_k)$$

Where B is the batch size. The algorithm uses a small subset (batch) of size B at each step.

L-Smoothness

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2$$

The Polyak–Łojasiewicz (PL) Condition

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f^*)$$

Theorem: PL Condition Guarantees Fast Convergence

Proof Sketch:

$$f(x_{k+1}) \leq f(x_k) - \eta \left(1 - \frac{L\eta}{2}\right) \|\nabla f(x_k)\|^2$$

$$\|\nabla f(x_k)\|^2 \geq 2\mu(f(x_k) - f^*)$$

$$f(x_{k+1}) \leq f(x_k) - \eta \left(1 - \frac{L\eta}{2}\right) \cdot 2\mu(f(x_k) - f^*)$$

$$f(x_{k+1}) - f^* \leq \left[1 - 2\eta\mu \left(1 - \frac{L\eta}{2}\right)\right] (f(x_k) - f^*)$$

$$(f(x_k) - f^*) \leq (\text{something small})^k \cdot (f(x_0) - f^*)$$

Many other methods - Adam W Optimizer

$$x_t = x_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda x_{t-1} \right)$$

Adaptive Movement Estimation
Momentum: Adam remembers past gradients
Adaptive Learning Rates: Each parameter gets its own step size
Employs Weight Decay

Application in Vision Transformers

Vision Transformers (ViTs) use the transformer architecture to process image data.

They are:

1. Large, with millions of parameters
2. Data hungry, trained on huge datasets
3. Non-convex

AdamW in ViTs:

1. Faster convergence
2. Improves generalization

What is up ahead?

Improved versions of Adam (e.g., AdaBelief, Lion)

Theoretical Advances:

. Why SGD works so well on non-convex problems

. Other Research on -

1. Efficient pre-training
2. Stability
3. Convergence guarantees
4. Scaling laws


THANK YOU!