Introduction to Machine Learning 4

Optimization ii

Tong Zhang UCAS 2025 fall

Recap

- We have learned GD, SGD
- The importance of choosing the learning rate

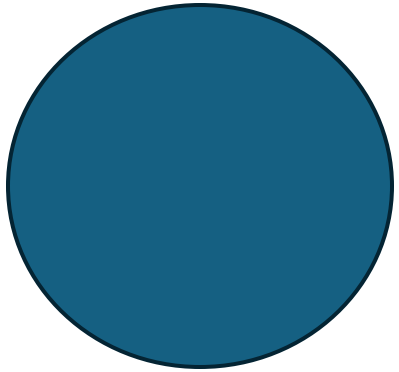
Plan

- Convexity: fundamental of optimization
- Why do we use Mini-batch SGD:
- Introduce more GD variants: momentum
- More optimization methods

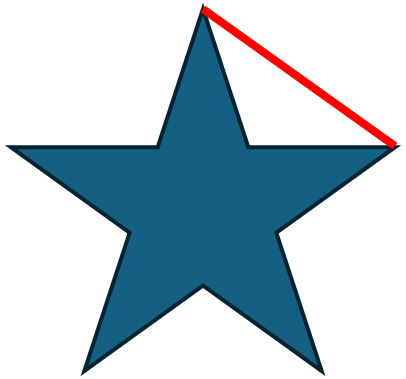
Convex set

- Concept: For any two points you pick within the set, the straight line segment connecting them lies entirely inside the set. Think of it this way: if you and a friend are inside a convex room, you can always see each other, with no walls blocking your line of sight.

Exercise

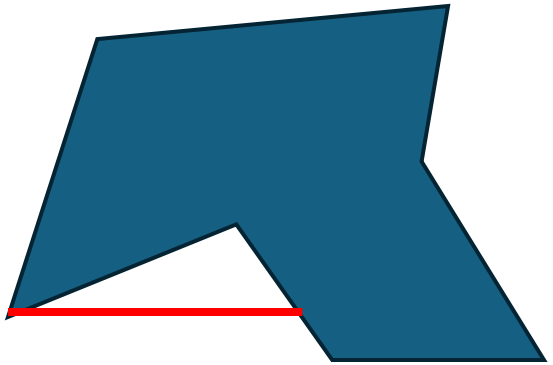


Yes

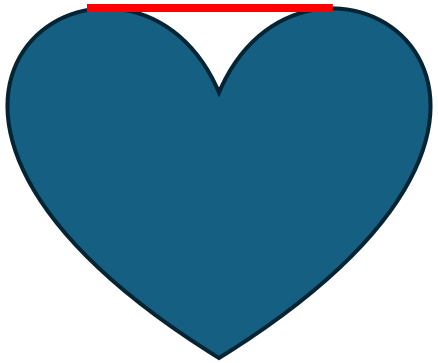


No

Yes



No

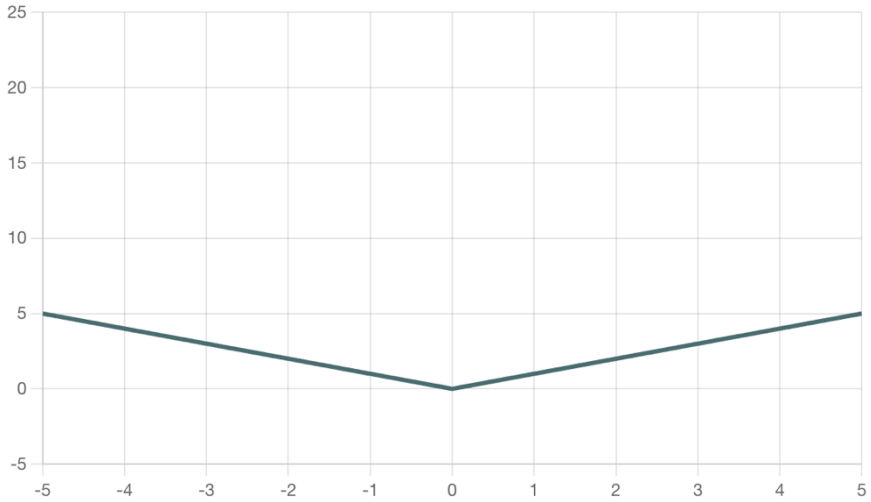


No

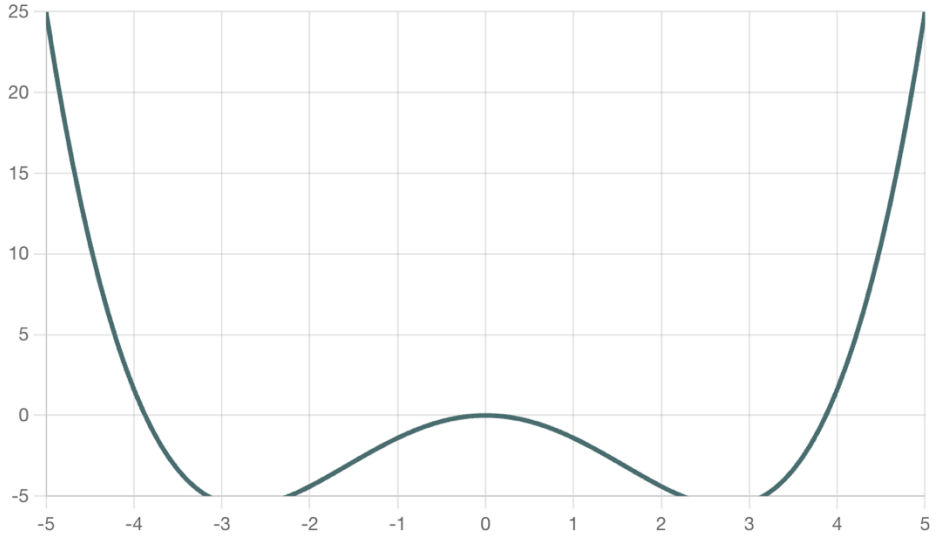
Convex function

- Concept: If you pick any two points on the function's curve, the line segment connecting them will never go below the function's graph.

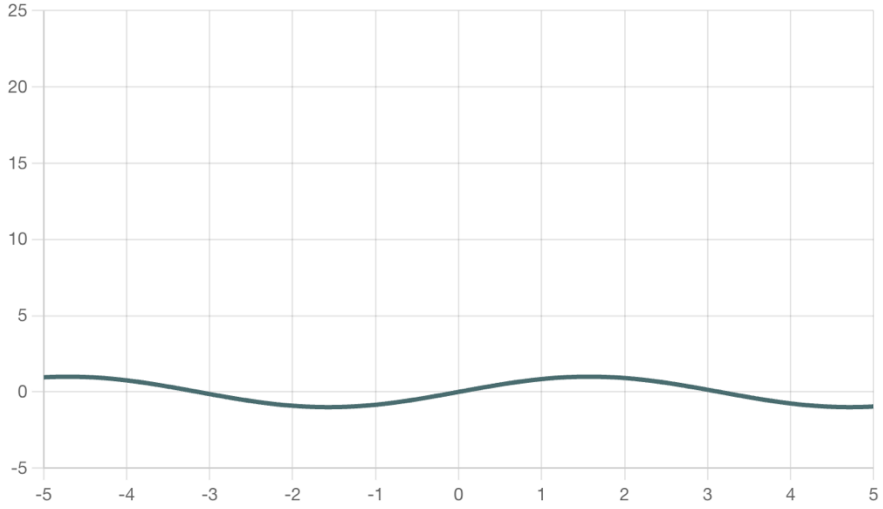
Exercise



Yes



No



No

Convexity

- Mathematically, convexity is the function property that guarantees local minima= global minima

- Global minima

$x_0 \in X$ is a global maximum point of function $f : X \to \mathbb{R}$, if $(\forall x \in X)\, f(x_0) \geq f(x)$.

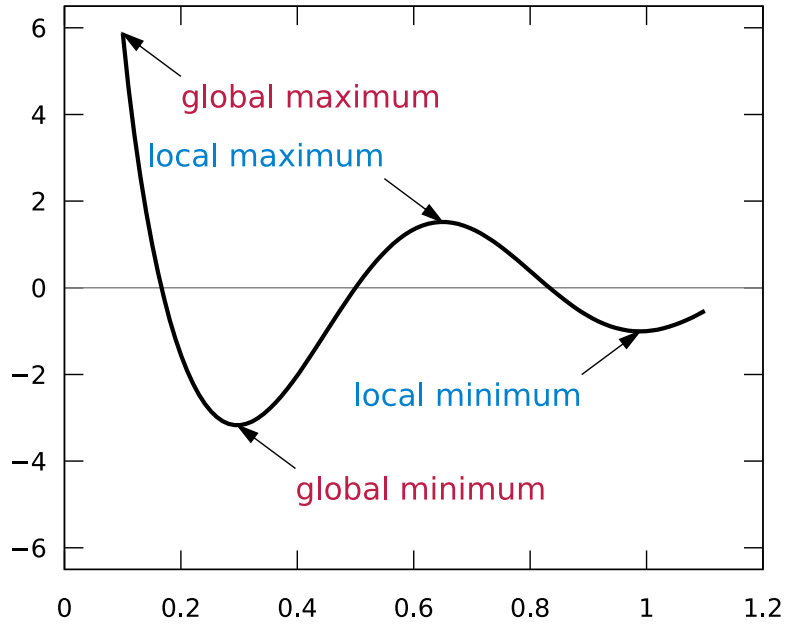
- Local minima

Let (X, d_X) be a metric space and function $f : X \to \mathbb{R}$. Then $x_0 \in X$ is a local maximum point of function f if $(\exists \varepsilon > 0)$ such that $(\forall x \in X)\, d_X(x, x_0) < \varepsilon \implies f(x_0) \geq f(x)$.

https://en.wikipedia.org/wiki/Maximum_and_minimum

Convexity

- Mathematically, convexity is the function property that guarantees local minima= global minimum
- Global minima
- Local minima



https://en.wikipedia.org/wiki/Maximum_and_minimum

Convexity

- 0-order condition (if and only if): for all $\lambda \in [0,1]$,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

- 1-order condition (if and only if):

$$f(x_2) \geq f(x_1) + \nabla f(x_1)(x_2 - x_1)$$

- 2-order condition (if and only if):

$$\nabla^2 f(x) \geq 0$$

Exercise

$f(x, y) = 2x^2 + 3y^2$

convex

$f(x, y) = e^x$

convex

$f(x, y) = -(x^2 + y^2)$

Concave

$f(x, y) = x^3$

Neither

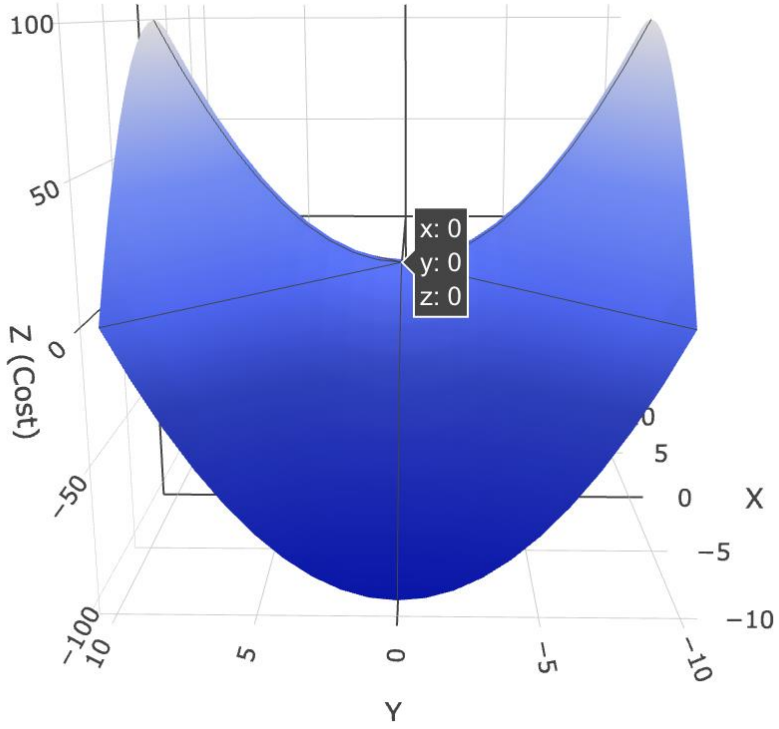
$f(x) = log(x)$

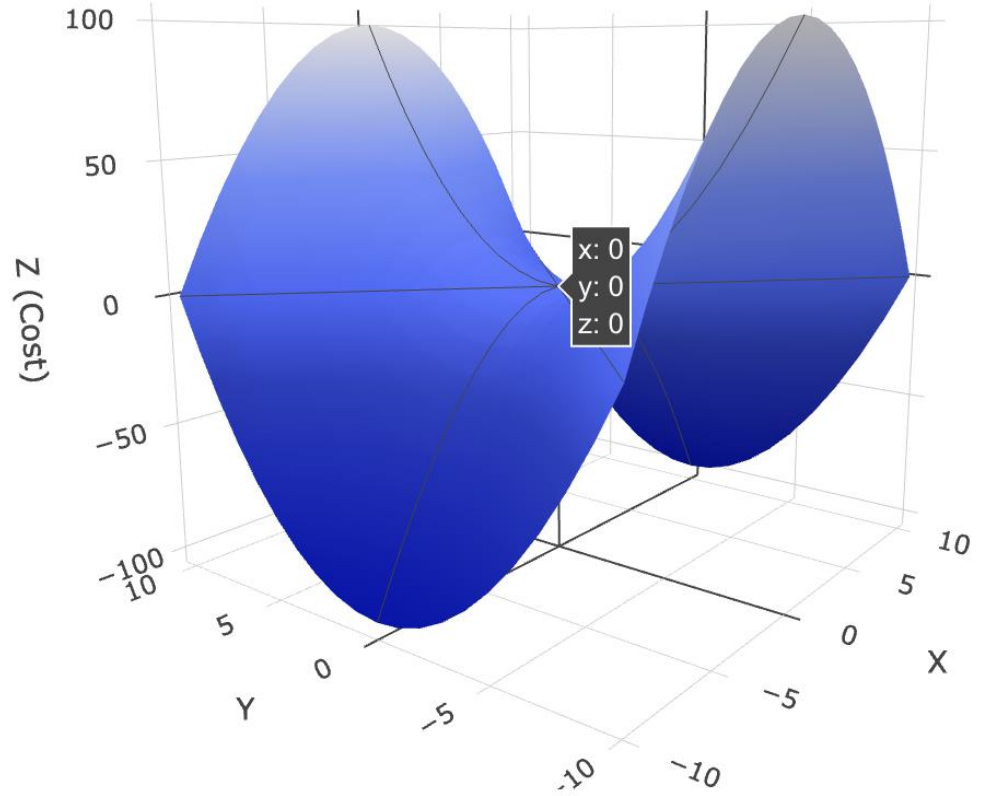
Concave

$f(x, y) = x^2 - y^2$

Neither

Saddle points





$$f(x, y) = x^2 - y^2$$

The hidden rule of Convex function

Defining The Core Concepts

Before diving in, let's clarify the two key players: the **set** (the domain) and the **function** itself. Both must meet specific criteria.

Convex Set

Any line drawn between two points within the set stays entirely inside the set. No holes, no gaps!

Non-Convex Set

A line connecting two points can pass through an area that is NOT part of the set.

Convex Function

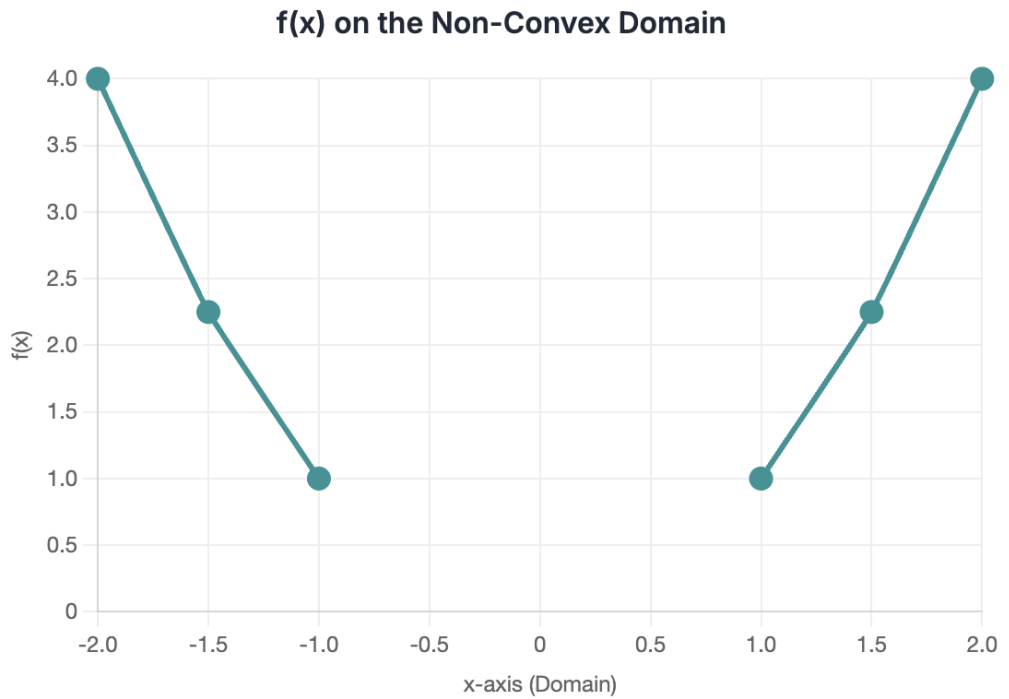
The line segment connecting any two points on its graph lies on or above the graph... but **only if its domain is a convex set**.

Example

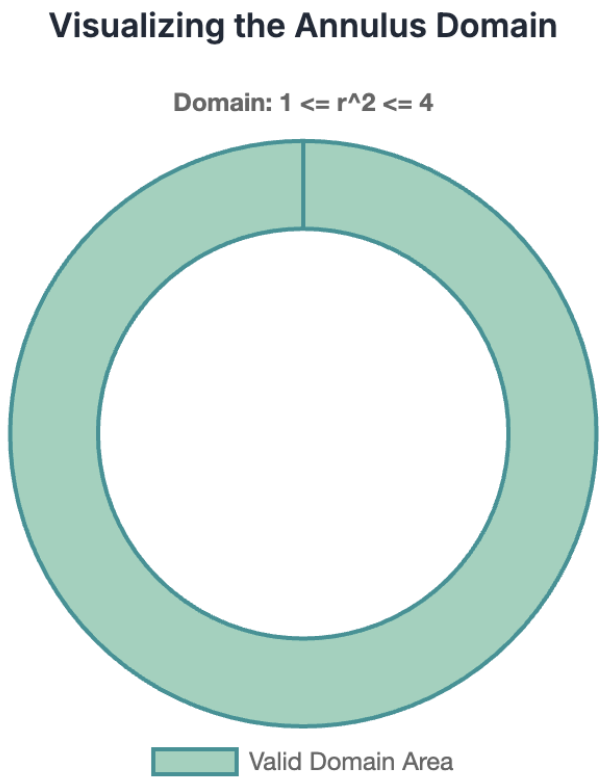
Case Study 1: The Disconnected Parabola

Let's examine the classic convex function, $f(x) = x^2$. Its shape is perfectly convex. But what happens when we give it a domain with a gap in the middle, like $D = [-2, -1] \cup [1, 2]$?

The function is disqualified from being convex simply because the domain is not a convex set. The point $x=0$, which lies on the line segment between $x_1=-1$ and $x_2=1$, is missing from the domain. This violation of the domain's convexity is the critical failure.



Example



Case Study 2: The Paraboloid on a "Donut"

Now consider the 2D function $f(x, y) = x^2 + y^2$. Its graph is a convex bowl shape. Let's restrict its domain to an annulus (a ring), defined by $1 \le x^2 + y^2 \le 4$.

This domain is not convex. As the visualization shows, a line connecting two points on opposite sides of the ring, like $P_1 = (-1.5, 0)$ and $P_2 = (1.5, 0)$, must pass through the center "hole," which is not part of the domain. Because the domain fails the convexity test, the function defined on it cannot be convex.

The hidden rule of convex function

The key takeaway is simple: both the function's formula and its domain must be convex.
If either one fails, the function is not convex. It's a package deal!

Some properties of convexity

- The sum of $\alpha f + \beta g$ is still convex for any $\alpha, \beta > 0$, if f, g are convex
- $f \circ g, where f \circ g = f(g(x))$ is convex if f is convex and g affine or

  f non-decreasing & convex and g convex
- $h(x) = \max\{f(x), g(x)\}$ is convex if g, f are convex

Some properties of convexity

- The sum of $\alpha f + \beta g$ is still convex for any $\alpha, \beta > 0$, if f, g are convex

- $f \circ g, where f \circ g = f(g(x))$ is convex if f is convex and g affine or

Let's pick the square loss $\ell\big(y, f_w(x)\big) = \big(y - f_w(x)\big)^2$. Which of the following statements is true?

(a) $L(w)$ is convex (b) $L(w)$ is convex for $f_w(x) = w^\top x$ (c) $L(w)$ is convex for $f_w(x) = \mathrm{relu}(w^\top x) \coloneqq \max(0, w^\top x)$

- $h(x) = \max\{f(x), g(x)\}$ is convex if g, f are convex

Some properties of convexity

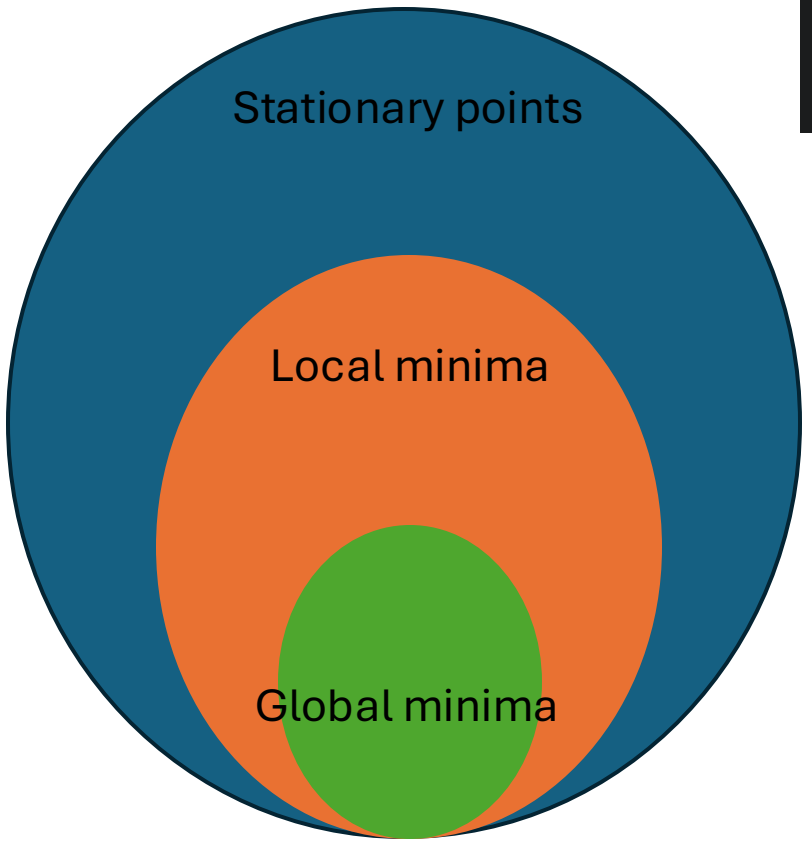
Which of the following statements are true:

1. All convex functions have only one local minima that is a global minima

2. A strictly convex function defined on all of $\mathbb{R}$ is guaranteed to have one unique global minima.

3. For a differentiable function f, the condition $\nabla f(x^*) = 0$ is sufficient to prove that x^* is a global minima."

Takeaways message



$$\nabla f(x^*) = 0$$

$$f(x^*) \leq f(x) \text{ for all } x \text{ in a neighborhood of } x^*$$

$$f(x^*) \leq f(x) \text{ for ALL } x$$

Stationary Point ⇔ Local minima ⇔ Global minima, if and only if the function is convex

Takeaway message

Stationary Point ? Saddle point

Feature	Stationary Point	Saddle Point
Type	**General Category** (umbrella term)	**Specific Type**
Gradient (Slope)	$\nabla f(x) = 0$ (Always flat)	$\nabla f(x) = 0$ (Always flat)
Curvature	Can curve up everywhere (min), down everywhere (max), or both.	**Must** curve up in some directions and down in others.
Behavior	A "flat spot" on the function.	A "flat spot" that is a minimum along one axis and a maximum along another.

Convergency of Gradient descent

Section 1: The Setup

Before we begin, we must define our problem and the conditions under which our proof holds. We aim to solve the optimization problem $\min_{x \in \mathbb{R}^n} f(x)$ using the Gradient Descent update rule: $x_{k+1} = x_k - \alpha \nabla f(x_k)$.

Core Assumption: Convexity

The function f is "bowl-shaped". If you draw a straight line between any two points on its graph, the function never rises above the line. This guarantees that any local minimum is a global minimum.

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Core Assumption: L-Smoothness

The function's gradient cannot change too erratically. L-smoothness means the function is bounded by a quadratic curve, preventing its slope from changing too quickly and allowing us to predict our progress.

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

Lipschitz Constant

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

L-smoothness:

$$f(y) - f(x) = \int_0^1 \nabla f(z(t))^T (y - x)\, dt$$

$$z(1) = y, z(0) = 0$$

$$f(y) - f(x) = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x) + \nabla f(x)\right)^T (y - x)\, dt$$

$$f(y) - f(x) = \underbrace{\int_0^1 \nabla f(x)^T (y - x)\, dt}_{\text{Integral A}} + \underbrace{\int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt}_{\text{Integral B}}$$

$$f(y) - f(x) = \nabla f(x)^T (y - x) + \text{Integral B}$$

L-smoothness:

$$f(y) - f(x) = \int_0^1 \nabla f(z(t))^T (y - x)\, dt$$

$z(1) = y, z(0) = 0$

$$f(y) - f(x) = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x) + \nabla f(x)\right)^T (y - x)\, dt$$

$$f(y) - f(x) = \underbrace{\int_0^1 \nabla f(x)^T (y - x)\, dt}_{\text{Integral A}} + \underbrace{\int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt}_{\text{Integral B}}$$

$$f(y) - f(x) = \nabla f(x)^T (y - x) + \text{Integral B}$$

Cauchy-Schwarz inequality:

$$a \cdot b < |a||b|$$

$$\left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x) \leq \left(L \cdot t\|y - x\|\right) \cdot \|y - x\| = Lt\|y - x\|^2$$

$$\text{Integral B} = \int_0^1 \left(\nabla f(z(t)) - \nabla f(x)\right)^T (y - x)\, dt \leq \int_0^1 Lt\|y - x\|^2\, dt$$

$$\text{Integral B} \leq \frac{L}{2}\|y - x\|^2$$

Strong Convexity

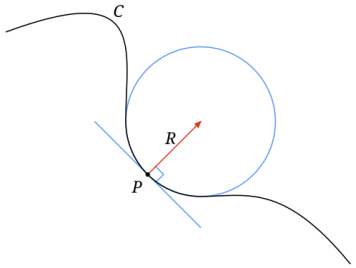
1st order condition

$$\underbrace{f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2}_{\text{Lower Bound (Floor)}} \leq f(y) \leq \underbrace{f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}\|y - x\|^2}_{\text{Upper Bound (Ceiling)}}$$

2nd order condition $\nabla^2 f(x) \geq \mu I$, positive curvature at every x

Osculating circle [edit]

Historically, the curvature of a differentiable curve was defined through the osculating circle, which is the circle that best approximates the curve at a point. More precisely, given a point P on a curve, every other point Q of the curve defines a circle (or sometimes a line) passing through Q and tangent to the curve at P. The osculating circle is the limit, if it exists, of this circle when Q tends to P. Then the *center* and the *radius of curvature* of the curve at P are the center and the radius of the osculating circle. The curvature is the reciprocal of radius of curvature. That is, the curvature is



$$\kappa = \frac{1}{R},$$

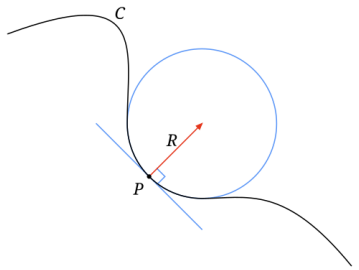
where R is the radius of curvature[5] (the whole circle has this curvature, it can be read as turn 2π over the length $2\pi R$).

This definition is difficult to manipulate and to express in formulas. Therefore, other equivalent definitions have been introduced.

Strong C

Osculating circle [edit]

Historically, the curvature of a differentiable curve was defined through the osculating circle, which is the circle that best approximates the curve at a point. More precisely, given a point P on a curve, every other point Q of the curve defines a circle (or sometimes a line) passing through Q and tangent to the curve at P. The osculating circle is the limit, if it exists, of this circle when Q tends to P. Then the *center* and the *radius of curvature* of the curve at P are the center and the radius of the osculating circle. The curvature is the reciprocal of radius of curvature. That is, the curvature is



$$\kappa = \frac{1}{R},$$

where R is the radius of curvature[5] (the whole circle has this curvature, it can be read as turn 2π over the length $2\pi R$).

This definition is difficult to manipulate and to express in formulas. Therefore, other equivalent definitions have been introduced.

$$+ \underbrace{\frac{L}{2}\|y-x\|^2}$$

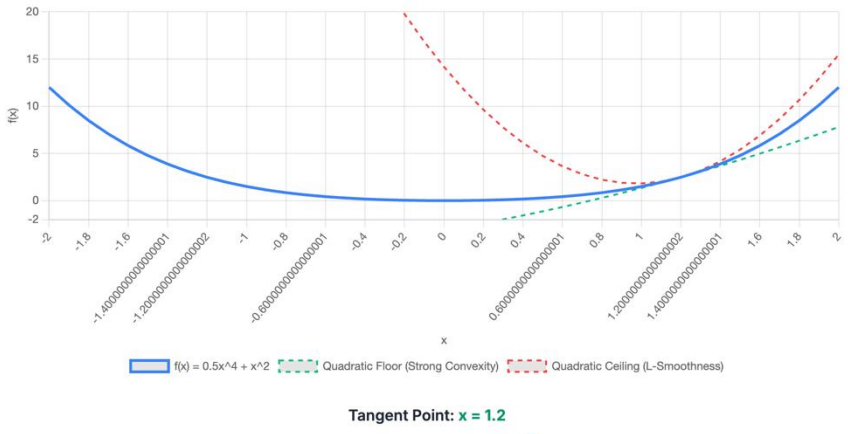
ing)

2nd order condition $\forall f(x) \geq \mu I$, positive curvature at every x

Example 1: A Strongly Convex & L-Smooth Function

Let's examine $f(x) = 0.5x^4 + x^2$. This function is strongly convex. Its curvature (second derivative) is $f''(x) = 6x^2 + 2$. The minimum value of the curvature is 2 (at x=0), so it's strongly convex with $\mu = 2$. It is also L-smooth on the plotted interval.

This means it's "sandwiched" between a quadratic floor (strong convexity) and a quadratic ceiling (L-smoothness). Use the slider to see how these bounds are tangent at point x but always contain the function.

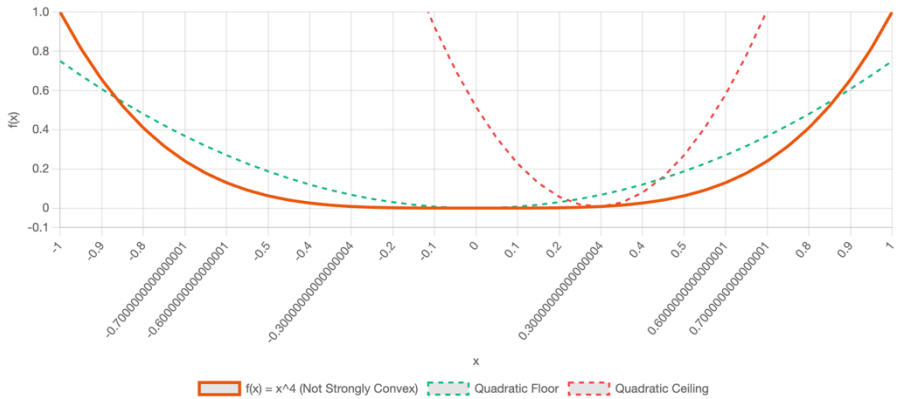


Tangent Point: x = 1.2

Example 2: A Convex (but not Strongly) Function

Now let's look at $f(x) = x^4$. This function is convex, but its curvature at the minimum, $f''(x) = 12x^2$, is zero, making it "too flat" to be strongly convex. The quadratic floor (green) "pokes through" the function (orange), violating the condition. The function is also not globally L-smooth, as its curvature is unbounded. However, on the interval we are plotting, it is L-smooth, allowing us to visualize the ceiling.

Use the sliders to see how no single $\mu > 0$ can create a floor, while the ceiling always holds true on this limited domain.



Proof

Objective: To prove that for a convex and L-smooth function f, the gradient descent algorithm finds a solution x^K such that $f(x^K) - f(x^*) \leq \epsilon$ in $O(1/\epsilon)$ iterations.

First, let's show that each step of gradient descent makes guaranteed progress in reducing the function value if we choose the right step size.

1. Start with the **Descent Lemma**:

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

2. Substitute the gradient descent update, $x_{k+1} - x_k = -\alpha \nabla f(x_k)$:

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

3. Factor out the gradient term:

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

Proof

1. Start with the **Descent Lemma**:

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

2. Substitute the gradient descent update, $x_{k+1} - x_k = -\alpha \nabla f(x_k)$:

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

3. Factor out the gradient term:

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

4. To guarantee that the function value decreases ($f(x_{k+1}) < f(x_k)$), we need the term $\alpha(1 - \frac{L\alpha}{2})$ to be positive. The simplest standard choice is setting the step size $\alpha = \frac{1}{L}$. This gives us:

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

Proof

Now, we'll build the main argument by connecting the function error to the distance from the optimum.

1. Start with the **convexity assumption**, letting $y = x^*$ and $x = x_k$:

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^T (x^* - x_k)$$

Rearranging gives us a bound on the function error, $e_k = f(x_k) - f(x^*)$:

$$e_k \leq \nabla f(x_k)^T (x_k - x^*)$$

2. Next, let's analyze the distance to the optimum, $\|x_{k+1} - x^*\|^2$:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f(x_k) - x^*\|^2$$

$$= \|x_k - x^*\|^2 - 2\alpha \nabla f(x_k)^T (x_k - x^*) + \alpha^2 \|\nabla f(x_k)\|^2$$

Proof

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^T (x^* - x_k)$$

Rearranging gives us a bound on the function error, $e_k = f(x_k) - f(x^*)$:

$$e_k \leq \nabla f(x_k)^T (x_k - x^*)$$

2. Next, let's analyze the distance to the optimum, $\|x_{k+1} - x^*\|^2$:

$$\|x_{k+1} - x^*\|^2 = \|x_k - \alpha \nabla f(x_k) - x^*\|^2$$

$$= \|x_k - x^*\|^2 - 2\alpha \nabla f(x_k)^T (x_k - x^*) + \alpha^2 \|\nabla f(x_k)\|^2$$

3. Rearrange this to isolate the term $\nabla f(x_k)^T (x_k - x^*)$:

$$\nabla f(x_k)^T (x_k - x^*) = \frac{1}{2\alpha} \left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2 + \alpha^2 \|\nabla f(x_k)\|^2 \right)$$

4. Finally, combine the results from steps 1 and 3:

$$e_k \leq \frac{1}{2\alpha} \left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2 \right) + \frac{\alpha}{2} \|\nabla f(x_k)\|^2$$

Proof

Bringing It All Together with a Telescoping S

1. Recall our two key inequalities (using $\alpha = 1/L$):
   - From Slide 4: $e_k \leq \frac{L}{2}\left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right) + \frac{1}{2L}\|\nabla f(x_k)\|^2$
   - From Slide 3: $\|\nabla f(x_k)\|^2 \leq 2L(f(x_k) - f(x_{k+1})) = 2L(e_k - e_{k+1})$

2. Substitute the second inequality into the first:

$$e_k \leq \frac{L}{2}\left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right) + (e_k - e_{k+1})$$

3. The e_k terms cancel out, leaving a surprisingly simple result!

$$e_{k+1} \leq \frac{L}{2}\left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right)$$

4. Now, sum this inequality from $k = 0$ to $K - 1$:

$$\sum_{k=0}^{K-1} e_{k+1} = \sum_{k=1}^{K} e_k \leq \sum_{k=0}^{K-1} \frac{L}{2}\left(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right)$$

Proof

5. The right-hand side is a **telescoping sum**, where intermediate terms cancel out:

$$\sum_{k=1}^{K} e_k \leq \frac{L}{2}\left(\|x_0 - x^*\|^2 - \|x_K - x^*\|^2\right) \leq \frac{L\|x_0 - x^*\|^2}{2}$$

Recall

4. To guarantee that the function value decreases ($f(x_{k+1}) < f(x_k)$), we need the term $\alpha(1 - \frac{L\alpha}{2})$ to be positive. The simplest standard choice is setting the step size $\alpha = \frac{1}{L}$. This gives us:

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

$$e_K = f(x_K) - f(x^*) \leq \frac{1}{K}\sum_{k=1}^{K} e_k$$

$$f(x_K) - f(x^*) \leq \frac{1}{K}\left(\frac{L\|x_0 - x^*\|^2}{2}\right)$$

Because the error is non-increasing, the final error must be less than or equal to the average of the errors:

To guarantee an error of at most ϵ, we need to run for K=O(1/ϵ)

Mini-batch SGD

Mini-batch SGD

Pseudocode

```
For each epoch:
  Shuffle the dataset
  For each mini_batch in dataset:
    // 1. Compute gradient over the mini-batch
    gradient_sum = 0
    for (x_i, y_i) in mini_batch:
      gradient_sum += compute_gradient(w, x_i, y_i)

    // 2. Update weights once per mini-batch
    w = w - learning_rate * (gradient_sum / batch_size)
}
```

The Best of Both Worlds

- **Efficient and fast.**
- **Smoother convergence** than SGD.
- Optimized for modern hardware like GPUs.

Why stochastic is preferable

Escaping Saddle Points: The Power of SGD's "Noise"

To understand one of the biggest advantages of Stochastic Gradient Descent (SGD), we first need to review some core concepts about the landscape of loss functions.

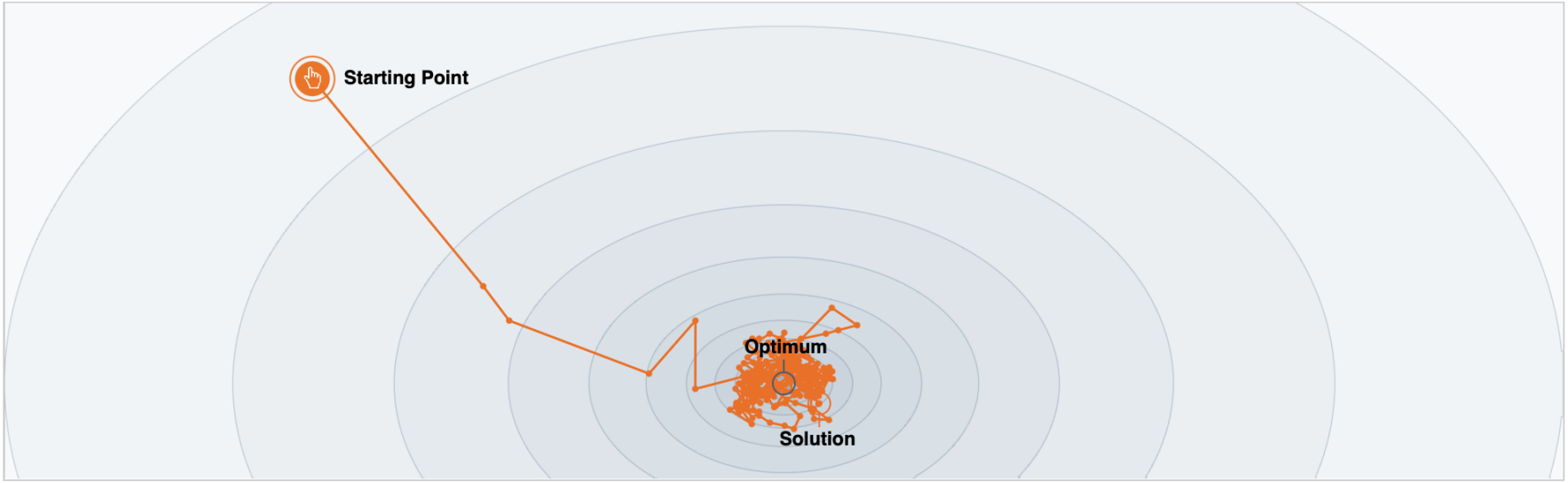
1. Convex Functions: The Ideal Case

A convex function is shaped like a single, perfect bowl. It has only one minimum—the global minimum. For these functions, standard Gradient Descent (GD) is guaranteed to find that minimum. The path is always downhill, and there's only one bottom. Linear regression is a classic example of an optimization problem with a convex loss function.

2. Non-Convex Functions: The Reality

The loss landscapes for deep neural networks are almost never convex. They are highly complex and can contain many local minima (small bowls) and, importantly, **saddle points**. This is where optimization gets tricky.

Dampen the oscillation



Momentum methods

- Momentum: the direction that previous time step keeps

- Momentum methods: Combine previous direction with negative gradient direction for some $\beta \in R$

$$w^{t+1} - w^{t} = \beta(w^{t} - w^{t-1}) - \alpha \nabla_{w} L(w^{t})$$

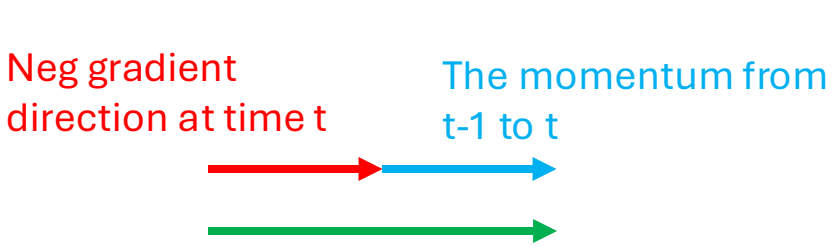
From physical perspective

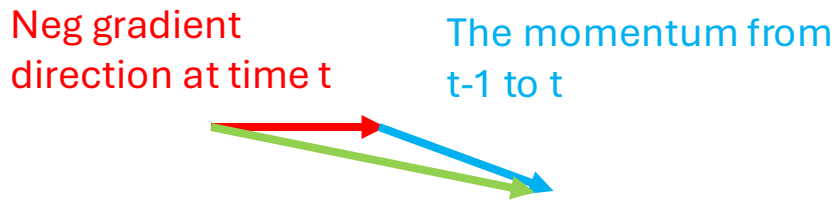
- We need to leverage the momentum of previous step:

$$v^{t+1} = \beta v^t + (1-\beta)\nabla_w L(w^t)$$

$$w^{t+1} = w^t - \alpha\, v^{t+1};\ \ v^0 = 0$$

$$w^{t+1} = w^t - \alpha\,\beta v^{t+1} - \alpha(1-\beta)\nabla_w L(w^t)$$





Decompose the momentum

- Let's give Gradient Descent a short memory:

$$w^{t+1} = w^{t} - \alpha\, v^{t+1};$$
$$v^{t+1} = \beta v^{t} + \nabla_{w} L(w^{t})$$

$\beta = 0 \Rightarrow$ GD. Usually, we set $\beta = 0.99$

The intuition behind the momentum

$$v_t = \beta v_{t-1} + g^t$$

Let's substitute the definition for $v_{t-1} = \beta v_{t-2} + g_{t-1}$

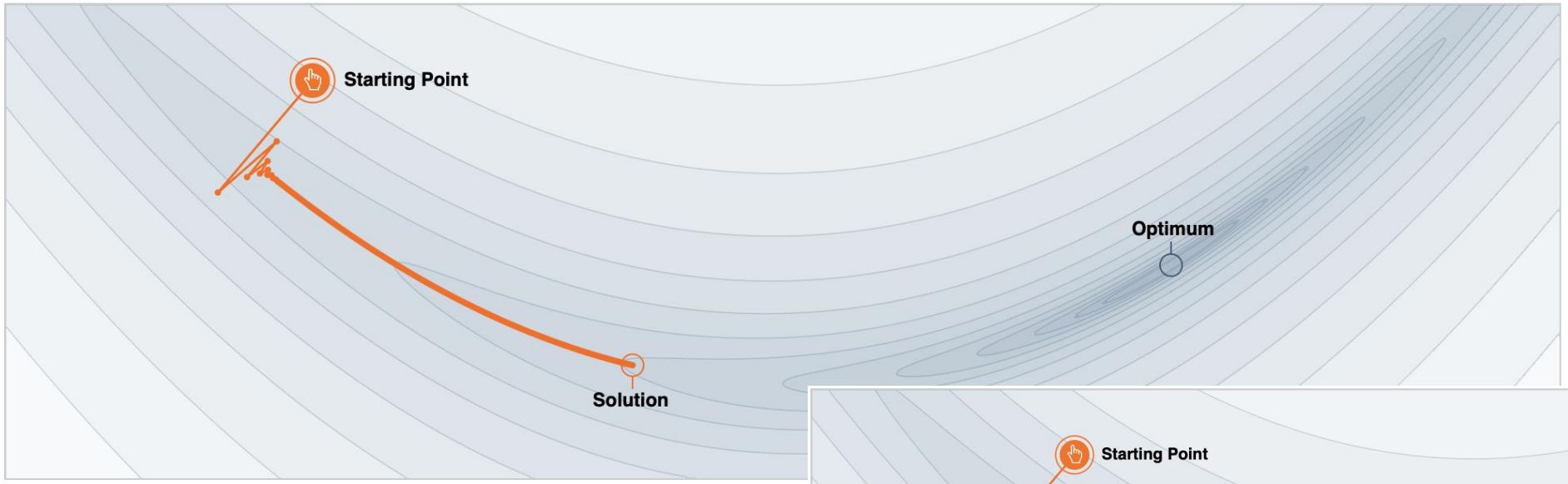
$$v_t = \beta(\beta v_{t-2} + g_{t-1}) + g_t = \beta^2 v_{t-2} + \beta g_{t-1} + g_t$$

Let's expanding to t=0:

$$v_t = \beta^{t-1} g_1 + \beta^{t-2} g_2 + \cdots + \beta g_{t-1} + g_t + v_0$$

This averaging process is what smooths out the path, cancels the oscillations, and builds up speed in a consistent direction

Examples

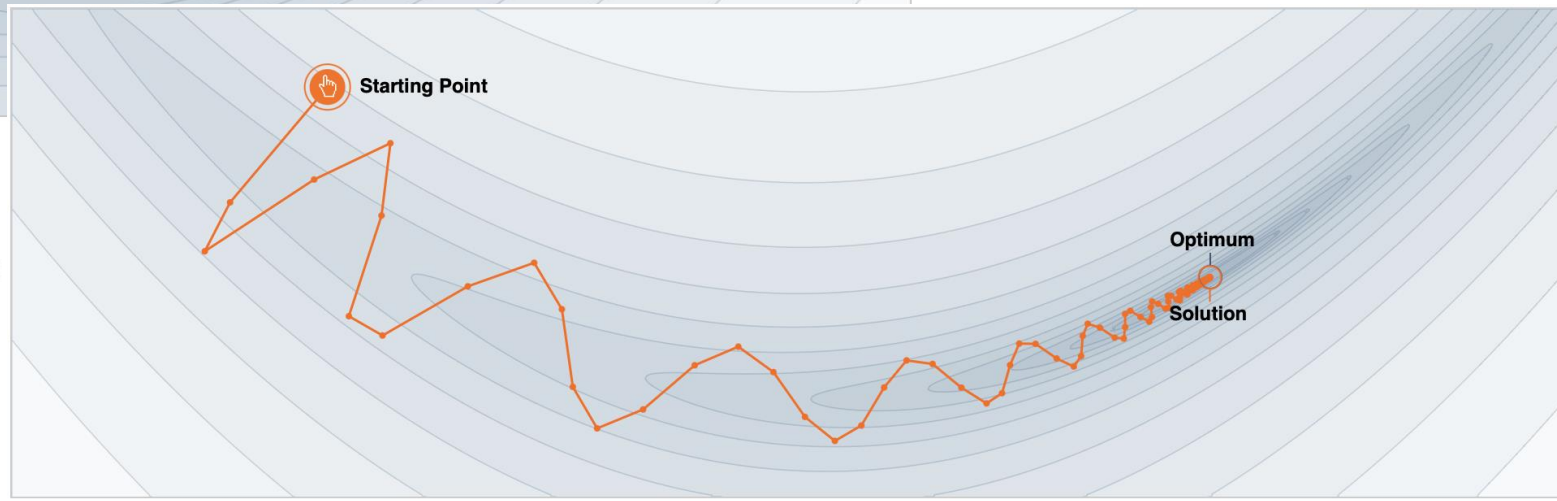


Step-size α = 0.0024

0 0.003 0.006

Momentum β = 0.0

0.00 0.500 0.990



Step-size α = 0.0024

0 0.003 0.006

Momentum β = 0.88

0.00 0.500 0.990

https://distill.pub/2017/momentum/

Nesterov Accelerated Gradient (NAG)

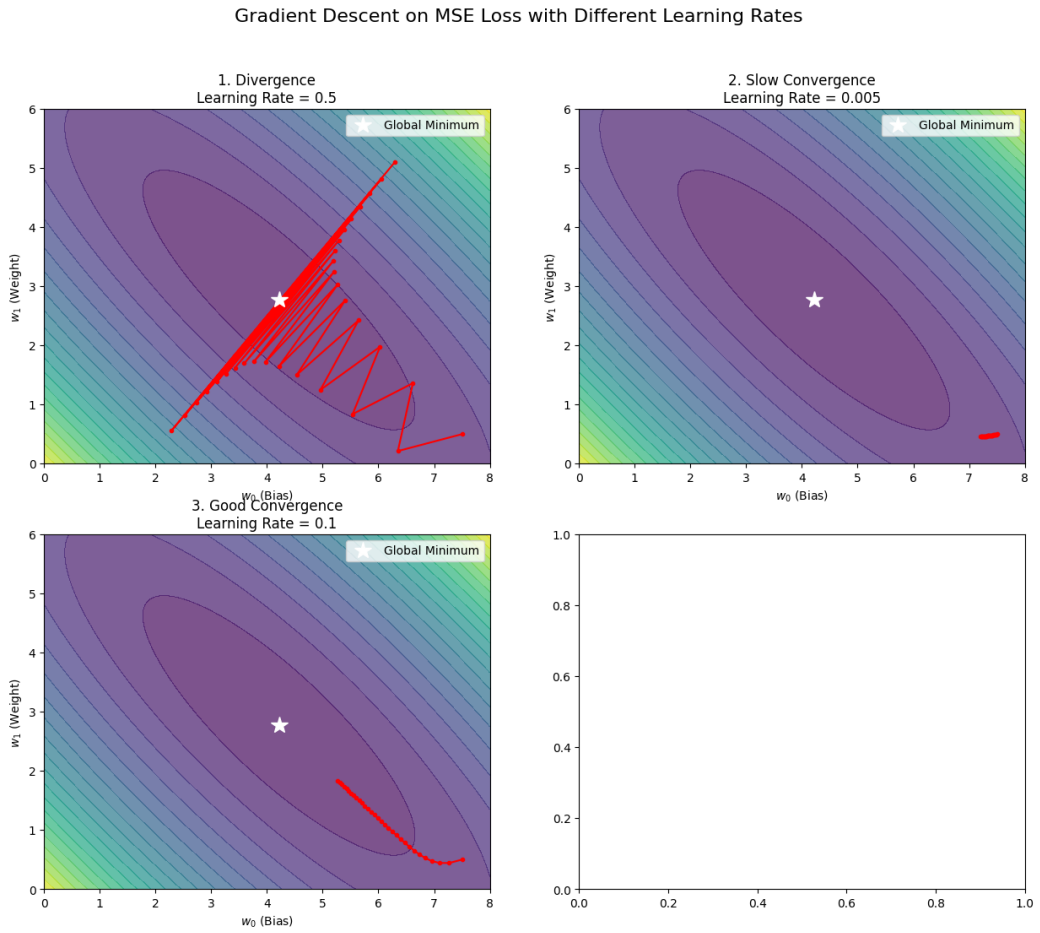
NAG is a clever improvement. Instead of calculating the gradient at the current position, it first takes a big "jump" based on the stored velocity, and then calculates the gradient at that "lookahead" position to make a correction.

$$w^{t+1} = w^{t} - \alpha\, v^{t};$$
$$v^{t} = \beta v^{t-1} + \nabla_{w} L(w^{t})$$

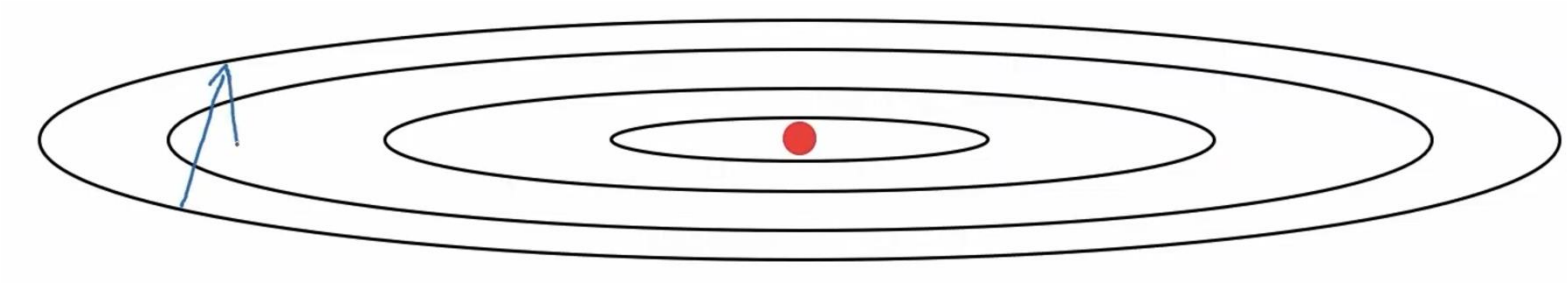
$$w^{t+1} = w^{t} - \alpha\, v^{t};$$
$$v^{t} = \beta v^{t-1} + \nabla_{w} L(w_{la})$$
$$w_{la} = w^{t} - \alpha\beta v^{t-1}$$

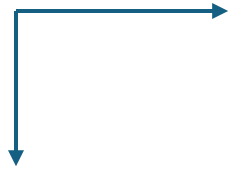
Speeding up convergence

- Rule of thumbs: Slow in steepest direction, while fast in flat region

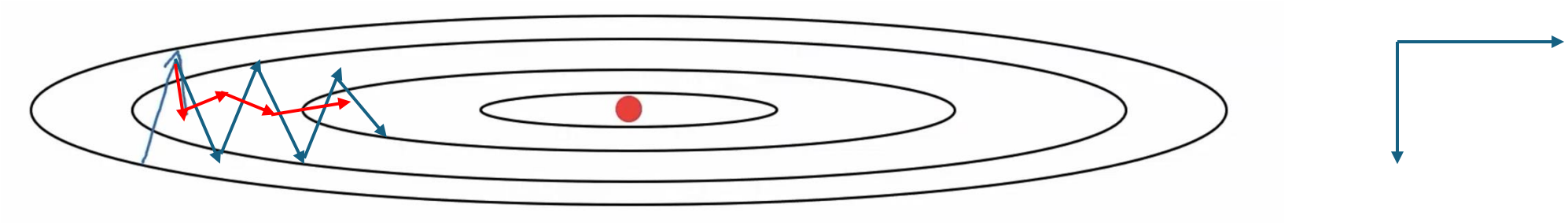


What else you can think of?





RMSprop: Root Mean Square Propagation



$$w^{t+1} = w^t - \alpha \frac{\nabla_w L(w^t)}{\sqrt{s^t + \epsilon}};$$

$$s^t = \beta s^{t-1} + (1-\beta)(\nabla_w L(w^t))^2$$

Adam: Adaptive Moment Estimation

Momentum $w^{t+1} = w^t - \alpha\, v^t;$

RMSprop $w^{t+1} = w^t - \frac{\alpha}{\sqrt{s^t + \epsilon}} \nabla_w L(w^t);$

$v^t = \beta_1 v^{t-1} + (1-\beta_1)\nabla_w L(w^t)$ $\quad s^t = \beta_2 s^{t-1} + (1-\beta_2)(\nabla_w L(w^t))^2$

Unbiased correction $\quad v^t_{corr} = \frac{1}{1-\beta_1^t} v^t \quad s^t_{corr} = \frac{1}{1-\beta_2^t} s^t$

Adam $\quad w^{t+1} = w^t - \frac{\alpha}{\sqrt{s^t_{corr}+\epsilon}} v^t_{corr}$

$\beta_1 = 0.9, \beta_2 = 0.999, \alpha$ is $tunable$

Comparison

Momentum

Core Idea

Accelerates gradient descent by adding a fraction of the past update vector to the current one.

Learning Rate

Global (one for all parameters).

Key Mechanism

Stores a moving average of gradients (first moment).

Best For

Improving convergence speed and overcoming local minima in many scenarios.

Main Weakness

Can struggle if parameters need vastly different learning rates.

RMSprop

Core Idea

Adapts the learning rate for each parameter based on the magnitude of recent gradients.

Learning Rate

Per-parameter and adaptive.

Key Mechanism

Stores a moving average of *squared* gradients (second moment).

Best For

Problems with noisy or sparse gradients, and non-stationary objectives (e.g., RNNs).

Main Weakness

Doesn't benefit from the acceleration that momentum provides.

Adam

Core Idea

Combines the ideas of both Momentum and RMSprop.

Learning Rate

Per-parameter and adaptive.

Key Mechanism

Stores moving averages of both gradients and squared gradients.

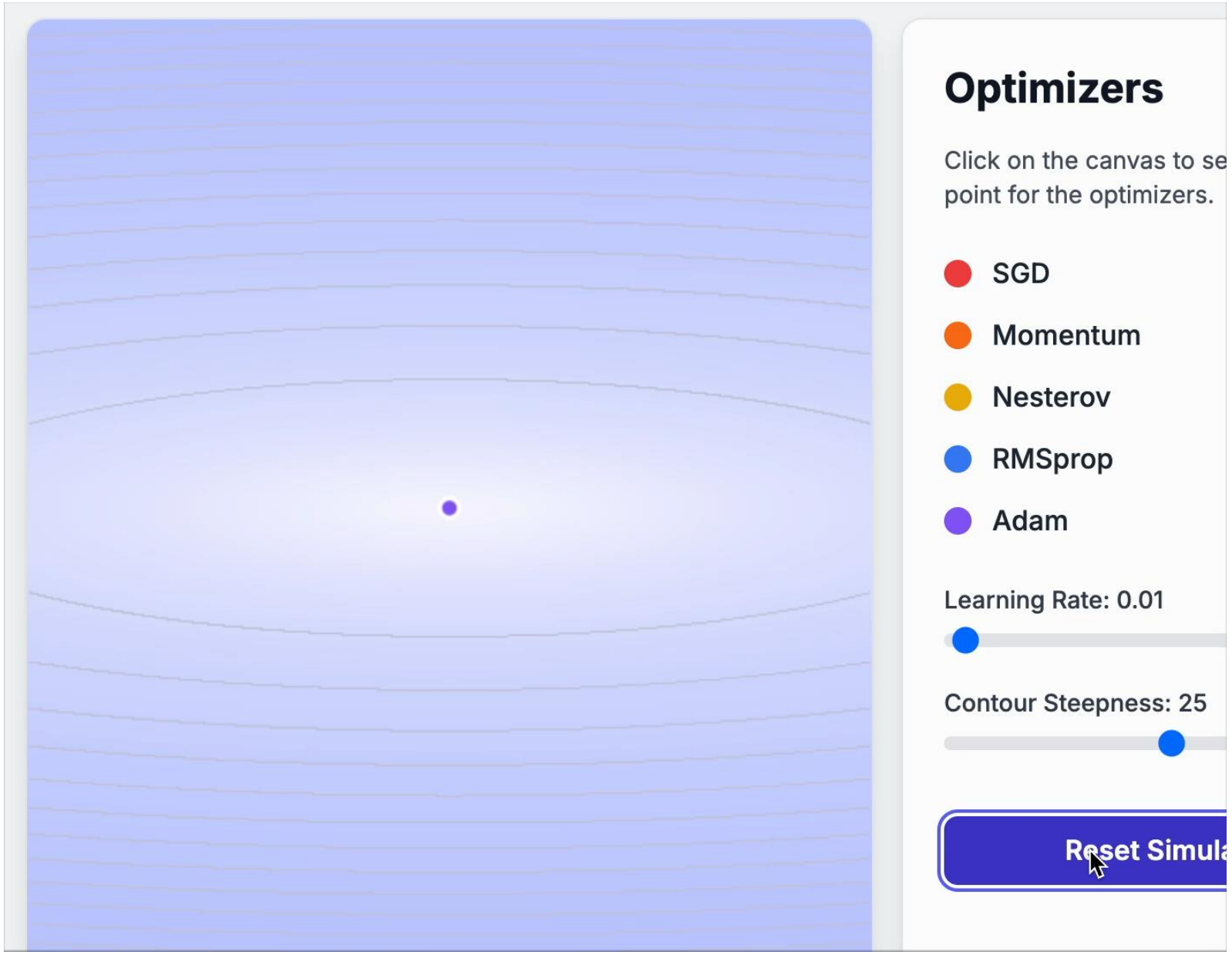
Best For

A robust, all-purpose default optimizer that works well for a wide variety of problems.

Main Weakness

Can sometimes converge to a less optimal solution than well-tuned SGD. Requires more memory.

Comparison


Optimizers
Click on the canvas to se
point for the optimizers.
SGD
Momentum
Nesterov
RMSprop
Adam
Learning Rate: 0.01
Contour Steepness: 25
Reset Simula

Takeaways messages

- **Reduced Oscillation:** Notice how the blue path (Momentum) is much smoother, especially in narrow "valleys". It dampens the zig-zag motion seen in the red path (standard SGD).
- **Faster Convergence:** The smoother path is often a more direct route to the minimum. Momentum builds up speed in the correct direction, leading to faster convergence.
- **Nesterov's Advantage:** By "looking ahead," NAG can slow down more effectively as it approaches the minimum, preventing overshooting and often leading to more stable training.
- **New Hyperparameter:** The main trade-off is the introduction of the momentum coefficient (β). This adds another parameter that needs to be tuned, but its default value (0.9) often works very well.

Takeaways messages

Practical Advice 💡

Start with Adam

For most deep learning applications, Adam is an excellent and safe first choice. Its default hyperparameter values often work very well out-of-the-box, making it a reliable workhorse.

Consider RMSprop for RNNs

RMSprop is known to perform particularly well for Recurrent Neural Networks (RNNs) and in reinforcement learning scenarios where objectives can be non-stationary.

Don't Forget SGD + Momentum

While Adam is the default, a well-tuned SGD with Momentum can sometimes find better, more generalizable minima. It's worth trying when fine-tuning or seeking peak performance.

More references

https://distill.pub/2017/momentum/

https://www.ruder.io/optimizing-gradient-descent/#momentum

Convex Optimization book: https://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
Chapter 2 and 3