Optimization

Gradient-Based Optimization for Machine Learning

Yasser Hifny

Faculty of Computers and Artificial Intelligence
Capital University

Overview

- Convex and non-convex functions
- Derivatives and methods for finding minima
- Gradient descent and its update rule
- Gradient descent examples
- Limitations of gradient descent: the zigzag effect
- Hessian-based optimization and Newton's method
- Adaptive learning-rate methods
- Momentum, AdaGrad, RMSprop, and Adam
- Assignment: minimizing a two-variable function using Adam

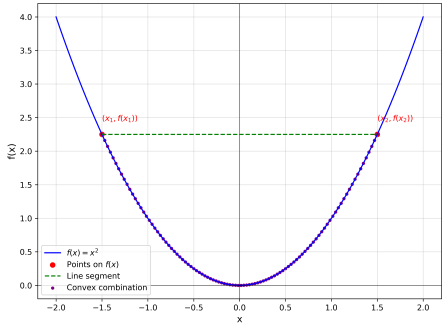
Convex Functions

A function $f(x)$ is convex if the line segment joining any two points on its graph lies above or on the graph.

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2), \qquad \lambda \in [0,1].$$

- Example: $f(x) = x^2$
- Example: $f(x) = e^x$
- Convexity is important because optimization of convex functions has a well-defined global minimum structure.

Illustration of Convexity



The line segment between two points lies above the convex function.

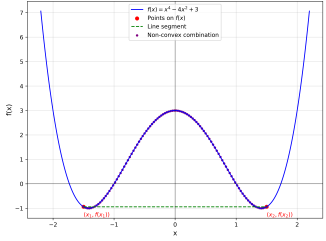
Non-Convex Functions

A function is non-convex when the convexity condition is violated for at least one pair of points.

- Non-convex functions may contain several local minima and maxima.
- Example are:

$$f(x) = \cos(x), \qquad f(x) = x^4 - 4x^2 + 3.$$

- Optimization may therefore depend on the initial point and optimization method.



Derivative

The derivative measures the rate of change of a function and can be interpreted as the slope of the tangent line.

$$f'(x) = \frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

- A function is not differentiable at a point when the derivative does not exist there.
- For several variables, partial derivatives measure the rate of change with respect to individual variables.

Methods for Finding a Minimum

We discusses three basic approaches:

1. **Sketch the function:** visually identify the lowest point.
2. **Find an analytical solution:** differentiate and solve

$$f'(x) = 0.$$

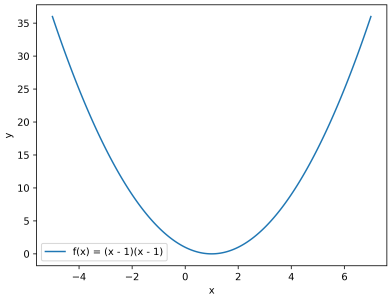
3. **Use gradient descent:** iteratively move in the opposite direction of the gradient.

For the quadratic function

$$f(x) = (x - 1)^2,$$

the global minimum occurs at $x = 1$.

A Simple Quadratic Minimum

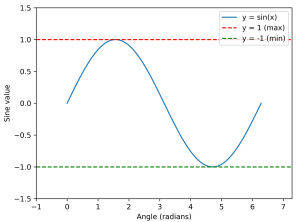


A simple quadratic function with a single global minimum.

Why Set the First Derivative to Zero?

At a differentiable local maximum or minimum, the tangent is horizontal.

$$f'(x) = 0.$$



The maximum and minimum occur at points where the slope is zero.

Analytical Solution: Quadratic Example

For

$$f(x) = (x-1)^2,$$

the derivative is

$$f'(x) = 2(x-1).$$

Setting the derivative to zero:

$$2(x-1) = 0 \quad \Longrightarrow \quad x = 1.$$

Thus, the minimum occurs at

$$\boxed{x = 1}.$$

Limitation

Analytical minimization does not scale well to the large and complex optimization problems commonly encountered in machine learning.

Gradient Descent

Gradient descent starts from an initial guess and repeatedly moves in the opposite direction of the gradient.

- The gradient gives the direction of steepest ascent.
- Moving against the gradient produces a descent direction.
- The learning rate η controls the step size.
- The method can be used when an analytical solution is unavailable or impractical.

For one variable:

$$g(x) = \frac{df(x)}{dx}.$$

Gradient Descent: Update Rule

Starting from $x^{(0)}$:

$$g(x) = \left.\frac{df(x)}{dx}\right|_{x^{(0)}}.$$

The parameter is updated as

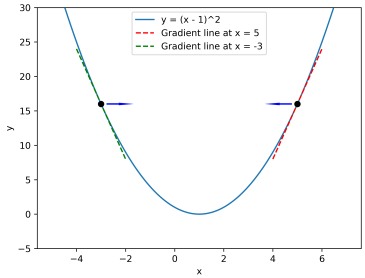
$$\boxed{x^{(1)} = x^{(0)} - \eta g(x)} \qquad \eta > 0.$$

The process is repeated until the gradient is close to zero or a maximum number of iterations is reached.

Key idea

The update moves in the direction opposite to the current gradient.

Gradient Descent on a Quadratic



The gradient is positive on the ascending side and negative on the descending side; the updates move toward the minimum.

Gradient Descent Algorithm

1. Initialize the parameters with $x = x^{(0)}$.
2. Compute the gradient at the current point.
3. Update the parameters:
$$x^{(t+1)} = x^{(t)} - \eta g^{(t)}.$$
4. Repeat until convergence or a maximum number of iterations.

The method is widely used to minimize machine-learning loss functions.

- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent
- Momentum gradient descent

Example: One-Variable Quadratic

Consider

$$f(x) = (x-1)^2, \qquad g(x) = 2(x-1).$$

We use these hyper-parameters:

$$x = 3, \qquad \eta = 0.001, \qquad \text{epochs} = 50000.$$

```
def grad(x):
    return 2.0 * (x-1.0)

x = 3.0
eta = 0.001
epochs = 50000

for i in range(epochs):
    x -= eta * grad(x)

print(x)
```

The iterations converge toward the minimum $x = 1$.

Example: Two-Variable Quadratic

Consider

$$f(x_1, x_2) = (x_1 - 2)^2 + 10(x_2 + 3)^2.$$

Its partial derivatives are

$$\frac{\partial f}{\partial x_1} = 2(x_1 - 2), \qquad \frac{\partial f}{\partial x_2} = 20(x_2 + 3).$$

We use a separate learning rate for each variable:

$$\eta_{x_1} = 0.1, \qquad \eta_{x_2} = 0.05.$$

Motivation

Different directions can have different scales and curvatures, so a single learning rate may cause convergence problems.

Gradient, Contours, and the Zigzag Effect

For a scalar function, the gradient points in the direction of steepest descent.
A contour connects points having the same function value:

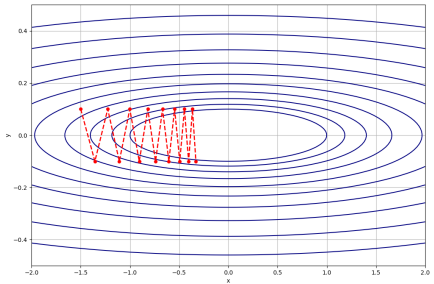
$$f(x, y) = c.$$

The gradient is perpendicular to the contour.

Zigzag effect

When the loss surface has different curvatures in different directions, successive steepest-descent steps can overshoot the direct path toward the minimum, producing a zigzag trajectory.

Zigzag Effect



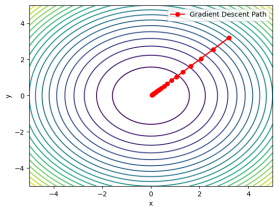
Different curvatures can cause gradient descent to oscillate across the direction toward the minimum.

When the Zigzag Effect Does Not Occur

For

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

the contours are circular and the curvature is equal in all directions.
Gradient descent can therefore follow a direct path toward the minimum.



Circular contours do not produce the zigzag behavior.

Second-Order Approximation

A second-order Taylor approximation is

$$f(\mathbf{x}) \approx f(\mathbf{x}^{(t)}) + (\mathbf{x} - \mathbf{x}^{(t)})^T \mathbf{g} + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(t)})^T \mathbf{H} (\mathbf{x} - \mathbf{x}^{(t)}).$$

The Hessian contains the second-order partial derivatives:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}.$$

Newton's Method

Let

$$\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}^{(t)}.$$

The quadratic approximation gives

$$\Delta f \approx \Delta \mathbf{x}^T \mathbf{g} + \frac{1}{2} \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x}.$$

Differentiating with respect to $\Delta \mathbf{x}$:

$$\mathbf{g} + \mathbf{H} \Delta \mathbf{x} = 0.$$

Hence,

$$\boxed{\Delta \mathbf{x} = -\eta \mathbf{H}^{-1} \mathbf{g}}.$$

When $\mathbf{H}$ is the identity matrix, this reduces to the gradient-descent update.

Hessian-Based Methods: Trade-Off

- Newton's method and quasi-Newton methods use second-order information.
- The Hessian can adjust both the direction and scale of the step.
- This can reduce the zigzagging effect.
- The main cost is the computational complexity of calculating or approximating the Hessian.

Practical consideration

The choice of optimization method involves a trade-off between the information used to guide the update and its computational cost.

Adaptive Learning Rate

Adaptive learning-rate methods adjust the learning rate for each parameter during optimization.
They use information such as:

- the history of gradients,
- the magnitude of gradients,
- accumulated or exponentially averaged gradient statistics.

The goal is to adapt the step size to the shape of the loss surface and reduce oscillations or zigzagging.

Momentum

Momentum accumulates a smoothed direction from previous gradients:

$$m^{(t)} = \beta m^{(t-1)} + (1-\beta) g^{(t)},$$

$$\boxed{\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta m^{(t)}}.$$

- $g^{(t)}$: current gradient
- $m^{(t)}$: accumulated gradient direction
- β: momentum coefficient
- η: learning rate

AdaGrad

AdaGrad accumulates squared gradients:

$$v^{(t)} = v^{(t-1)} + g^{(t)} \odot g^{(t)}.$$

The update is

$$\boxed{\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{\eta}{\sqrt{v^{(t)}} + \epsilon} g^{(t)}}.$$

Parameters with larger accumulated squared gradients receive smaller effective learning rates.

RMSprop

RMSprop replaces AdaGrad's cumulative sum with an exponentially decaying average:

$$v^{(t)} = \beta v^{(t-1)} + (1-\beta) g^{(t)} \odot g^{(t)}.$$

The update is

$$\boxed{\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \frac{g^{(t)}}{\sqrt{v^{(t)}} + \epsilon}}.$$

This prevents the accumulated squared gradients from growing without bound.

Adam

Adam combines momentum-like first-moment estimation with RMSprop-like second-moment estimation:

$$m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)},$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2) g^{(t)} \odot g^{(t)}.$$

Bias correction:

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t}, \qquad \hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}.$$

Update:

$$\boxed{\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \frac{\hat{m}^{(t)}}{\sqrt{\hat{v}^{(t)}} + \epsilon}}.$$

Bias Correction

- At early steps (small t): β_1^t is close to 1 (e.g., $0.9^1 = 0.9$), making the denominator $(1 - 0.9) = 0.1$ small. Dividing by this small number scales up the moment estimate, compensating for the zero-initialization drag.
- At later steps (large t): As training progresses and t grows large, β_1^t approaches 0. The denominator $(1 - \beta_1^t)$ approaches 1, meaning the bias correction naturally fades away once the optimizer has accumulated enough historical gradient data.
- Without bias correction, the initial steps of the Adam optimizer would move too sluggishly or erratically, leading to suboptimal convergence at the start of training.

Optimization Methods at a Glance

Method	**Main information**	**Main mechanism**
Gradient descent	First-order gradient	Fixed learning rate
Momentum	Gradient history	Smoothed update direction
AdaGrad	Accumulated squared gradients	Parameter-wise scaling
RMSprop	Exponential squared-gradient average	Adaptive scaling
Adam	First + second moments	Adaptive, bias-corrected update
Newton	Gradient + Hessian	Curvature-aware update

Summary

- Optimization is central to machine-learning model training.
- Convexity describes an important property of optimization landscapes.
- Derivatives identify stationary points and provide local information about a function.
- Gradient descent moves opposite to the gradient:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \mathbf{g}^{(t)}.$$

- Taylor expansion explains why the negative-gradient direction decreases the objective locally.
- The zigzag effect motivates curvature-aware and adaptive methods.
- Momentum, AdaGrad, RMSprop, and Adam adapt gradient updates using information from previous iterations.

Assignment

Using the Adam algorithm, find the minimum of

$$\boxed{f(x_1, x_2) = (x_1 - 2)^2 + 10(x_2 + 3)^2}.$$

- Derive $\partial f/\partial x_1$ and $\partial f/\partial x_2$.
- Initialize x_1 and x_2.
- Apply the Adam update iteratively.
- Stop when the parameters and/or objective function converge.
- Report the final x_1, x_2, and $f(x_1, x_2)$.

Expected analytical minimizer: $(x_1, x_2) = (2, -3)$.

Thank You

Questions?