Gradient Descent

Uwni 党鸦 uwni@example.com

Optimization problems can be divided into minimization and maximization categories, and maximization problems can always be transformed into equivalent minimization problems. Therefore, in the following text, we will focus on minimization problems. Gradient descent is an iterative optimization algorithm used to find the local minimum of a function. It gradually approaches the minimum point by moving in the opposite direction of the function's gradient.

1 EQUIVALENT PROBLEMS

First, let us prove some lemmas about monotonicity.

Proposition 1 (Order Preservation of Strictly Monotonic Functions) If f is a strictly increasing function, then

$$x_1 < x_2 \leftrightarrow f(x_1) < f(x_2)$$

That is, when the output strictly increases, the input must strictly increase.

Proof . (→) This is the definition of strictly increasing, $x_1 < x_2 \to f(x_1) < f(x_2)$
(←) By contradiction, if $x_1 \geq x_2$ then $f(x_1) \geq f(x_2)$, which is a contradiction. ■

Proposition 2 (Strictly Monotonic Function → Injection) If f is a strictly monotonic function, then f is injective. That is,

$$x_1 = x_2 \leftrightarrow f(x_1) = f(x_2)$$

Proof . (→) This follows from the definition of functions.
(←) We need to prove injectivity. Taking f strictly increasing on X as an example. Let $x_1, x_2 \in X, f(x_1) = f(x_2)$. Suppose $x_1 < x_2$, then by strict monotonicity, $x_1 < x_2 \to f(x_1) < f(x_2)$, which is a contradiction. Similarly $x_1 \not> x_2$, therefore $x_1 = x_2$. ■

Proposition 3 (Strictly Monotonic Functions Preserve Extreme Points) Let $Y \subseteq \mathbf{R}$, $f : X \to Y$ be an arbitrary function, and $g : Y \to \mathbf{R}$ be a strictly increasing function. Then $g \circ f$ and f have the same extreme points. Conversely, they have opposite extreme points.

Proof . By the lemma, we know $x_1 \leq x_2 \leftrightarrow g(x_1) \leq g(x_2)$ Thus for some point $x^* \in X$, $\exists \delta > 0, \forall x \in U(x^*, \delta)$

$$f(x^*) \leq f(x) \leftrightarrow g(f(x^*)) \leq g(f(x))$$

This proves that a minimum point of f is also a minimum point of $g \circ f$, and a minimum point of $g \circ f$ is also a minimum point of f. ■

Based on this, for the optimization problem

$$\arg\min f(x)$$

it always has the same solution as $\arg\min g \circ f(x)$, if g is a strictly increasing function. Or $\arg\max g \circ f(x)$, if g is a strictly decreasing function.

Example 1

Consider an interesting matrix optimization problem that demonstrates the equivalence of different objective functions under monotonic transformations.

Let $\boldsymbol{X} = \left(x_{ij}\right)_{n\times n}$ be a non-negative matrix with column sum constraints: $\sum_{i=1}^{n} x_{ij} = c$ for all j (where c is a constant).

Define two objective functions:

$$f_1(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} x_{ii}}{\sum_{i\neq j} x_{ij}}$$

(diagonal elements / off-diagonal elements)

$$f_2(\boldsymbol{X}) = \frac{\sum_{i=1}^{n} x_{ii}}{\sum_{i,j=1}^{n} x_{ij}}$$

(diagonal elements / all elements)

Due to the column sum constraint, the total sum of all elements is: $\sum_{i,j=1}^{n} x_{ij} = nc$

Therefore: $f_2(\boldsymbol{X}) = \left(\sum_{i=1}^{n} x_{ii}\right)/(nc)$

The sum of off-diagonal elements is: $\sum_{i\neq j} x_{ij} = nc - \sum_{i=1}^{n} x_{ii}$

So: $f_1(\boldsymbol{X}) = \left(\sum_{i=1}^{n} x_{ii}\right)/\left(nc - \sum_{i=1}^{n} x_{ii}\right)$

Key Observation: Let $s = \sum_{i=1}^{n} x_{ii}$, then:

- $f_2 = s/(nc)$
- $f_1 = s/(nc - s)$

These two functions have a monotonic relationship: $f_1 = f_2/(1 - f_2) \cdot 1/n$

Since f_1 is a strictly increasing function of f_2 (in the range $f_2 < 1$), we have:

The optimization problems $\max f_1(\boldsymbol{X})$ and $\max f_2(\boldsymbol{X})$ are equivalent and have the same optimal solution.

2 UNCONSTRAINED CASE

$\mathbf{R}^n$ is the n-dimensional Euclidean space. $\boldsymbol{x} \in \mathbf{R}^n$, $f : \mathbf{R}^n \to \mathbf{R}$ is a differentiable function. Finding the minimum point and minimum value of f is the optimization problem

$$\min_{\boldsymbol{x} \in \mathbf{R}^n} f(\boldsymbol{x})$$

The iterative equation is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{\nabla} f(\boldsymbol{x}_k)$$

where the step size α is a positive number that determines the update magnitude at each iteration. $\boldsymbol{\nabla} f(\boldsymbol{x}_k)$ is the gradient of function f at point $\boldsymbol{x}_k$. The algorithm's goal is to make $\boldsymbol{x}_k \to \boldsymbol{x}^*$ as $k \to \infty$ where $\boldsymbol{x}^* \in \arg\min f(\boldsymbol{x})$. that is to say, $\boldsymbol{x}^*$ is a minimum. The pseudocode is as follows:

```
Input: initial point $\boldsymbol{x}_0$, step length $\alpha > 0$, tolerance $\varepsilon > 0$, max iterations $N$
Output: $\boldsymbol{x}^*$
$k \leftarrow 0$
while $k < N$
    $\boldsymbol{g}_k \leftarrow \boldsymbol{\nabla} f(\boldsymbol{x}_k)$
    $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \alpha \boldsymbol{g}_k$
    if $f(\boldsymbol{x}_{k+1}) < \varepsilon$ then
        return $\boldsymbol{x}_{k+1}$
    $k \leftarrow k + 1$
return $\boldsymbol{x}_k$
```

Algorithm 1: Pseudocode of Gradient Descent

Let us look at an example

$$\min_{(x_1, x_2) \in \mathbf{R}^2} f(x_1, x_2)$$

where $f(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2$. First, we know through analytical methods that for $x_1, x_2 \in \mathbf{R}$

$$x_1^2 + x_1 x_2 + x_2^2 = (x_1, x_2) \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq 0$$

Equality holds if and only if $x_1 = 0, x_2 = 0$. Therefore, the minimum value 0 is achieved at the origin. Next, we use gradient descent to solve this.

$$\boldsymbol{\nabla} f \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ 2x_2 + x_1 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{k+1} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_k - \alpha \begin{pmatrix} 2x_1 + x_2 \\ 2x_2 + x_1 \end{pmatrix}_k$$

We set the initial condition as $x_0 = (1.0, 2.0)^{\mathrm{T}}$, step size $\alpha = 0.1$, and stopping condition as $f \leq 10^{-20}$. After 212 iterations, the function value reaches 9.925765507684842e-21. The trajectory left by each iteration in the feasible region is shown in the figure below. The black solid lines in the figure are the contour lines of function f, and the arrows indicate the gradient field. The coloring indicates the magnitude of the function value, with darker colors representing larger function values. The red points represent the positions updated at each iteration, and the connecting lines are the iteration trajectories. It can be seen that each iteration moves opposite to the gradient direction with step size proportional to the gradient magnitude, and the iteration points gradually approach the origin—the theoretical minimum point.

When we increase the step size to 0.4, after 44 iterations, the function value reaches 7.503260807194337e-21.

When we increase the step size to 0.5, after 35 iterations, the function value reaches 5.929230630780102e-21.

If the step size is increased to 0.6, it leads to divergence. Therefore

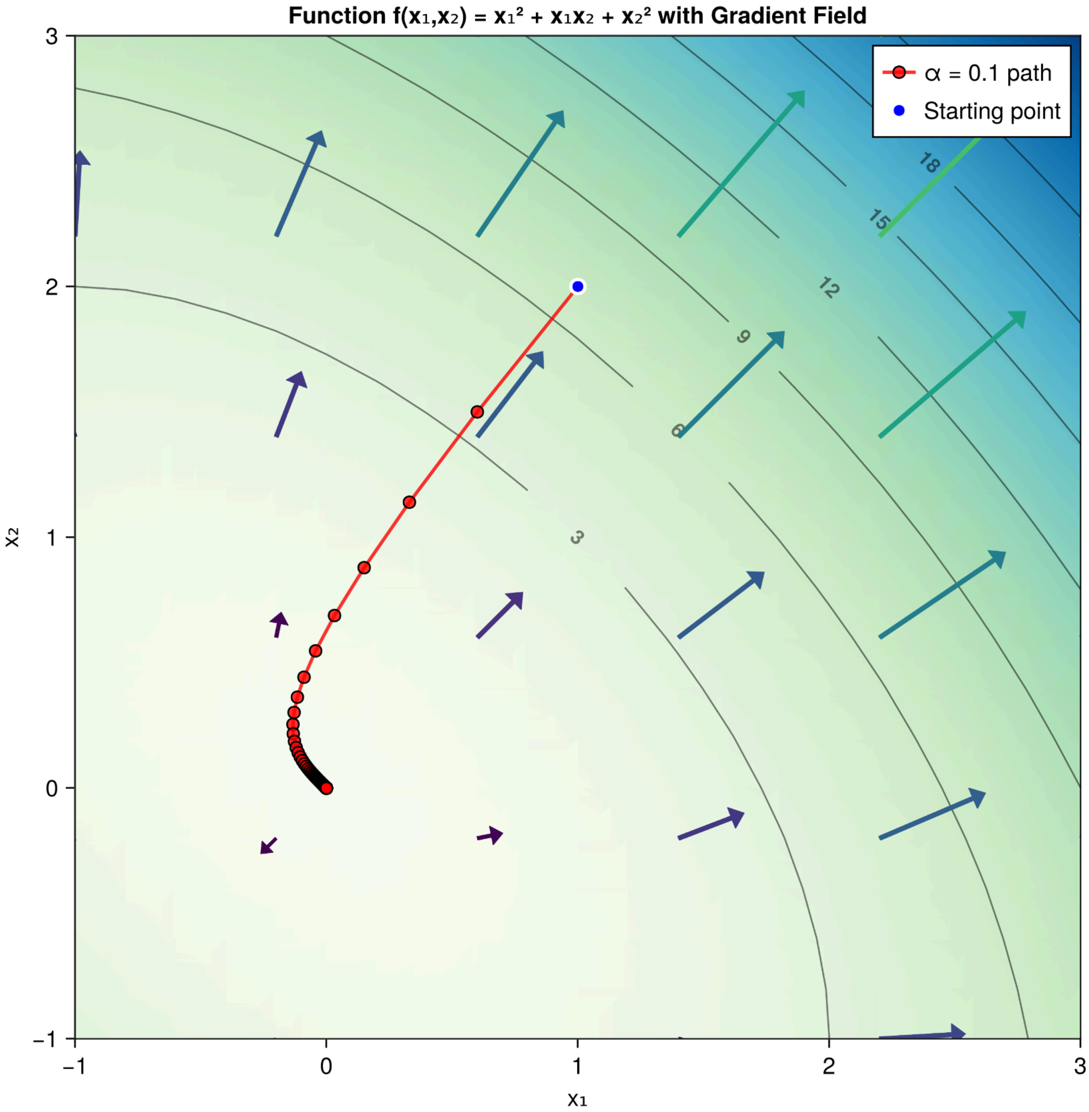

Figure 1: Gradient Descent Visualization

It is not difficult to see that the iteration speed is related to the step size, which can lead to divergence. Therefore, we need to choose an appropriate step size to ensure convergence.

3 CONSTRAINED CASE

When dealing with constrained optimization problems, we need to find the minimum of a function $f(\boldsymbol{x})$ subject to constraints. The general form is:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbf{R}^n} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \le 0,\ \ i = 1, 2, ..., m \\ & h_j(\boldsymbol{x}) = 0,\ \ j = 1, 2, ..., l \end{aligned}$$

For constrained problems, we cannot simply move in the negative gradient direction as this may violate the constraints. Instead, we need to project the gradient onto the feasible region or use penalty methods.

3.1 Projected Gradient Method

The projected gradient method modifies the standard gradient descent by projecting each iteration onto the feasible set $\mathcal{C}$:

$$\boldsymbol{x}_{k+1} = \Pi_{\mathcal{C}}(\boldsymbol{x}_k - \alpha \boldsymbol{\nabla} f(\boldsymbol{x}_k))$$

where $\Pi_{\mathcal{C}}$ denotes the projection operator onto the constraint set $\mathcal{C}$.

The pseudocode for the projected gradient method is:

Algorithm Projected Gradient Method for minimize f subject to $\boldsymbol{x} \in \mathcal{C}$

Input: initial point $\boldsymbol{x}_0 \in \mathcal{C}$, step length $\alpha > 0$, tolerance $\varepsilon > 0$, max iterations N

Output: $\boldsymbol{x}^*$

```
$k \leftarrow 0$
while $k < N$
    $\boldsymbol{g}_k \leftarrow \boldsymbol{\nabla} f(\boldsymbol{x}_k)$
    $\boldsymbol{y}_{k+1} \leftarrow \boldsymbol{x}_k - \alpha \boldsymbol{g}_k$
    $\boldsymbol{x}_{k+1} \leftarrow \Pi_{\mathcal{C}}(\boldsymbol{y}_{k+1})$
    if $f(\boldsymbol{x}_{k+1}) < \varepsilon$ then
        return $\boldsymbol{x}_{k+1}$
    end
    $k \leftarrow k + 1$
end
return $\boldsymbol{x}_k$
```

3.2 Example: Linear Constraint

Consider the optimization problem:

$$\begin{aligned} \min_{(x_1, x_2) \in \mathbf{R}^2} \quad & f(x_1, x_2) \\ \text{subject to} \quad & x_2 = 1 \end{aligned}$$

where $f(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2$ (same as the unconstrained case).

The feasible set is the line $\mathcal{C} = \{(x_1, x_2) : x_2 = 1\}$. The unconstrained minimum $(0, 0)$ is not feasible, so we expect the constrained optimum to lie on the constraint.

The gradient is the same as in the unconstrained case.

For the linear constraint $x_2 = 1$, the projection operation onto the line is:

$$\Pi_{\mathcal{C}}(\boldsymbol{y}) = \begin{pmatrix} y_1 \\ 1 \end{pmatrix}$$

Algorithm implementation:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{k+1} = \Pi_{\mathcal{C}}\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{k} - \alpha \begin{pmatrix} 2x_1 + x_2 \\ 2x_2 + x_1 \end{pmatrix}_{k} \right)$$

We demonstrate the algorithm starting from the same initial point as the unconstrained case:

We set the initial point as $x_0 = (1.0, 2.0)^{\mathrm{T}}$, which lies off the constraint. The algorithm first projects this point onto the constraint $x_2 = 1$, resulting in $(1.0, 1.0)$. After 1000 iterations, it converges to the constrained optimal point with function value 0.75.

The visualization below shows the optimization trajectory and the projection process. The black line represents the constraint $x_2 = 1$.

For all iterations, the visualization shows (with later iterations becoming more transparent):

- **Red arrows**: gradient steps $-\alpha \boldsymbol{\nabla} f(\boldsymbol{x}_k)$ from current point to unconstrained update
- **Orange dotted lines**: projection steps from the unconstrained update back to the constraint

This clearly demonstrates how the projected gradient method alternates between taking gradient steps and projecting back to the feasible set. The gradient arrows show both the direction and magnitude of the descent step, while the projection steps ensure feasibility. The path successfully reaches the constrained optimal point $(-0.5, 1.0)$.

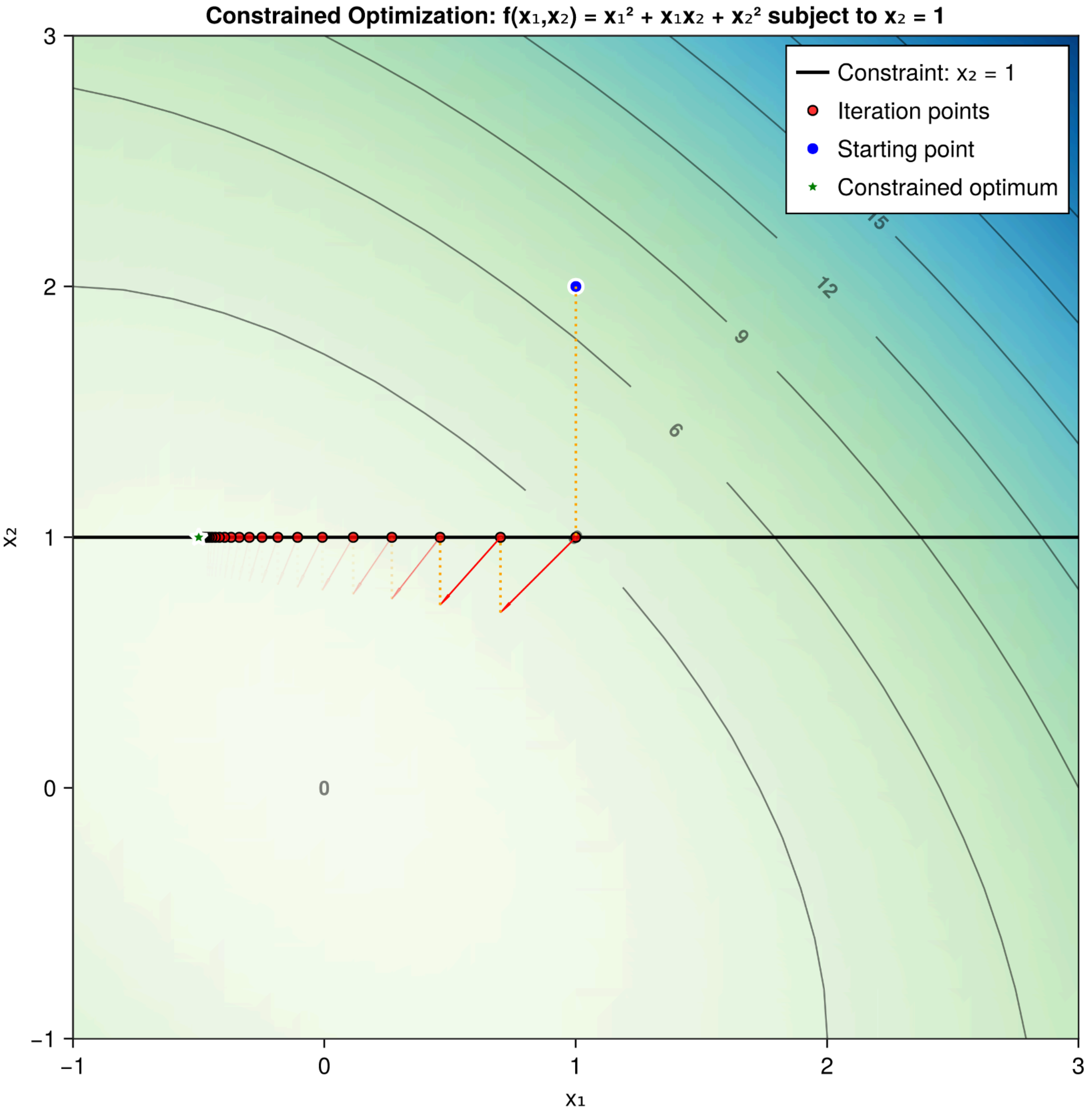

Figure 2: Constrained Gradient Descent Visualization

3.3 Penalty Method

Another approach for handling constraints is the penalty method, where we convert the constrained problem into an unconstrained one by adding penalty terms:

$$\min_{\boldsymbol{x}\in\mathbf{R}^n} L(\boldsymbol{x},\rho) = f(\boldsymbol{x}) + \rho\sum_{i=1}^{m}\max\big(0, g_{i(\boldsymbol{x})}\big)^2 + \rho\sum_{j=1}^{l} h_{j(\boldsymbol{x})}^2$$

where $\rho > 0$ is the penalty parameter. As $\rho \to \infty$, the solution of the penalized problem approaches the solution of the original constrained problem.

4 CONVERGENCE

Next, we rigorously analyze the convergence of gradient descent.

5 CONVEX OPTIMIZATION

If the optimization problem is convex, then gradient descent can guarantee finding the global minimum. The definition of a convex function is: for any $x, y \in \mathbf{R}^n$ and $\lambda \in [0, 1]$, we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$