

Optimization techniques in Machine Learning

Xuanye Qiu Xin Zhu Liwen Chen Anqi Yin Yunyi Li

June 16, 2025

- 1. Introduction**
- 2. Problem Statement**
- 3. Mathematical Analysis**
- 4. Simulation and Conclusion**

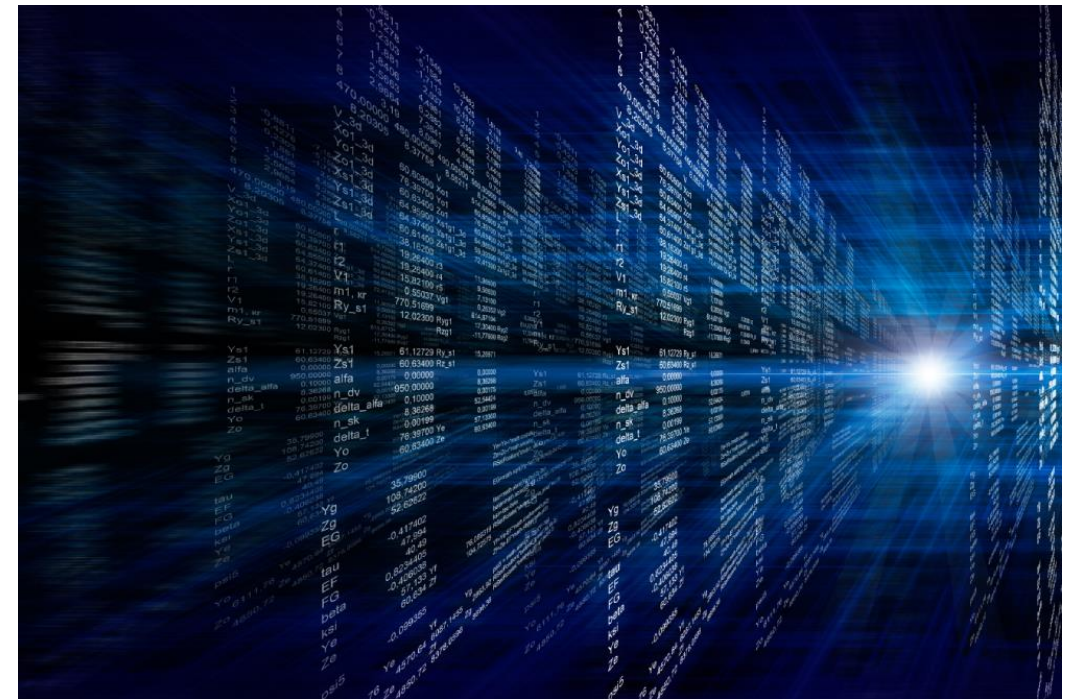
1. Introduction

Why is optimization important?



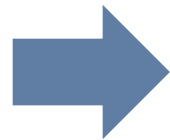
Why optimization methods must adapt?

- Explosive growth of data
- Rising complexity of machine learning models



2. Problem Statement

- Most ML task
 - Classification
 - Regression
 - Clustering
 - Reinforcement Learning
 - ...



**Optimization
problems**

- Objective function:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N L(y^i, f(x^i, \theta))$$

- Objective function with regularization:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N L(y^i, f(x^i, \theta)) + \lambda \| \theta \|_2^2$$

- First-Order Methods (Gradient-based)
 - Use gradient only
 - Examples: Gradient Descent, SGD, AdaGrad,...
 - Scalable and widely used
 - High-Order Methods (Curvature-based)
 - Use second or higher-order derivatives
 - Examples: Conjugate Gradient method, Quasi-Newton method, L-BFGS...
 - Faster convergence but higher computational cost
-

3. Mathematical Analysis

- Gradient Descent

$$L(\theta) = \frac{1}{2N} \sum_{i=1}^N (y^i - f_{\theta}(x^i))^2$$

$$f_{\theta}(x) = \sum_{j=1}^D \theta_j x_j$$

$$\frac{\partial L(\theta)}{\partial \theta_j} = -\frac{1}{N} \sum_{i=1}^N (y^i - f_{\theta}(x^i)) x_j^i$$

$$\theta'_j = \theta_j - \eta \nabla_{\theta} L(\theta_t)$$

- Stochastic Gradient Descent(SGD)

Loss Function $L(\theta)$

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (y^i - f_{\theta}(x^i))^2 = \frac{1}{N} \sum_{i=1}^N \text{cost}(\theta, (x^i, y^i))$$

Loss Function $L^*(\theta)$:

$$L^*(\theta) = \text{cost}(\theta, (x^i, y^i)) = \frac{1}{2} (y^i - f_{\theta}(x^i))^2$$

Update Rule :

$$\theta' = \theta + \eta (y^i - f_{\theta}(x^i)) x^i$$

- Nesterov Accelerated Gradient(NAG)

$$\tilde{\theta}_t = \theta_t + \beta v_t \rightarrow v_{t+1} = \beta v_t - \eta \nabla f(\tilde{\theta}_t) \rightarrow \theta_{t+1} = \theta_t + v_{t+1}$$
$$v = \eta \left(-\frac{\partial L(\theta)}{\partial(\theta)} \right) + v^{old} \beta$$

- Derivative-Free Technique

$$\theta_{t+1,j} = \arg \min L(\theta_{t+1,1}, \dots, \theta_{t+1,j-1}, \theta_j, \theta_{t,j+1}, \dots, \theta_{t,D})$$

- Adaptive Learning Rate Method(AdaGrad)

$$\begin{cases} g_t = \nabla_{\theta} L(\theta_t) \\ V_t = \sqrt{\sum_{i=1}^t (g_i)^2 + \epsilon} \\ \theta_{t+1} = \theta_t - \eta \frac{g_t}{V_t} \end{cases}$$

Second-Order Method

Conjugate Gradient Method

- optimization objective

$$\min_{\theta} F(\theta) = \frac{1}{2} \theta^T A \theta - b \theta + c$$

Core idea

Build a sequence of A-conjugate search directions

$$\{d_0, d_1, \dots, d_n\}: d_i^T A d_j = 0 \text{ for } i \neq j$$

At each iteration do an exact 1 – D line search along d_t

- Property: generating a new vector d_n only using the previous vector d_{t-1}

Algorithm Conjugate Gradient Method

Input: A, b, θ_0

Output: The solution θ^*

$$r_0 = A\theta_0 - b$$

$$d_0 = -r_0, t = 0$$

while Unsatisfied convergence condition **do**

$$\eta_t = \frac{r_t^\top r_t}{d_t^\top A d_t}$$

$$\theta_{t+1} = \theta_t + \eta_t d_t$$

$$r_{t+1} = r_t + \eta_t A d_t$$

$$\beta_{t+1} = \frac{r_{t+1}^\top r_{t+1}}{r_t^\top r_t}$$

$$d_{t+1} = -r_{t+1} + \beta_{t+1} d_t$$

$$t = t + 1$$

end while

Second-Order Method

Quasi-Newton Method

Core idea:

Use a positive definite matrix to approximate the inverse of the Hessian matrix

Quasi-Newton Condition:

Extend $f(\theta)$ to Talor series at $\theta = \theta_{t+1}$. Then compute the gradient on both sides of it, set $s_t = \theta_{t+1} - \theta_t$, and $u_t = \nabla f(\theta_t + 1) - \nabla f(\theta_t)$, we get:

$$u_t = B_{t+1} s_t$$

The search direction of quasi-Newton method is

$$d_t = -B_t^{-1} g_t$$

where g_t is the gradient of f , and the update of quasi-Newton Is

$$\theta_{t+1} = \theta_t + \eta_t d_t$$

Quasi-Newton method uses B_t to approximate the true Hessian matrix. We set $H_t = B_t^{-1}$:

DFP: (William C. Davidon, Fletcher and Powell)
The DFP correction formula is

$$B_{t+1}^{(DFP)} = (I - \frac{u_t s_t^\top}{u_t^\top s_t}) B_t (I - \frac{s_t u_t^\top}{u_t^\top s_t}) + \frac{u_t u_t^\top}{u_t^\top s_t}.$$

The update formula of H_{t+1} is

$$H_{t+1}^{DFP} = H_t - \frac{H_t u_t u_t^\top H_t}{u_t^\top H_t u_t} + \frac{s_t s_t^\top}{u_t^\top s_t}.$$

BFGS: (Broyden, Fletcher, Goldfarb and Shanno)
 B_{t+1} is updated according to

$$B_{t+1}^{(BFGS)} = B_t - \frac{B_t s_t s_t^\top B_t}{s_t^\top B_t s_t} + \frac{u_t u_t^\top}{u_t^\top s_t}.$$

corresponding update of H_{t+1} is

$$H_{t+1}^{(BFGS)} = (I - \frac{s_t u_t^\top}{s_t^\top u_t}) H_t (I - \frac{u_t s_t^\top}{s_t^\top u_t}) + \frac{u_t s_t^\top}{s_t^\top u_t}.$$

Second-Order Method

Stochastic Quasi-Newton Method

- Consider $\min_{\theta \in R} F(\theta) = \mathbb{E}[f(\theta, \xi)]$, where ξ is a random seed, f usually has the following form:

$$f(\theta, \xi) = f(\theta; x_i, y_i) = l(h(w; x_i); y_i)$$

use the empirical loss to define the objective:

$$F(\theta) = \frac{1}{N} \sum_{i=1}^N f_i(\theta)$$

- use mini batch stochastic gradient

$$\nabla F_{S_t}(\theta_t) = \frac{1}{c} \sum_{i \in S_t} \nabla f_i(\theta_t)$$

where S_t is randomly selected.

- Let S_t^H be a randomly chosen subset of the training samples, the stochastic Hessian estimate is

$$\nabla^2 F_{S_t}(\theta_t) = \frac{1}{c_h} \sum_{i \in S_t^H} \nabla^2 f_i(\theta_t)$$

Algorithm SQN Framework

Input: θ_0, V, m, η_t

Output: The solution θ^*

for $t=1, 2, 3, 4, \dots$, **do**

$s'_t = H_t g_t$ using the two-loop recursion.

$s_t = -\eta_t s'_t$

$\theta_{t+1} = \theta_t + s'_t$

if update pairs **then**

 Compute s_t and u_t

 Add a new displacement pair $\{s_t, u_t\}$ to V

if $|V| > m$ **then**

 Remove the eldest pair from V

end if

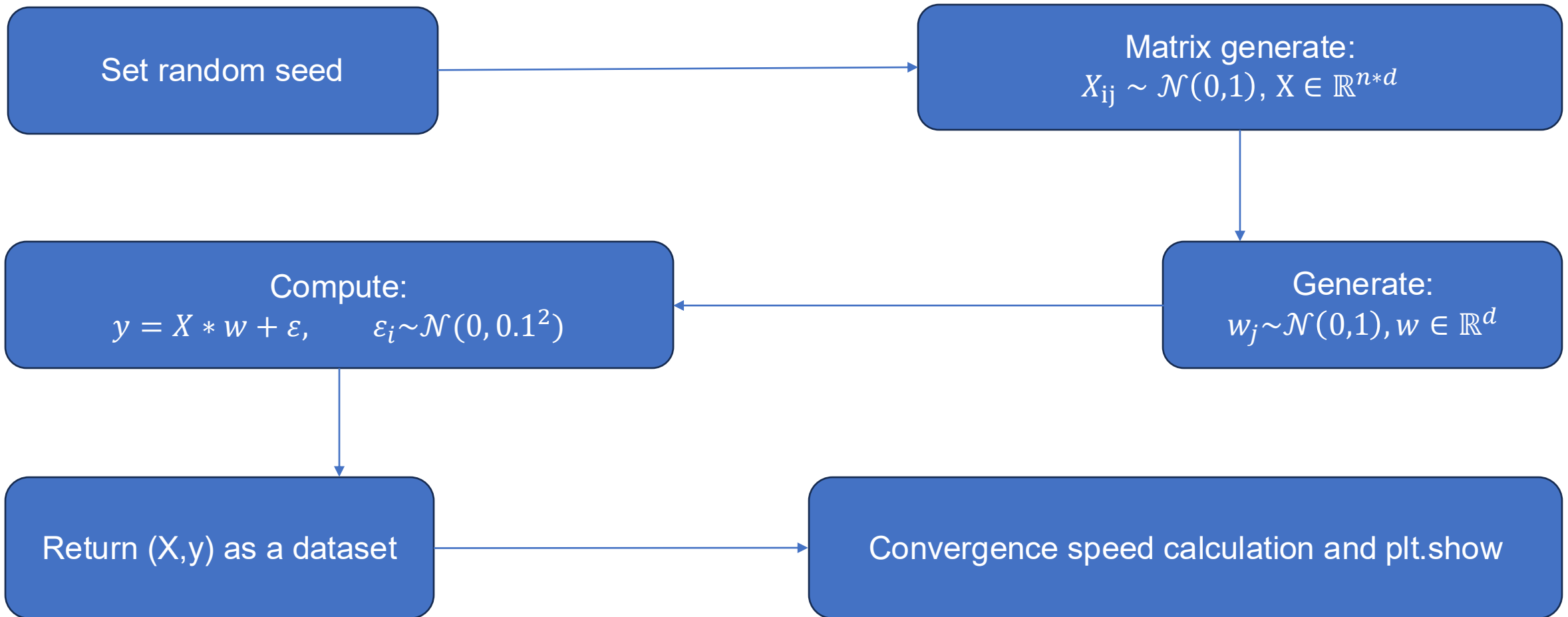
end if

end for

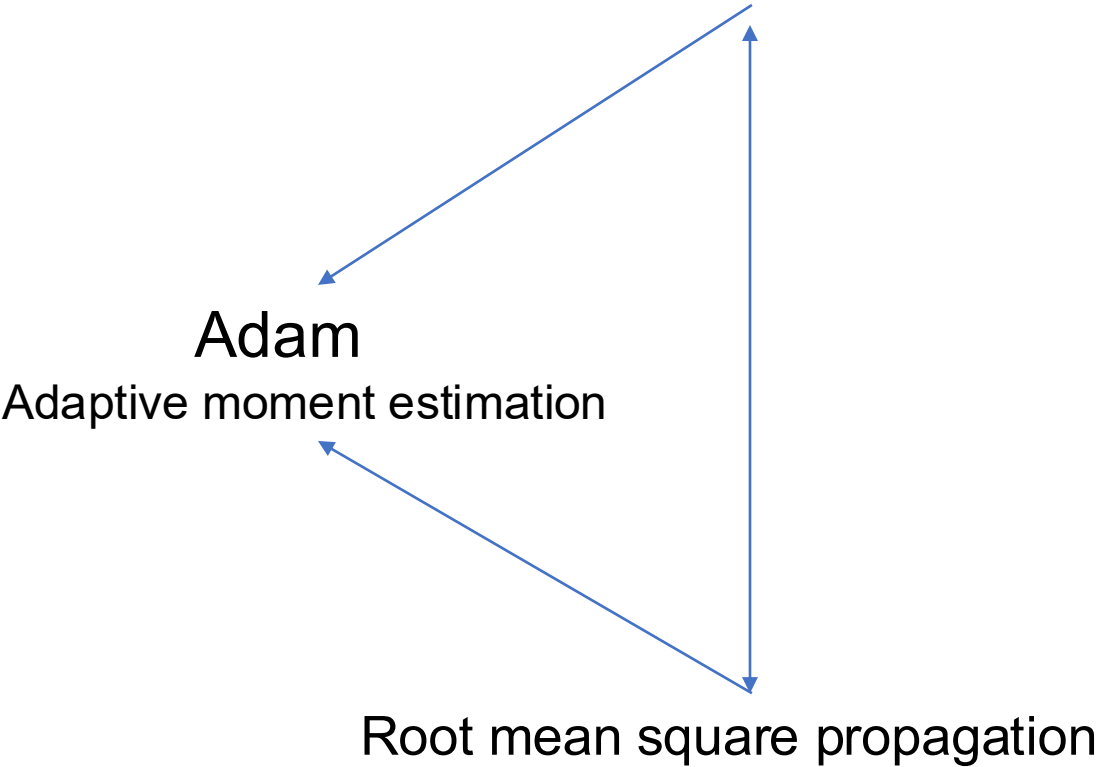
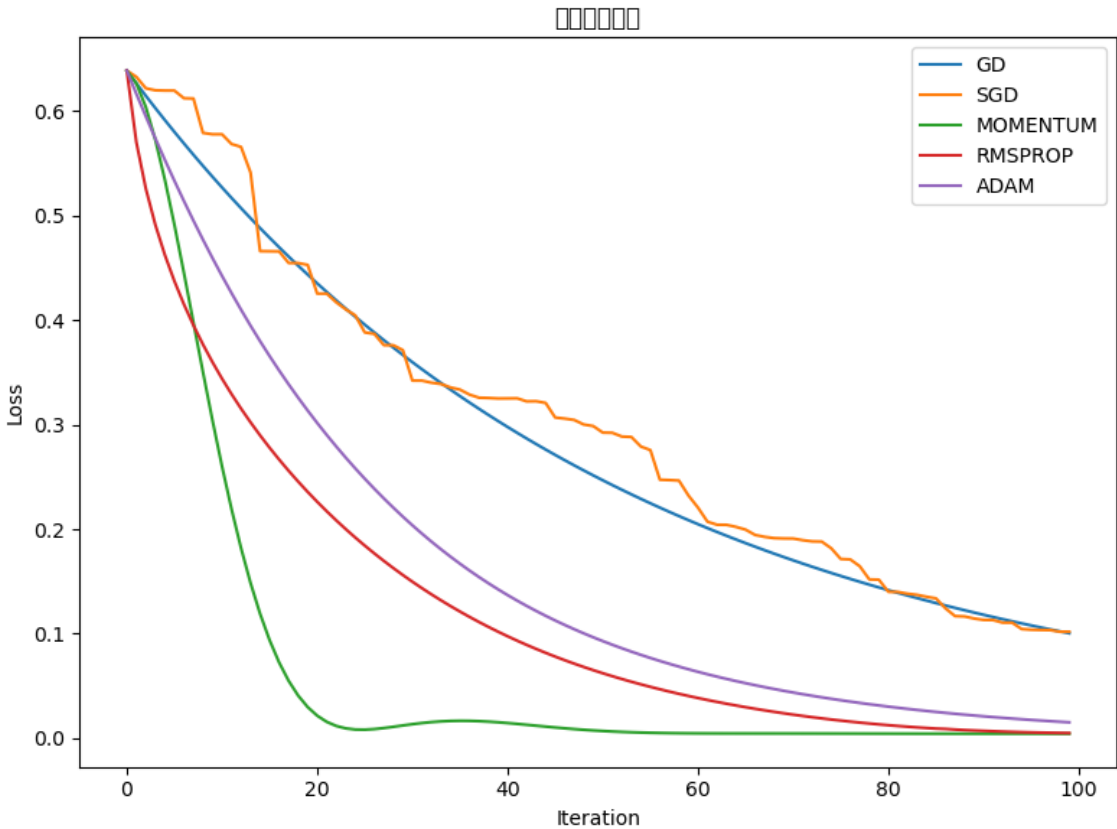
4. Simulation and Conclusion

Simulation and Conclusion

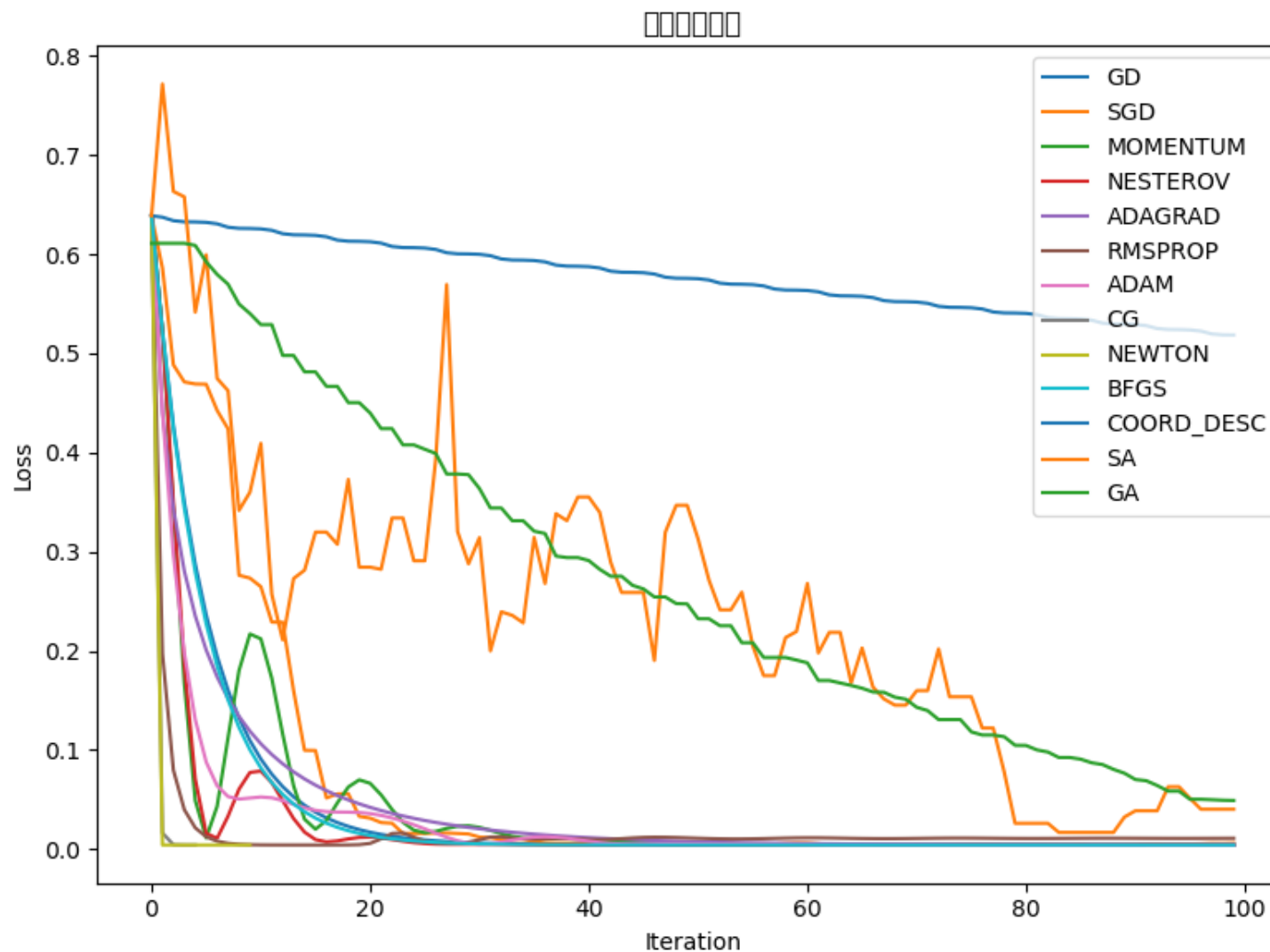
Dataset description and convergence speed in different algorithm



Gradient descent → Stochastic Gradient descent → Momentum with SGD



Simulation and Conclusion



Mark:
Lr = 0.01,
Iters = 100

Thank you!