梯度下降法的缺陷及其改进

数学原理和代码实现

小胖

目录

随机梯度下降法

梯度下降法的效率问题

损失函数等于每点损失之和

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - ax_i - b)^2 = \frac{1}{n}\sum_i L_i$$

由于损失函数的梯度等于各点梯度的平均值：

$$a_{k+1} = a_k - \gamma\frac{\partial L(a_k, b_k)}{\partial a}, b_{k+1} = b_k - \gamma\frac{\partial L(a_k, b_k)}{\partial b}$$

- 计算开销大
- 面对大量训练数据时，几乎不可用

每次计算梯度都需要加总所有点的梯度

$$\frac{\partial L}{\partial a} = \frac{1}{n}\sum_i \frac{\partial L_i}{\partial a} \qquad \frac{\partial L}{\partial b} = \frac{1}{n}\sum_i \frac{\partial L_i}{\partial b}$$

随机梯度下降法

SGD、Mini-Batch Gradient Descent

为了提升效率，使用小批量的数据的梯度平均值代替损失函数的梯度：

- 在实现时，引入epoch和batch_size两个超参数
- 极限情况下，使用一个数据点的梯度代替损失函数的梯度

用小批量（mini batch）的梯度平均值代替全局梯度平均值

$$\frac{\partial L}{\partial a} = \frac{1}{n}\sum_i \frac{\partial L_i}{\partial a} \approx \frac{1}{m}\sum_{i=0}^{m} \frac{\partial L_i}{\partial a} \qquad \frac{\partial L}{\partial b} = \frac{1}{n}\sum_i \frac{\partial L_i}{\partial b} \approx \frac{1}{m}\sum_{i=0}^{m} \frac{\partial L_i}{\partial b}$$

迭代公式变为

$$a_{k+1} = a_k - \gamma \frac{1}{m}\sum_{i=0}^{m} \frac{\partial L_i(a_k, b_k)}{\partial a}, \quad b_{k+1} = b_k - \gamma \frac{1}{m}\sum_{i=0}^{m} \frac{\partial L_i(a_k, b_k)}{\partial b}$$

$$a_{k+1} = a_k - \gamma \frac{\partial L(a_k, b_k)}{\partial a}$$

两个超参数：batch_size（公式中的m）、epoch（所有数据循环的上限）

目录

随机梯度下降法的改进

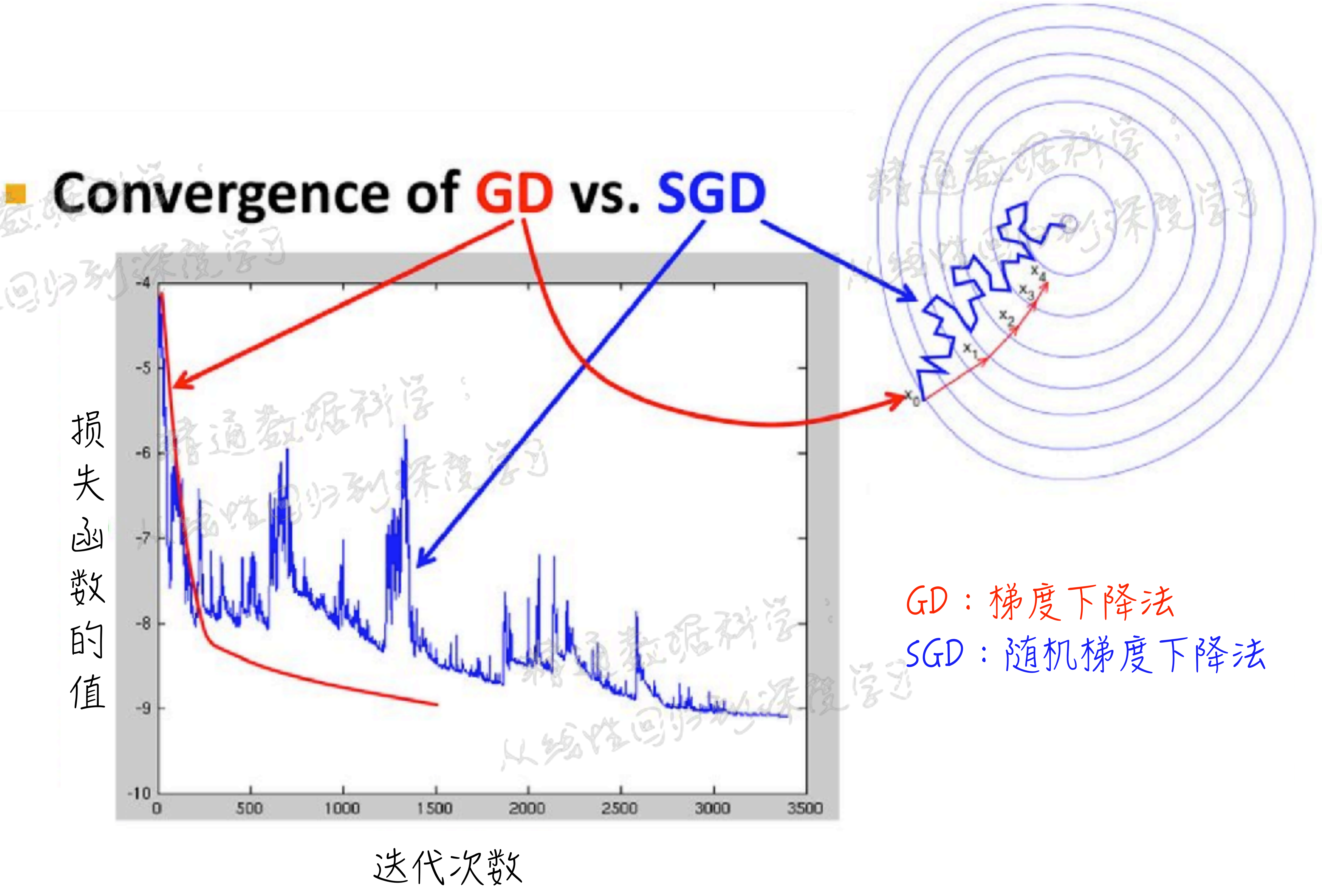
随机梯度下降法面对的主要困难

梯度下降法 vs 随机梯度下降法

随机梯度下降法的梯度估计并不稳定，下降路径时常“弯弯曲曲”

随机梯度下降法中，函数梯度的估计并不稳定

- 遇到鞍点（saddle point）和山谷（Rosenbrock's valley）时，随机梯度下降法的效果很差

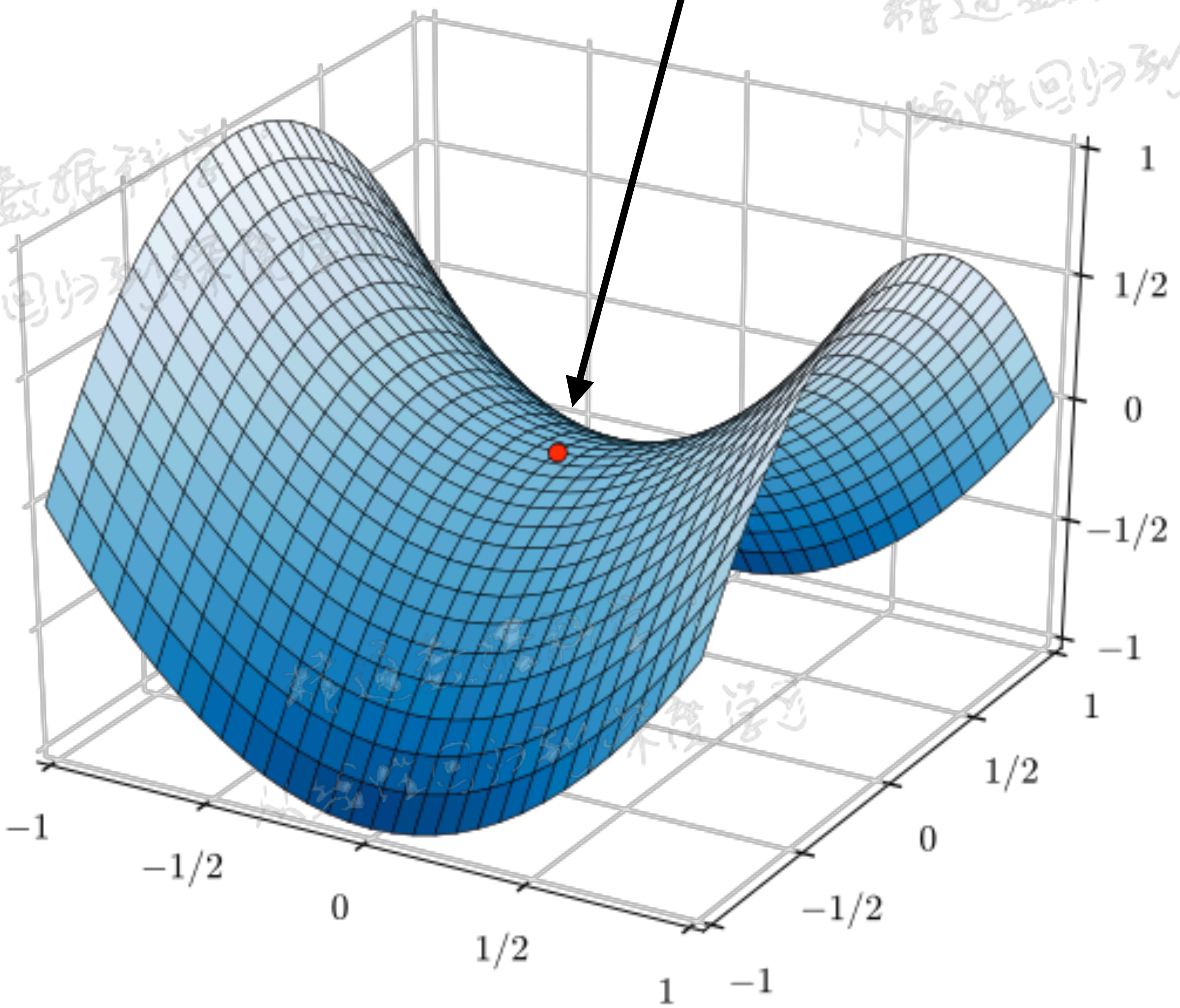


随机梯度下降法的改进

随机梯度下降法面对的主要困难

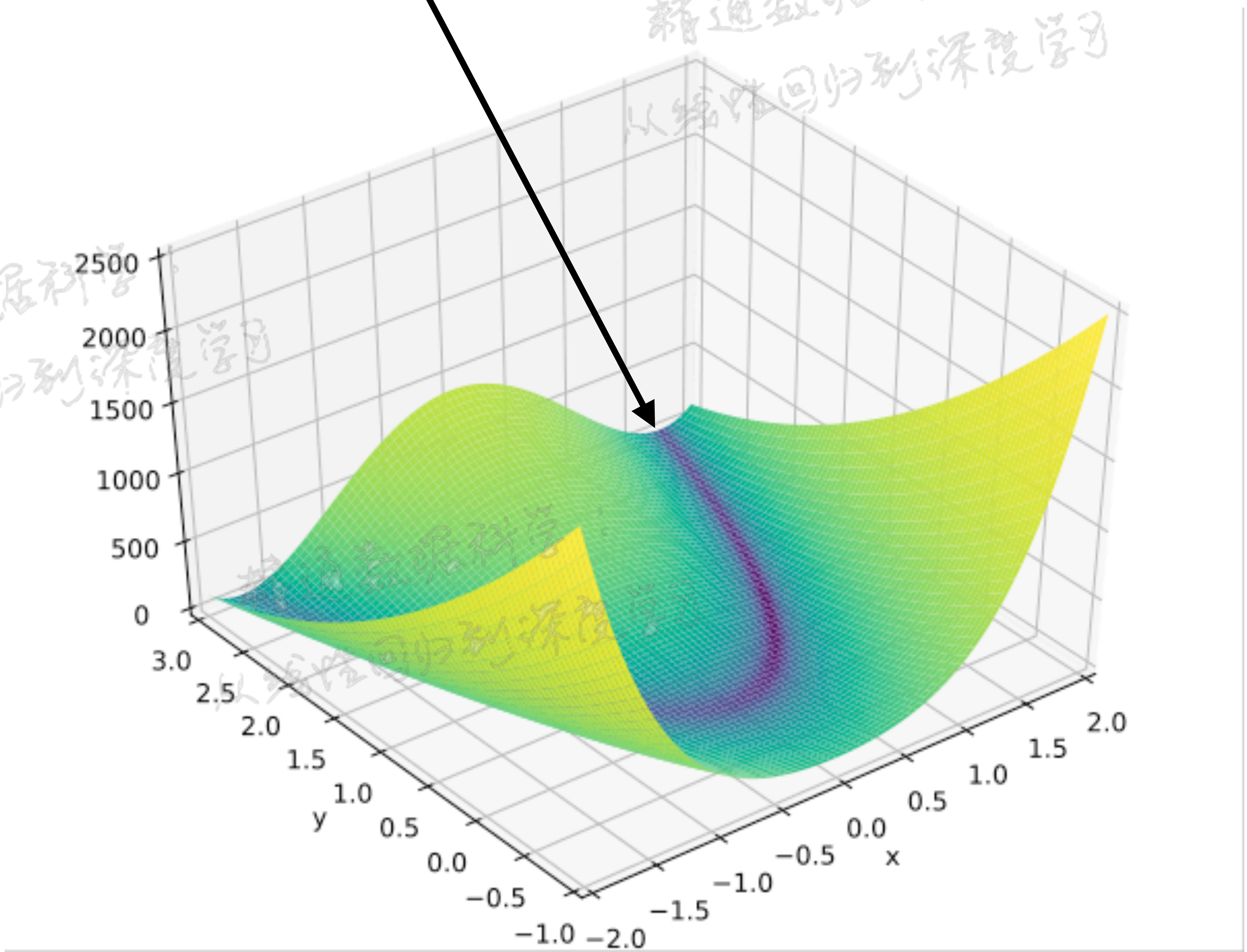
鞍点（saddle point）

梯度等于0，但并不是极小值点



山谷（Rosenbrock's valley）

在一个狭长的区域内梯度几乎等于0，
但两边的梯度绝对值很大



随机梯度下降法的改进

动量方法

随机梯度下降法

$$a_{k+1} = a_k - \gamma \frac{1}{m}\sum_{i=0}^{m}\frac{\partial L_i(a_k, b_k)}{\partial a}, \quad b_{k+1} = b_k - \gamma \frac{1}{m}\sum_{i=0}^{m}\frac{\partial L_i(a_k, b_k)}{\partial b}$$

动量方法（momentum）模拟物理中的惯性作用：

- 上一次迭代的改变量视为速度
- 此次迭代的梯度视为加速度

动量方法 （对参数b的迭代公式类似）

$$g_k = \frac{1}{m}\sum_{i=0}^{m}\frac{\partial L_i(a_k, b_k)}{\partial a}$$

$$m_k = \eta m_{k-1} + g_k, \quad a_{k+1} = a_k - \gamma m_k$$

衰减系数 momentum

上一次迭代的改变量

这次迭代的梯度

学习速率

随机梯度下降法的改进

Adam方法

动量方法 （对参数b的迭代公式类似）

$$g_k = \frac{1}{m}\sum_{i=0}^{m}\frac{\partial L_i(a_k, b_k)}{\partial a} \qquad m_k = \eta m_{k-1} + g_k, \quad a_{k+1} = a_k - \gamma m_k$$

Adam（Adaptive Moment Estimation）方法保留了“惯性作用”，并加入了如下作用：

- 对于历史梯度较小的模型参数，放大当前的更新步伐；反之则变小

Adam方法

$$m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$$

$$v_k = \beta_2 v_{k-1} + (1-\beta_2) g_k^2$$

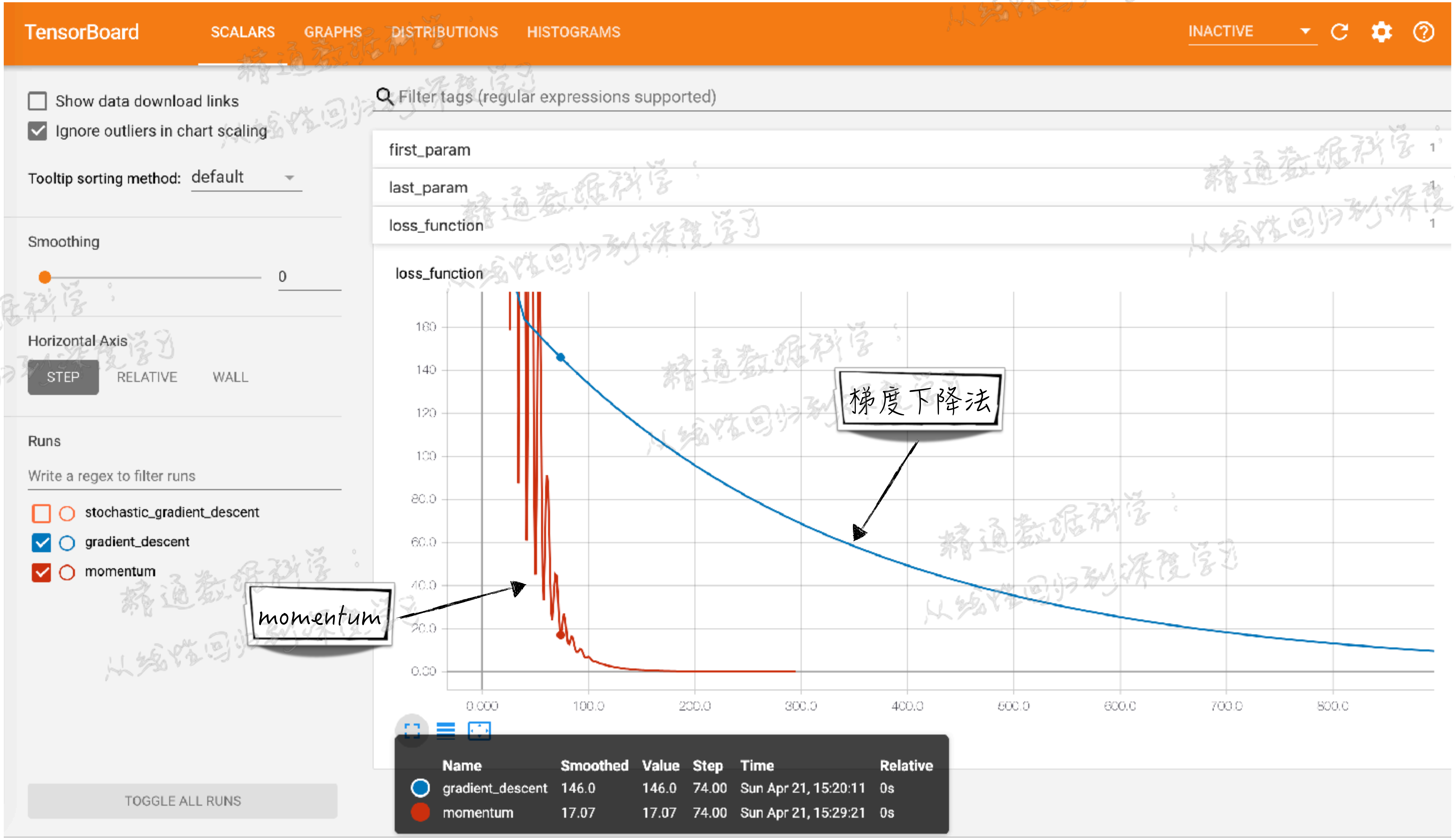
$$a_{k+1} = a_k - \gamma\frac{\hat{m}_k}{\sqrt{\hat{v}_k + \varepsilon}}$$

$$E[m_k] = (1-\beta_1^k)E[g_k] \quad E[v_k] = (1-\beta_2^k)E[g_k^2] \qquad \hat{m}_k = \frac{m_k}{1-\beta_1^k} \quad \hat{v}_k = \frac{v_k}{1-\beta_2^k}$$

目录

代码实现

下降细节

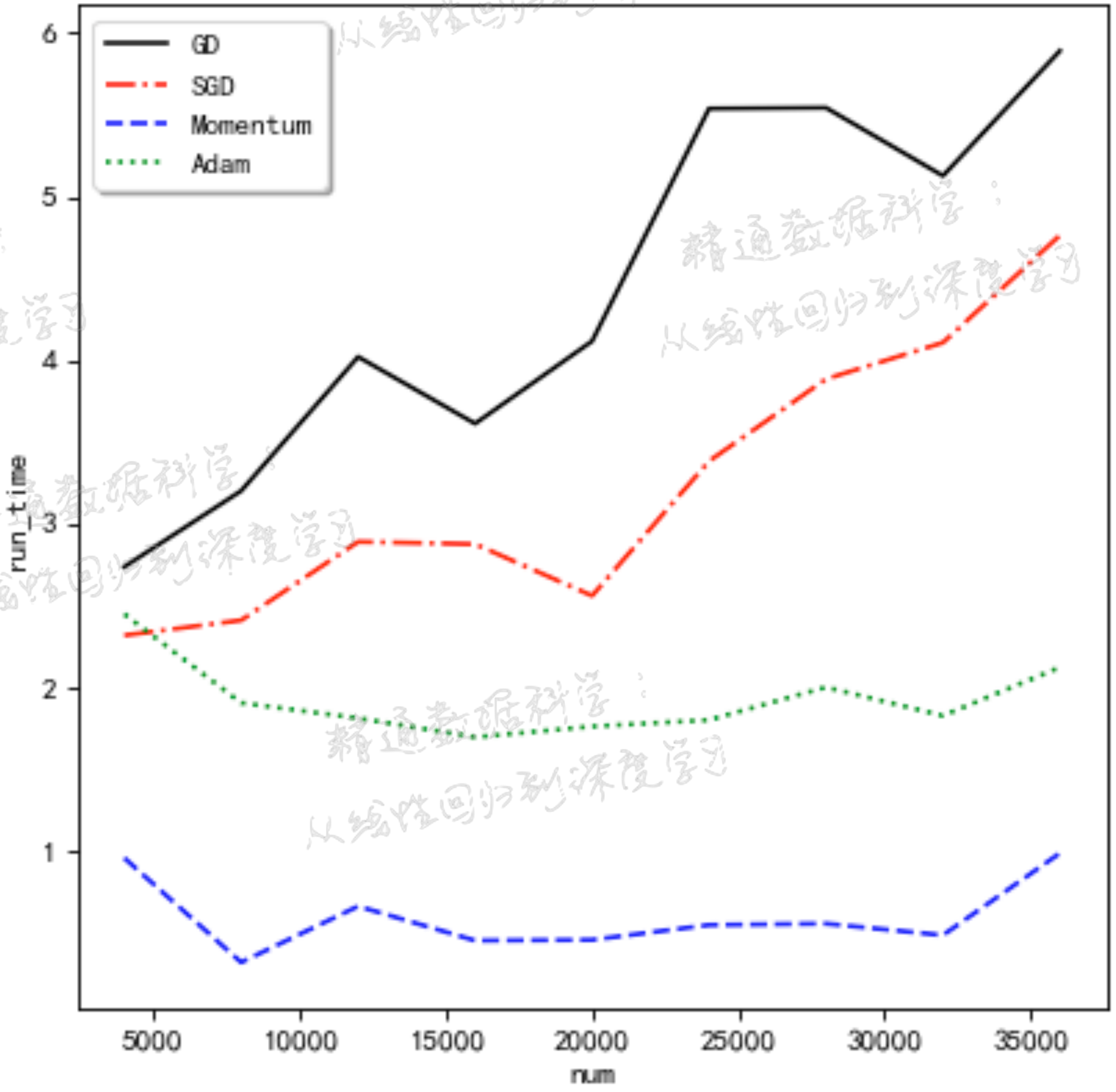


TensorBoard
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
stochastic_gradient_descent
gradient_descent
momentum
TOGGLE ALL RUNS
Filter tags (regular expressions supported)
first_param
last_param
loss_function
loss_function
梯度下降法
momentum
Name Smoothed Value Step Time Relative
gradient_descent 146.0 146.0 74.00 Sun Apr 21, 15:20:11 0s
momentum 17.07 17.07 74.00 Sun Apr 21, 15:29:21 0s

代码实现

算法比较

四种不同最优化算法的效率比较



THANK YOU