上海交通大学 SHANGHAI JIAO TONG UNIVERSITY 学生创新中心 Student Innovation Center

深度学习优化器

学生创新中心：肖雄子彦

SJTU

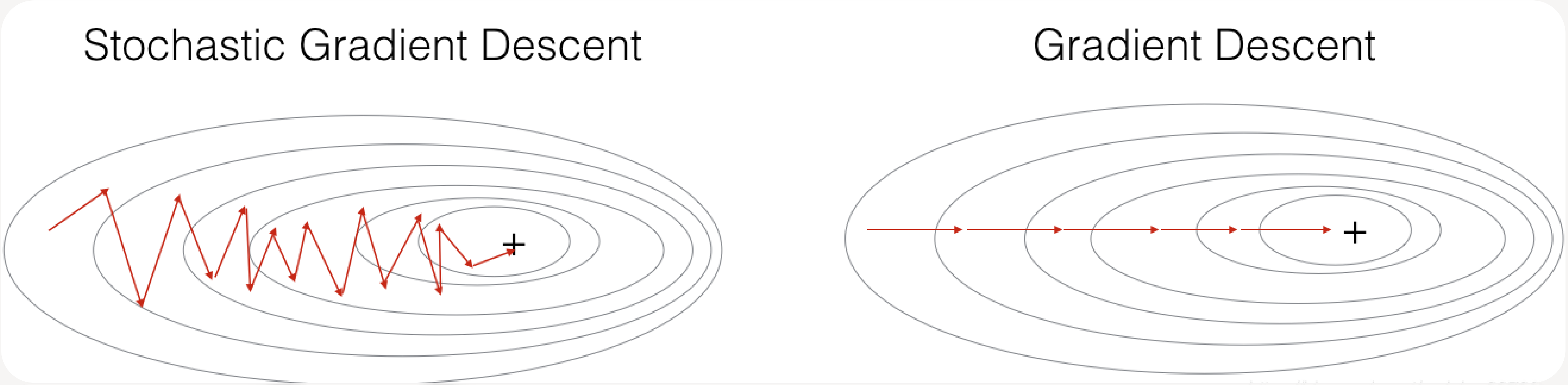
目录 / CONTENTS

01

SGD 随机梯度下降
MBGD 小批量梯度下降

SGD vs BGD

随机梯度下降（SGD） 批量梯度下降（BGD）



小批量梯度下降 Mini-Batch GD

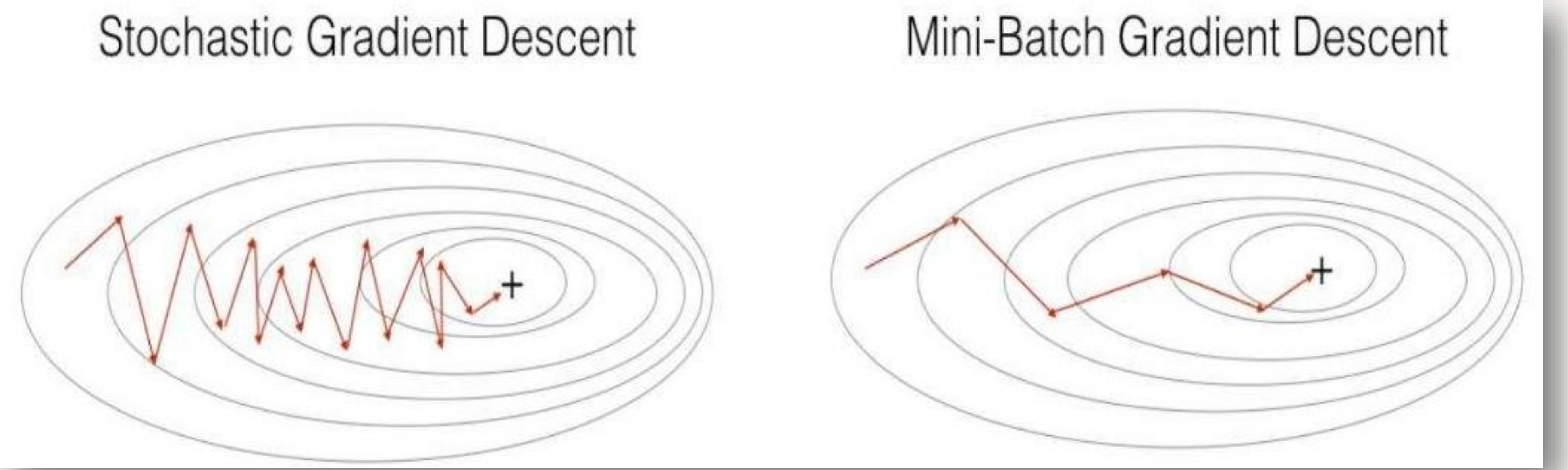
小批量梯度下降（Mini-Batch Gradient Descent, MBGD）

结合了BGD和SGD的优点，迭代速度比BGD快，精度比SGD高。

•每次使用一个batch，批量运算，迭代次数介于BGD和SGD之间

•通过矩阵加速运算，在一个batch上优化神经网络参数速度快。

•方向更准，收敛结果更加接近全批量梯度下降的效果。



02

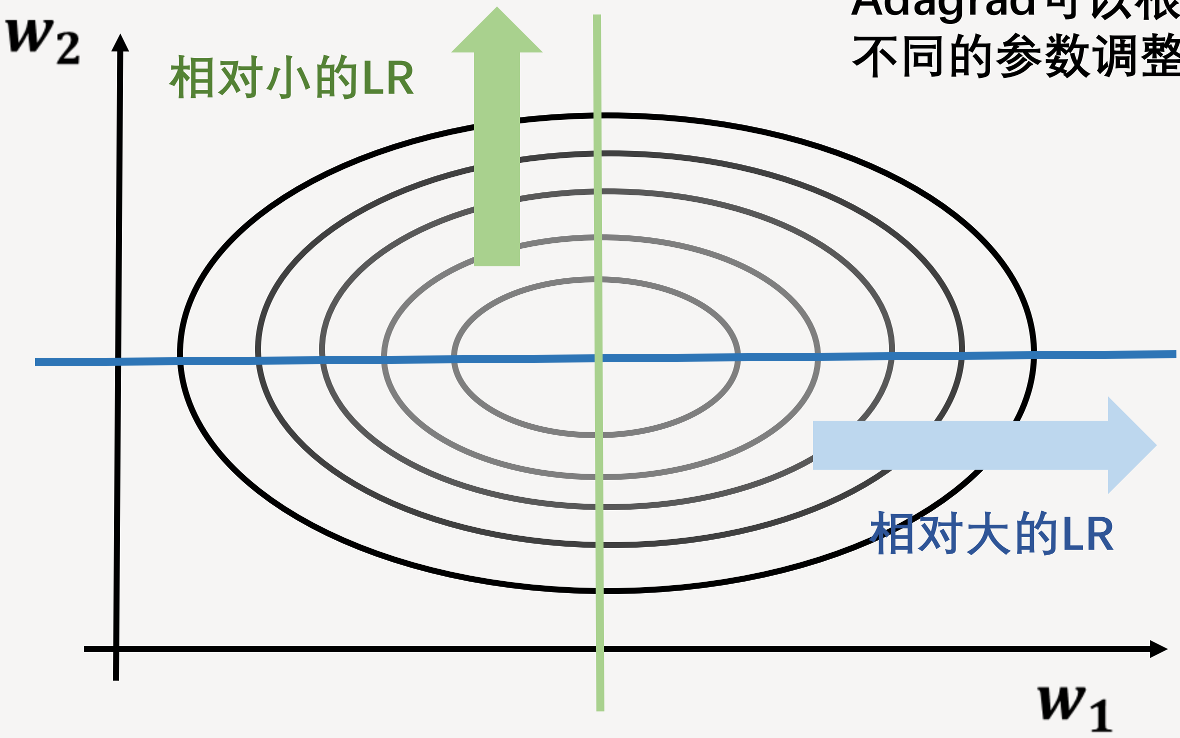
AdaGrad

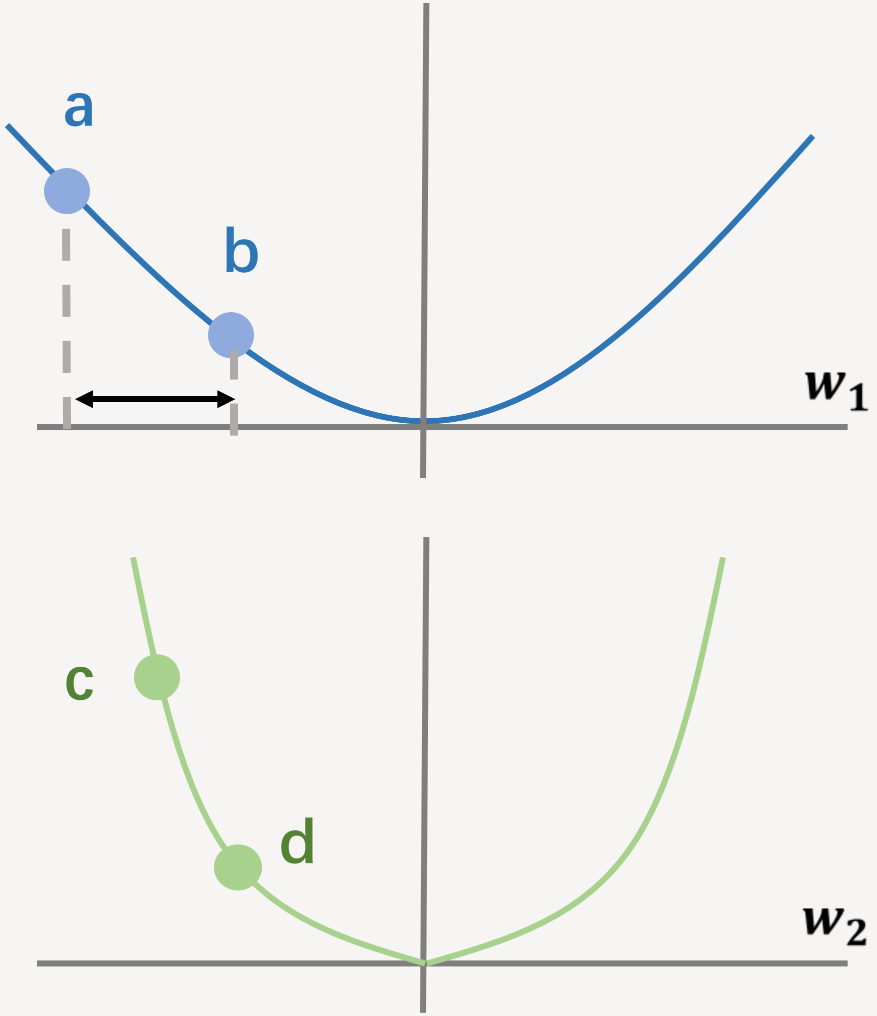
不同"坡度"使用不同的学习率

多个不同的参数情况

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

Adagrad可以根据
不同的参数调整LR





Adaptive gradient

Adagrad

learning rate

gradient

$$\sigma^0 = \sqrt{(g^0)^2} \qquad w^1 \leftarrow w^0 - \frac{\boxed{\eta^0}}{\boxed{\sigma^0}}\boxed{g^0}$$

$$\sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]} \qquad w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1$$

$$\sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]} \qquad w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2$$

$$\vdots$$

$$\sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2} \qquad w^{t+1} \leftarrow w^t - \frac{\boxed{\eta^t}}{\sigma^t} g^t$$

$$\eta^t = \frac{\eta}{\sqrt{t+1}}$$

随着更新次数增多
采用更小的learning rate

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

- 梯度频繁更新，累积的分母项逐渐变大，更新的步长相对就会变小
- 梯度比较稀疏，累积的分母项比较小，那么更新的步长则相对较大。

AdaGrad能够自动为不同参数适应不同的学习率

03 Momentum

Momentum

累积过去的梯度来决定当前更新方向

平滑梯度更新：减少梯度方向的震荡
给梯度加一点“惯性”，考虑最近N个梯度的历史值。

梯度

$$v_0 = 0$$

$$v_1 = \beta v_0 + (1-\beta)\boxed{x_1}$$

$$v_2 = \beta v_1 + (1-\beta)x_2$$

……

如果 $\beta = 0.9$ 经验值

$$v_1 = 0.9v_0 + 0.1x_1$$

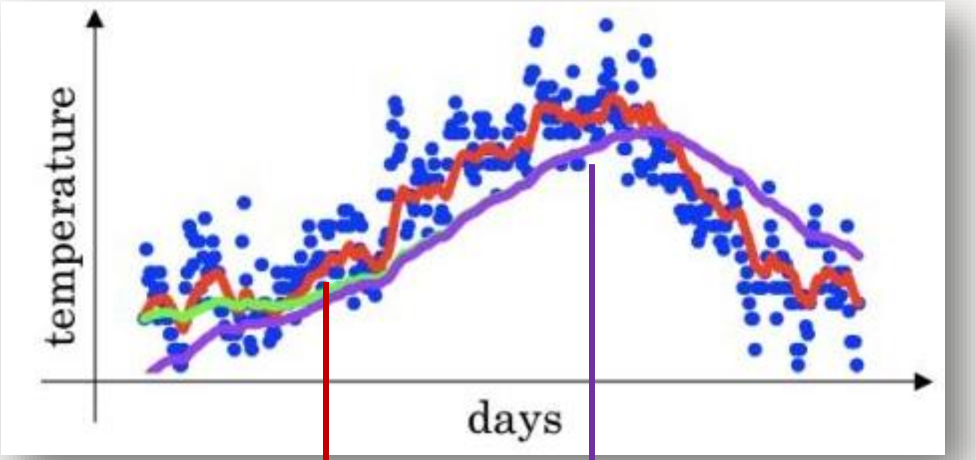
$$v_2 = 0.9\boxed{v_1} + 0.1x_2 \quad = 0.9^2 v_0 + 0.9{*}0.1x_1 + 0.1x_2$$

$$v_3 = 0.9\boxed{v_2} + 0.1x_3$$

到底和过去多少个观测值相关?

$$= 0.9^3 v_0 + 0.9^2 * 0.1x_1 + 0.9{*}0.1x_2 + 0.1x_3$$

N的大小是由β的大小来决定的



如果 $\beta = 0.9$
那么 $N = 10$

如果 $\beta = 0.98$
那么 $N = 50$

$$\mathbf{N} \approx \frac{\mathbf{1}}{\mathbf{1}-\boldsymbol{\beta}}$$

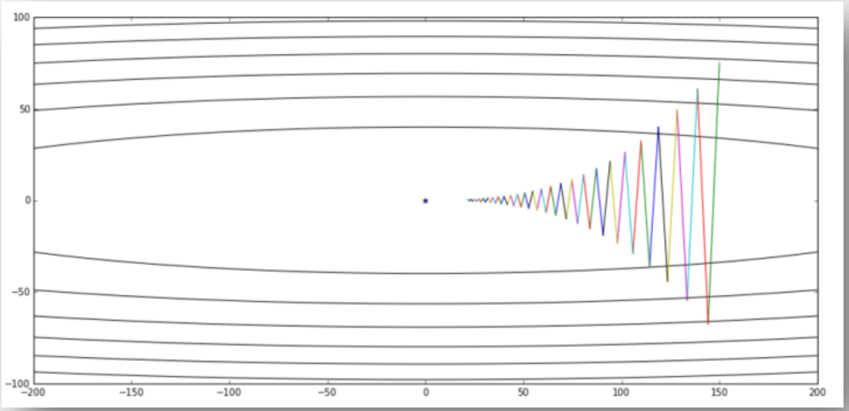
Momentum

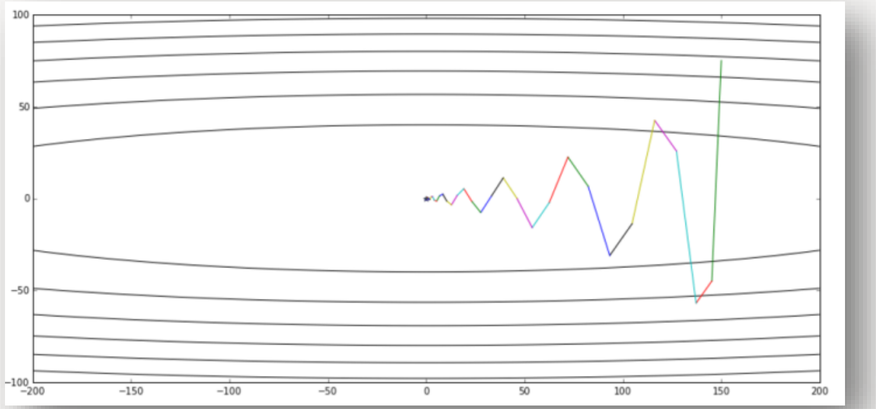
在梯度下降中

当learning rate比较小，担心收敛太慢；当learning rate比较大，担心跳过最优点或产生震荡。Momentum能一定程度解决这个问题，通俗来说，它的作用就是“修正急转弯”，更缓和，也少走一些“弯路”。

$v_t = \beta v_{t-1} + (1-\beta) g_t$ → SGD with momentum，在SGD基础上引入了一阶动量

$W := W - \alpha v_t$ 一阶动量是各个时刻梯度方向的指数移动平均值

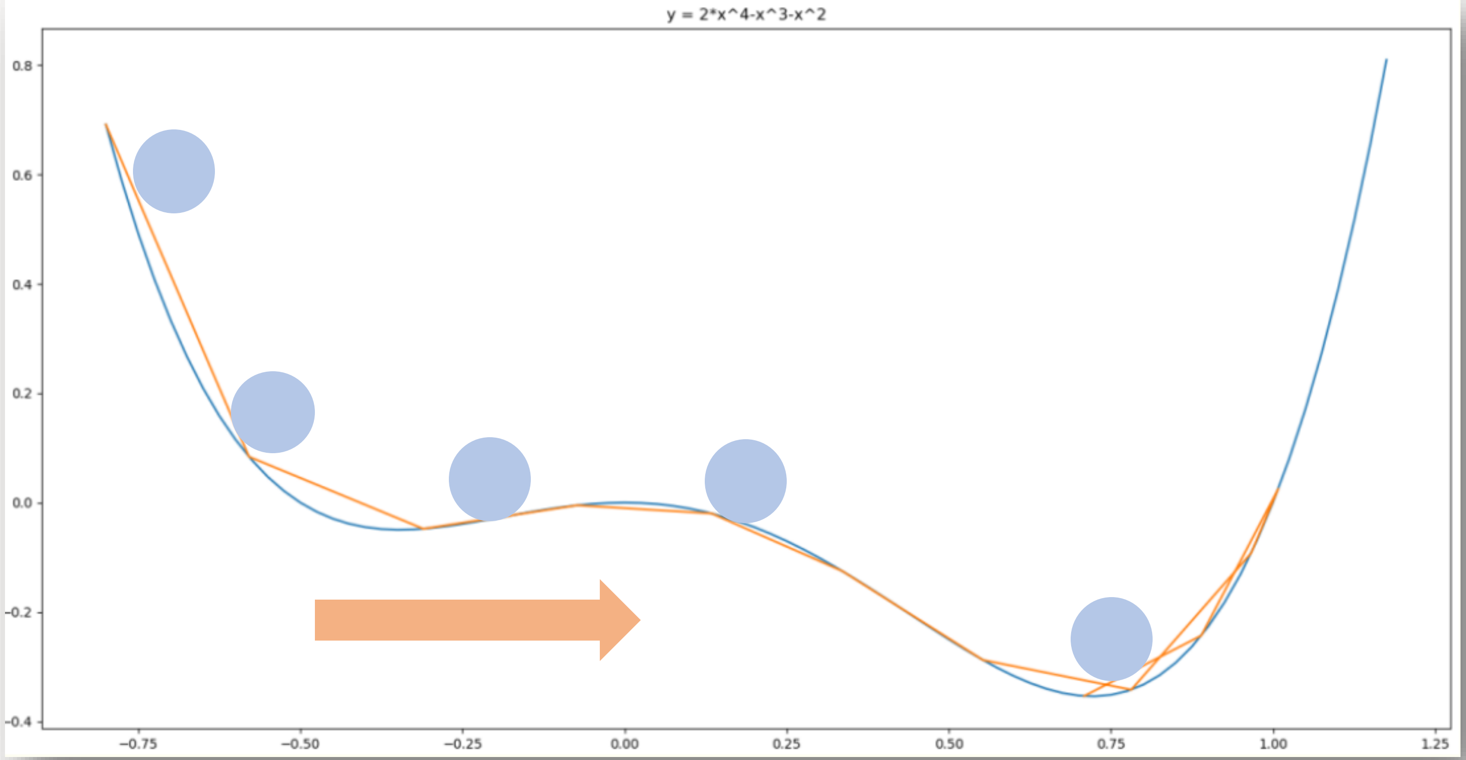




t时刻的梯度，不仅由当前梯度决定，也考虑了此前累积的N个梯度方向。

Momentum

Momentum的实际表现，就是不会停在一些“平原”或“小坑”。
因为它有之前累计的动量，就会向前冲。
（当然如果面对较大的坡，也可能因动量不足，爬不过去停在谷底）



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

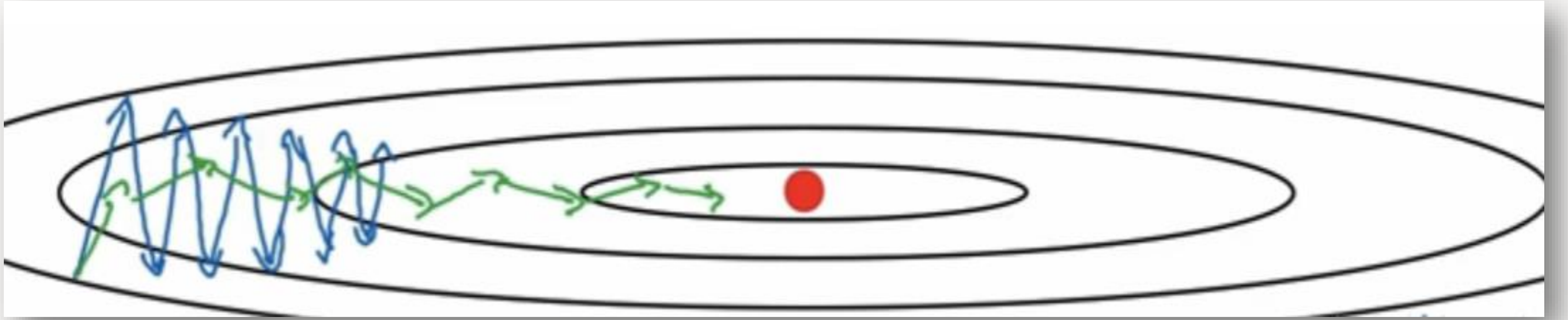
$$W := W - \alpha v_t$$

一阶动量

04 RMSProp

Root Mean Square Prop

RMSProp是Hinton提出的一种优化算法，进一步优化在更新迭代中摆动幅度过大的问题，来加快网络的收敛。



图中蓝色为经过Momentum优化后迭代的路线，绿色为经过RMSProp优化迭代的路线

RMSProp对梯度使用了微分平方加权平均

以W为例：

二阶动量

$$S_t = \beta S_{t-1} + (1-\beta)\, {g_t}^2$$

$$W := W - \alpha \frac{g_t}{\sqrt{S_t + \varepsilon}}$$

开根号可以使较大的梯度大幅度变小，而较小的梯度小幅度变小，从而减少波动。于是可设置相对较大的learning rate来加快学习速度。

05

Adam
自适应矩估计

Adaptive Moment Estimation

Adam（Adaptive Moment Estimation）

其实就是将Momentum算法和RMSProp算法结合起来。

- 在训练的最开始我们需要初始化梯度的累积量和平方累积量
- 在训练的第 t轮训练中，我们计算得到Momentum和RMSProp的参数更新

一阶动量 Momentum

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^{\,t}} \qquad \beta_1 = 0.9$$

二阶动量 RMSProp

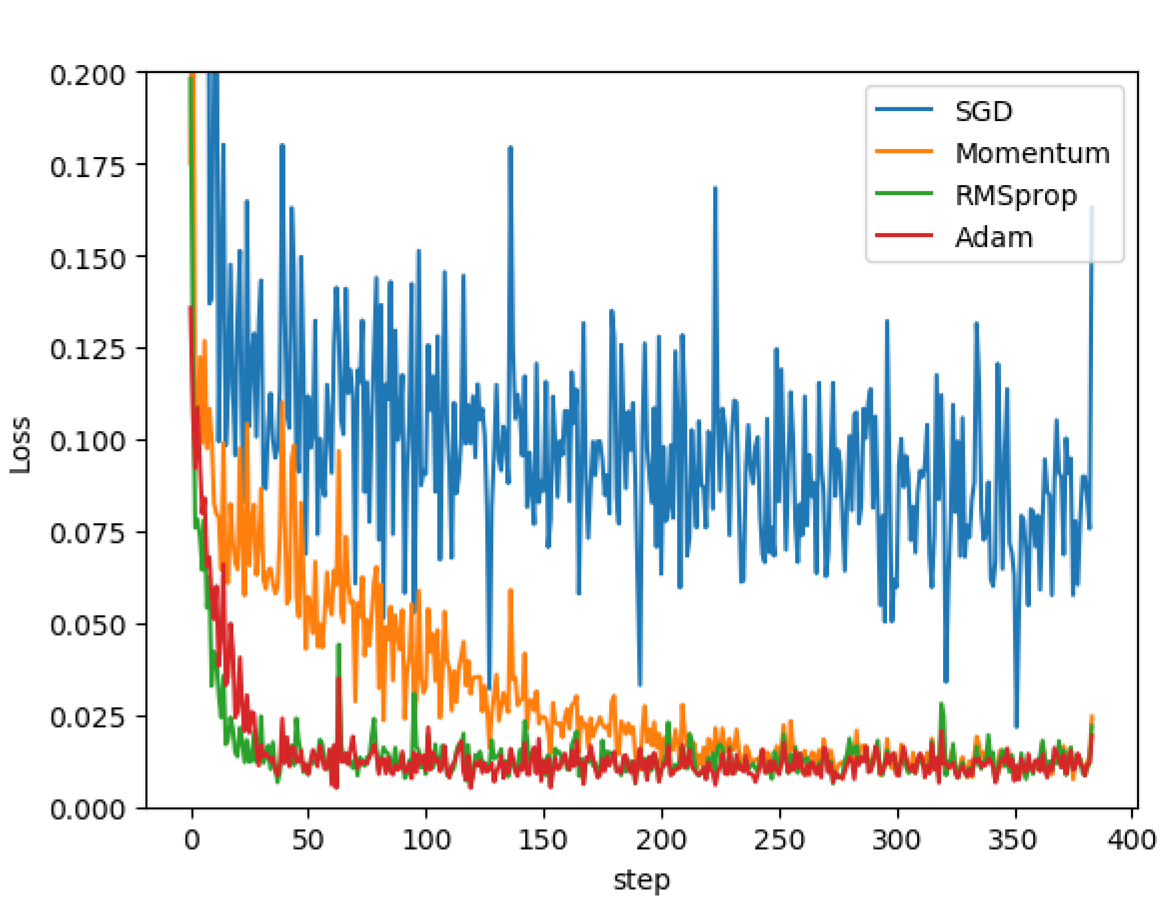
$$S_t = \beta_2 S_{t-1} + (1-\beta_2)\, g_t^{\,2}$$

$$\hat{S}_t = \frac{S_t}{1-\beta_2^{\,t}} \qquad \beta_2 = 0.999$$

移动指数平均在迭代初期会和开始值有较大差异，于是我们对上面求得的值做偏差修正。

$$W := W - \alpha \frac{\hat{m}_t}{\sqrt{\hat{S}_t} + \varepsilon}$$

4种常用优化器比较



- SGD 表现最普通, 波动依然比较大。
- Momentum加入一阶动量，波动缓和，迭代效率有所提升。
- RMSProp在Momentum基础上加入二阶动量，表现更好。
- Adam综合了Momentum和RMSProp，效果和RMSProp不分上下。

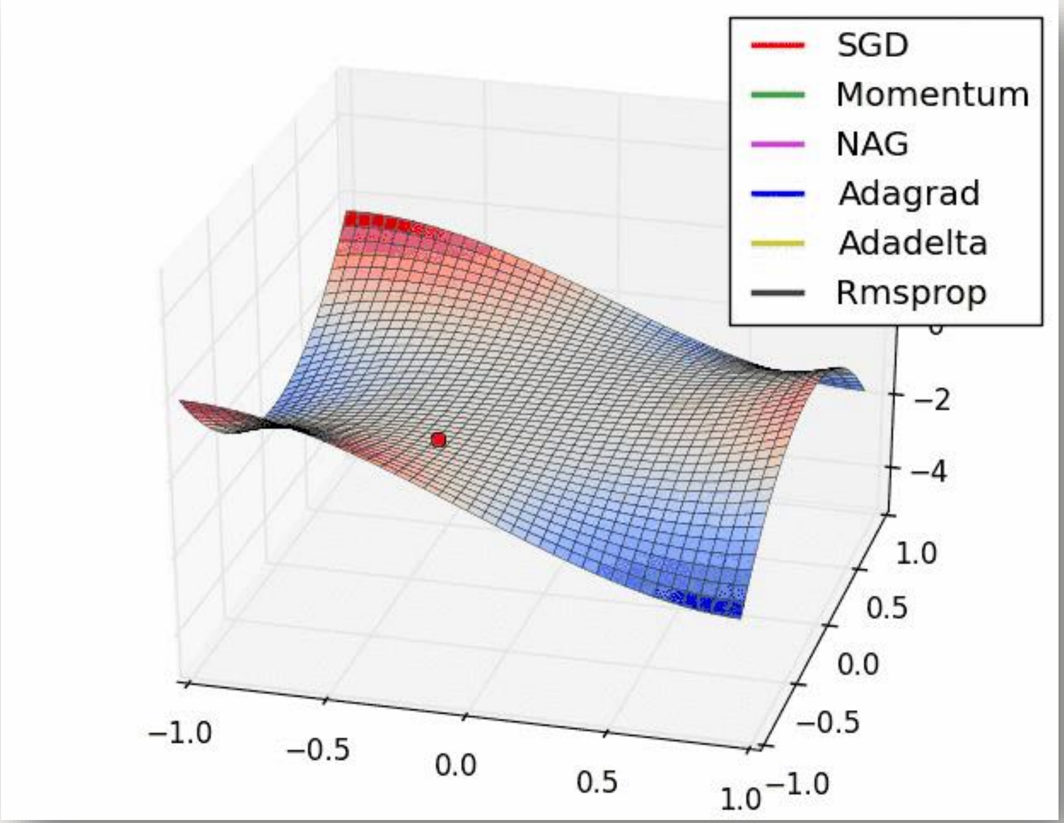
其他优化器

Adadelta

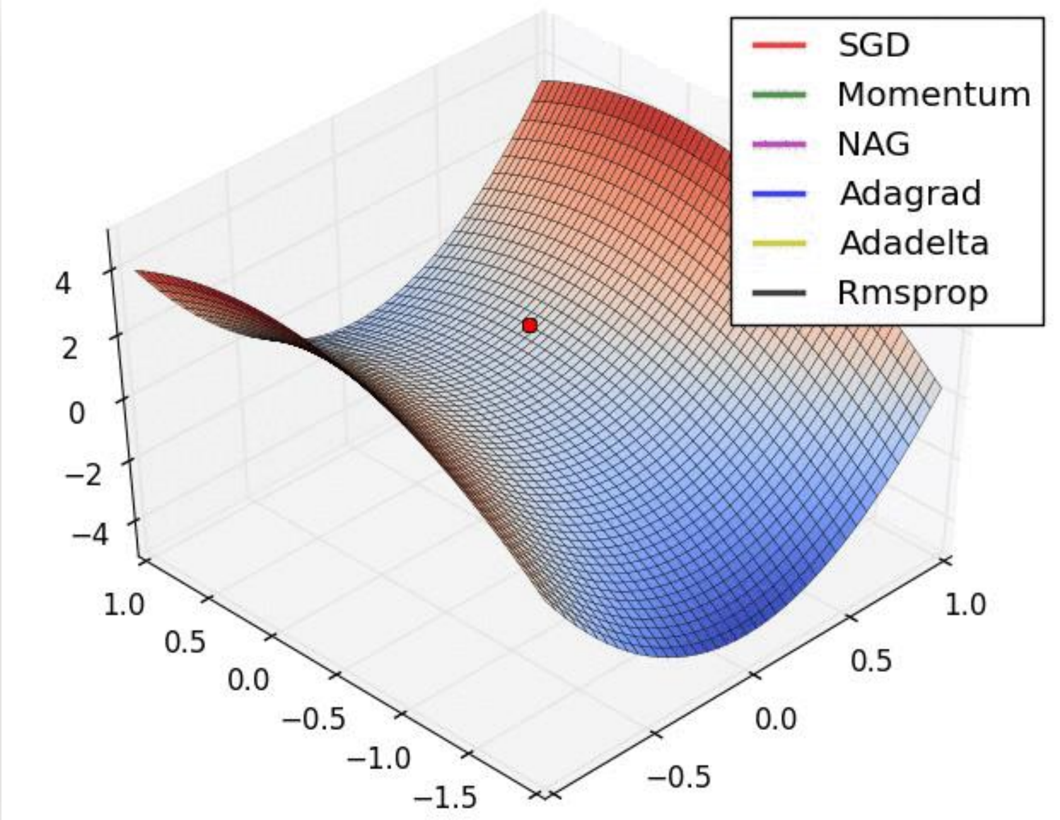
Nestrov

……

各类优化器表现Gif

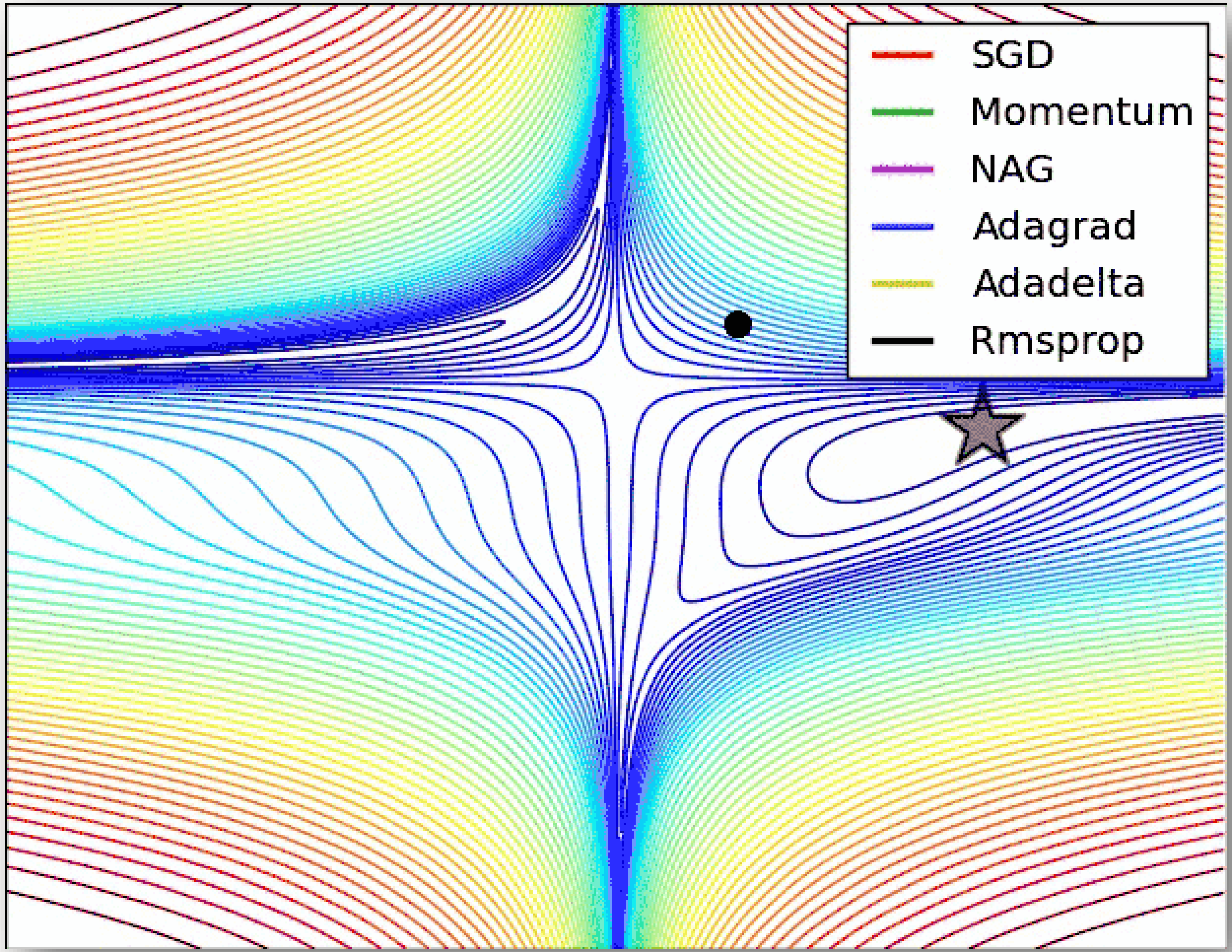


三处低地的曲面.gif



鞍点曲面.gif

各类优化器表现Gif



峡谷地貌.gif

· Thanks ·

学生创新中心：肖雄子彦