西安电子科技大学
XIDIAN UNIVERSITY

IPIU
智能感知与图像理解

最优化理论
第五章：无约束最优化方法

人工智能学院
智能感知与图像理解实验室

无约束最优化方法

1. 随机梯度降
2. 小批量随机梯度降
3. 动量
4. 动量改进算法和对比



无约束最优化方法

1. 随机梯度降
2. 小批量随机梯度降
3. 动量
4. 动量改进算法和对比



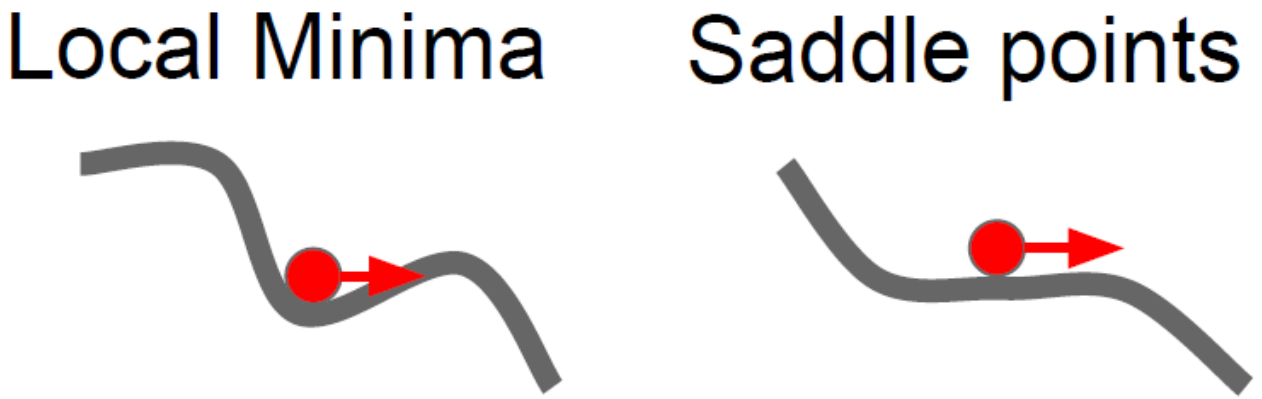
最速下降法

向梯度反方向行进 $X_{k+1} = X_k - t_k \nabla f(X_k)$

最速下降法

Gradient Descent

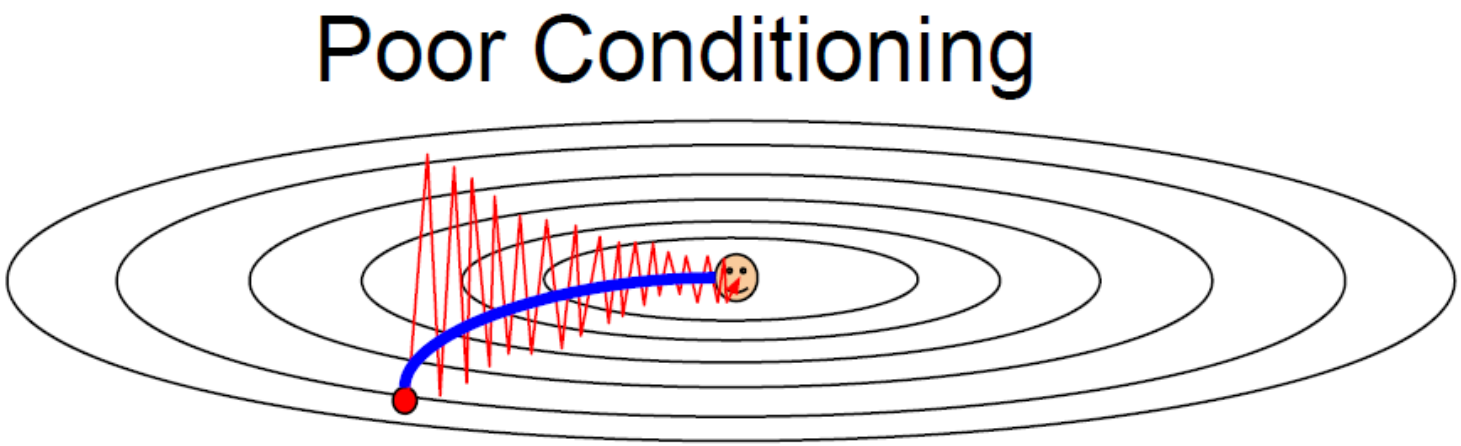
◆该方法利用目标函数的局部性质，得到局部最优解，具有一定的“盲目性”无法解决鞍点问题



最速下降法

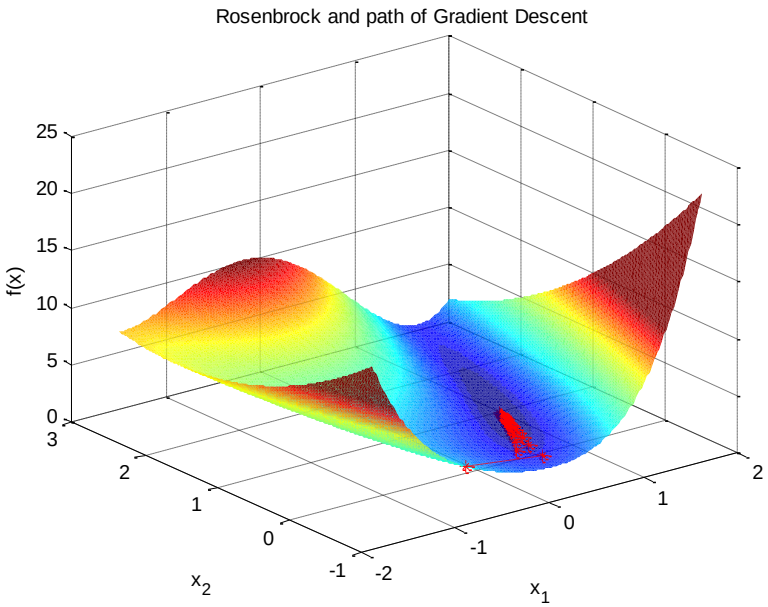
Gradient Descent

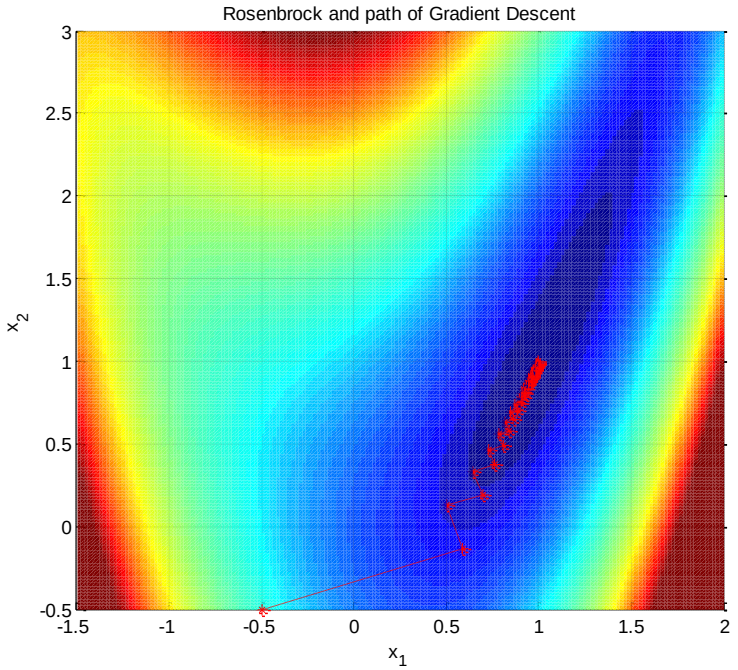
◆每一次迭代的移动方向都与出发点的等高线垂直，其锯齿现象（ zig-zagging）将会导致收敛速度变慢



Rosenbrock函数上的例子

最速下降法





随机梯度降

机器学习中，目标函数通常是训练数据集中有关各个样本的损失函数的平均。设$f_i(x)$是有关索引为i的训练数据样本的损失函数，n 是训练数据样本数，x 是模型的参数向量，那么目标函数定义为：

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$$

目标函数在x处的梯度计算为：

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$$



随机梯度降

如果使用梯度下降，每次自变量迭代的计算开销为O(n) ，它随着n 线性增长。因此，当训练数据样本数很大时，梯度下降每次迭代的计算开销很高。

随机梯度下降(stochastic gradient descent, SGD) 减少了每次迭代的计算开销。在随机梯度下降的每次迭代中，我们随机均匀采样一个索引为i 的样本$i \in \{1, \cdots n\}$ ，并计算梯度$\nabla f_i(x)$来迭代x ：

$$x_{k+1} \leftarrow x_k - t\nabla f_i(x_k)$$



随机梯度降

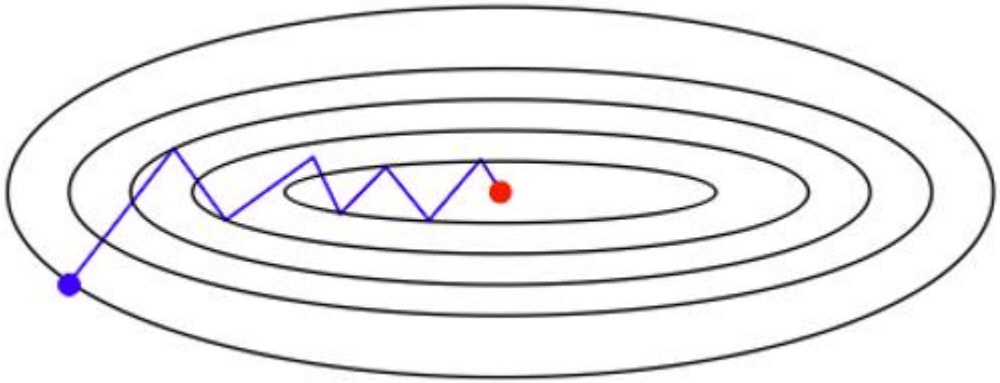
这里t是固定步长，在深度学习中也叫学习率，可以看到每次迭代的计算开销从梯度下降的$O(n)$降到了常数$O(1)$. 需要指出的是，随机梯度$\nabla f_i(x)$是对梯度$\nabla f(x)$的无偏估计：

$$E[\nabla f_i(x)] = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(x) = \nabla f(x)$$

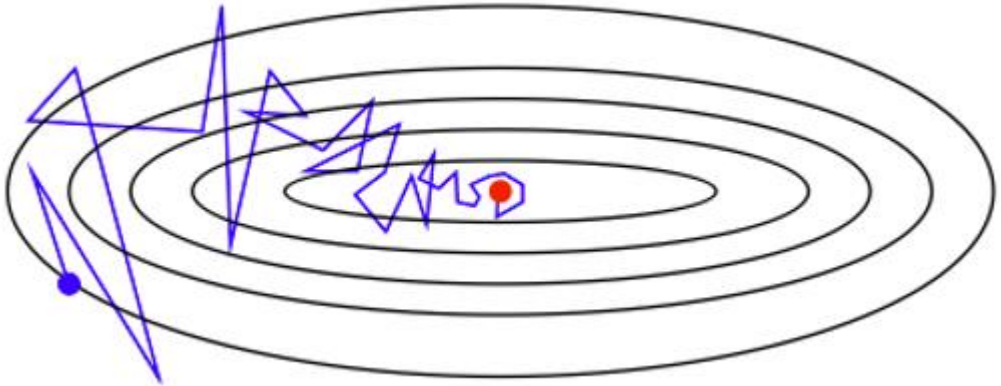
这意味着，平均来说，随机梯度是对梯度的一个良好的估计。



随机梯度降



梯度降



随机梯度降



无约束最优化方法

1. 随机梯度降
2. 小批量随机梯度降
3. 动量
4. 动量改进算法和对比



小批量随机梯度降

为了在梯度降和随机梯度降中取得一个平衡，我们可以在每轮迭代中随机均匀采样多个样本来组成一个小批量，然后用这个小批量来计算梯度。

在梯度下降的第k+1步中，小批量随机梯度降随机均匀采样一个由训练数据样本索引组成的小批量数据B_{k+1}。我们可以通过重复采样(sampling with replacement) 或者不重复采样(sampling without replacement)得到一个小批量中的各个样本，相应的第k+1步小批量$\mathcal{B}_{k+1}$数据的目标函数位于$\mathbf{x}_k$处的梯度$\mathbf{g}_k$可计算为：

$$\mathbf{g}_k \leftarrow \nabla f_{\mathcal{B}_{k+1}}(\mathbf{x}_k) = \frac{1}{|\mathcal{B}_{k+1}|}\sum_{i\in\mathcal{B}_{k+1}} \nabla f_i(\mathbf{x}_k)$$

$|\mathcal{B}|$表示批量中数据的个数。



小批量随机梯度降

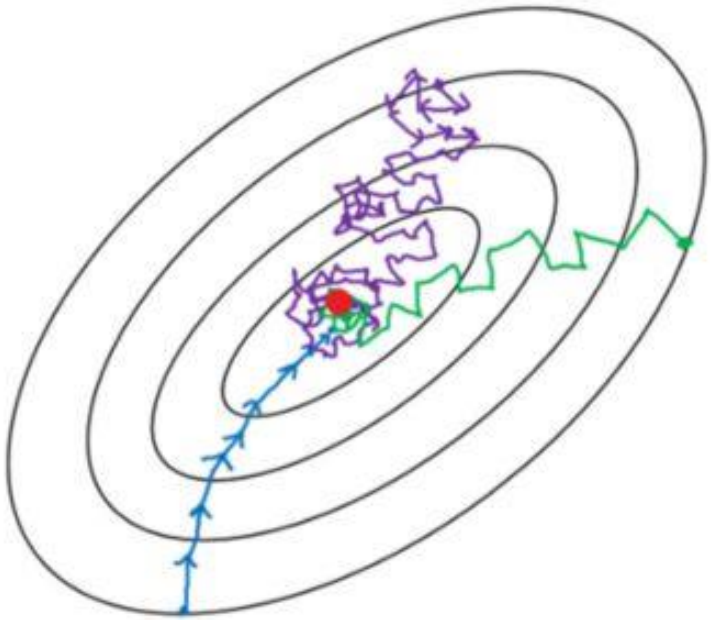
同随机梯度一样，重复采样所得的小批量随机梯度$\mathbf{g}_k$ 也是对梯度$\nabla f\left(\mathbf{x}_{k-1}\right)$的无偏估计。给定学习率$\lambda$ ，则小批量随机梯度下降对自变量的迭代如下：

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \lambda \mathbf{g}_k$$

小批量随机梯度下降中每次迭代的计算开销为$O(|B_k|)$.当批量大小为1 时，该算法即为随机梯度下降；当批量大小等于训练数据样本数时，该算法即为梯度下降。



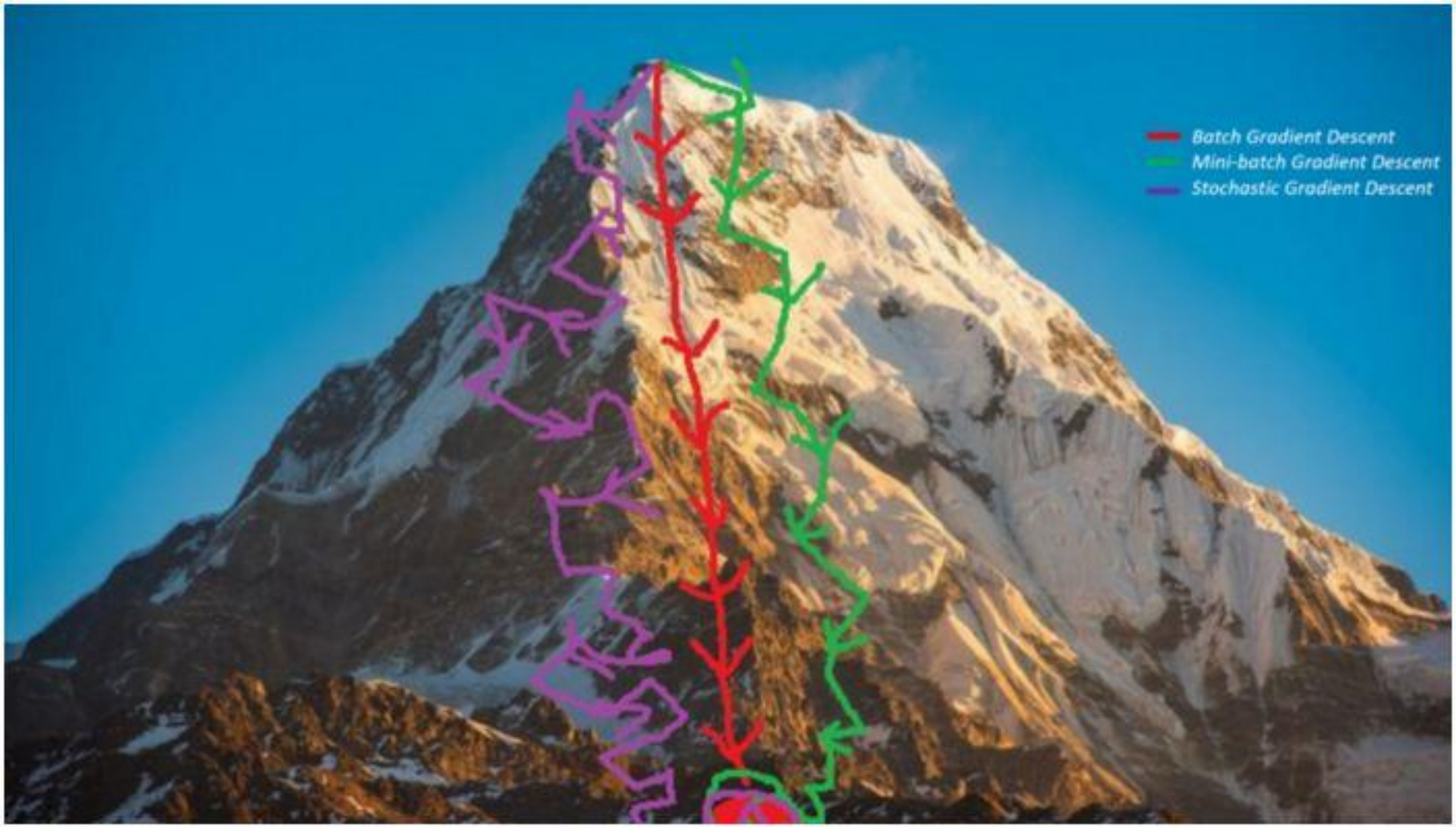
小批量随机梯度降



Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent



小批量随机梯度降


Batch Gradient Descent
Mini-batch Gradient Descent
Stochastic Gradient Descent



小批量随机梯度降

小批量随机梯度降面临的挑战：

◆学习率选取比较困难

◆与梯度降类似，SGD容易收敛到局部最优，并且在某些情况下可能被困在鞍点



无约束最优化方法

1. 随机梯度降
2. 小批量随机梯度降
3. 动量
4. 动量改进算法和对比



动量

动量法的提出是为了解决小批量随机梯度下降的上述问题，对每次迭代的步骤做如下修改：

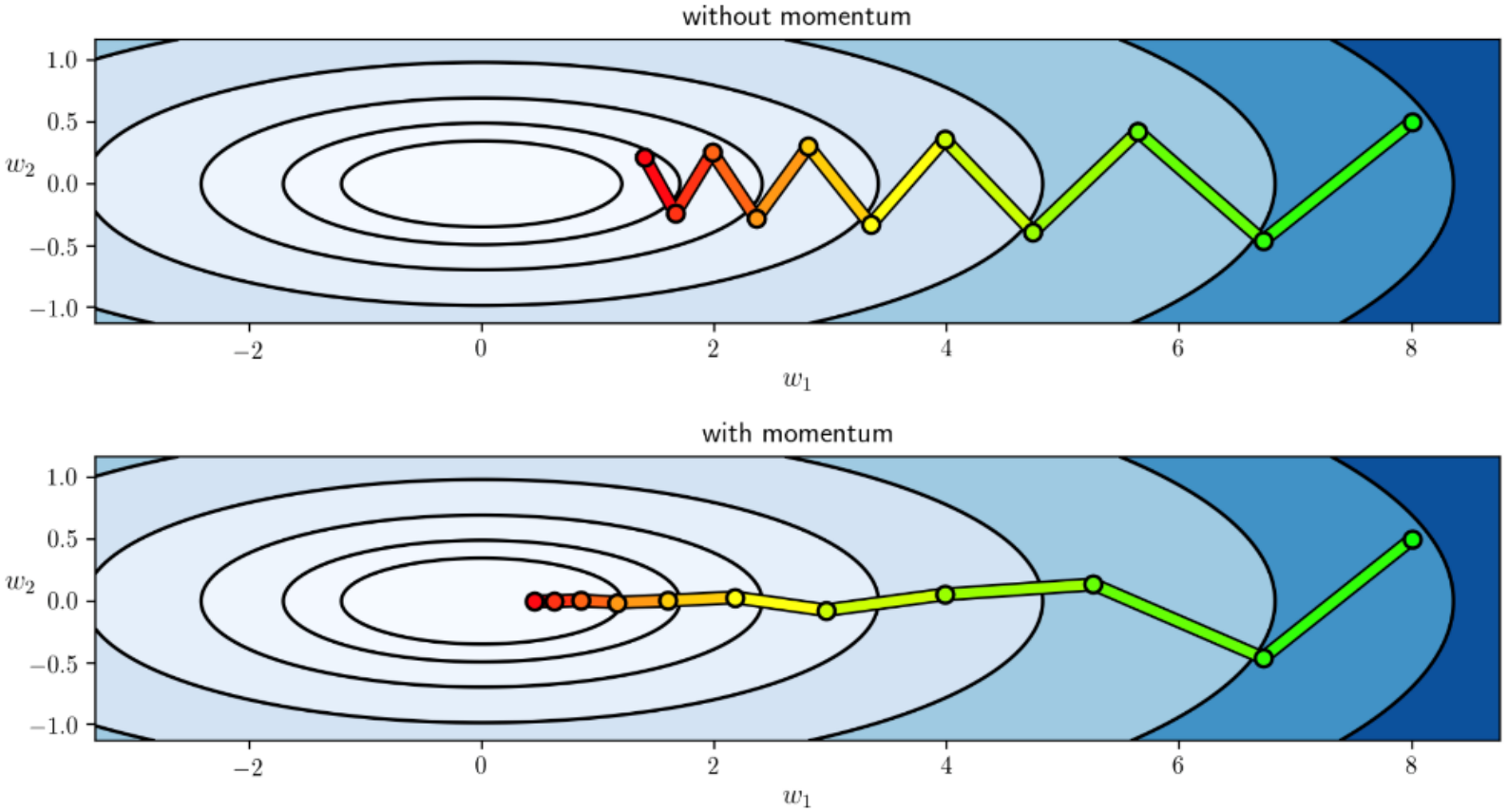
$$\mathbf{v}_{k+1} \leftarrow \gamma \mathbf{v}_k + \lambda_k \mathbf{g}_k,$$

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \mathbf{v}_k$$

其中，动量超参数γ满足$0 \le \gamma < 1$,当γ=0时，动量法等价于小批量随机梯度下降。



动量


without momentum
1.0
0.5
0.0
−0.5
−1.0
w_2
−2
0
2
4
6
8
w_1
with momentum
1.0
0.5
0.0
−0.5
−1.0
w_2
−2
0
2
4
6
8
w_1



动量

◆特点：

◆动量是对梯度的叠加，在和上一次梯度方向一致的时候，能够进行很好的加速；

◆因为有记忆效应，所以在陷入局部最小值的时候能够跳出陷阱；

◆在梯度改变方向的时候，动量能减少迭代次数；

总之，momentum项能够在相关方向加速SGD，抑制振荡，从而加快收敛。



无约束最优化方法

1. 随机梯度降
2. 小批量随机梯度降
3. 动量
4. 动量改进算法和对比



动量相关改进算法

Nesterov Momentum

在计算参数的梯度时，在损失函数中考虑了动量项，即$\mathbf{g}_k = \nabla \mathrm{f}\left(x_k + \gamma v_k\right)$，这种方式预估了下一次参数所在的位置。

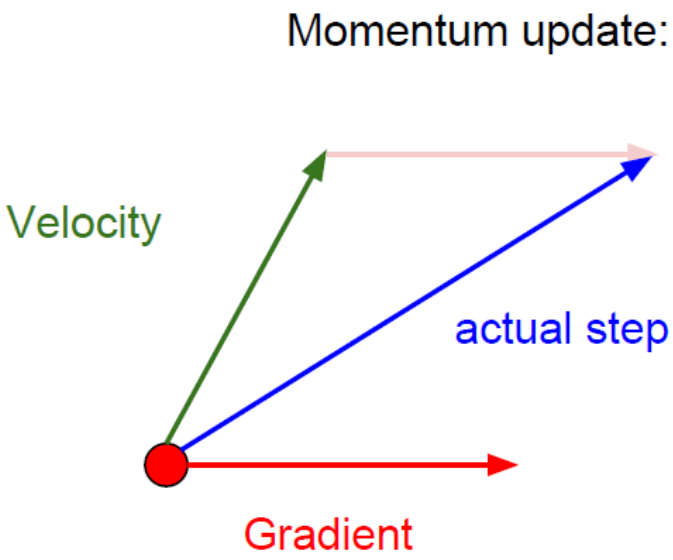
$$\mathbf{v}_{k+1} \leftarrow \gamma \mathbf{v}_k - \lambda_k \nabla \mathrm{f}\left(x_k + \gamma v_k\right),$$

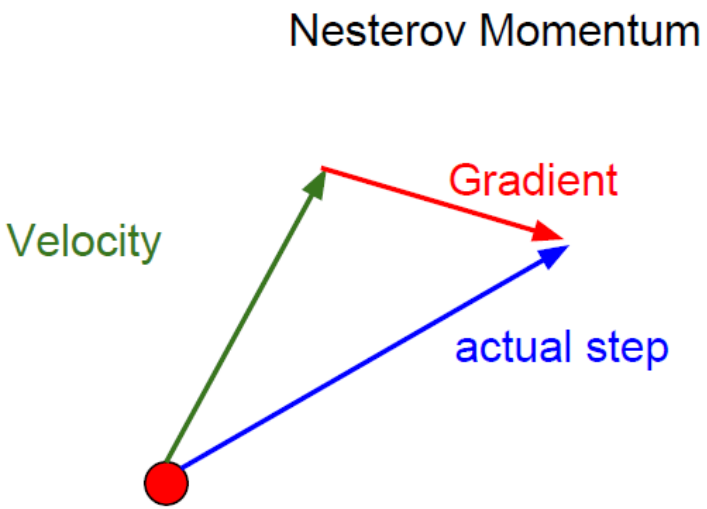
$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{v}_{k+1}$$



动量相关改进算法

Nesterov Momentum对比

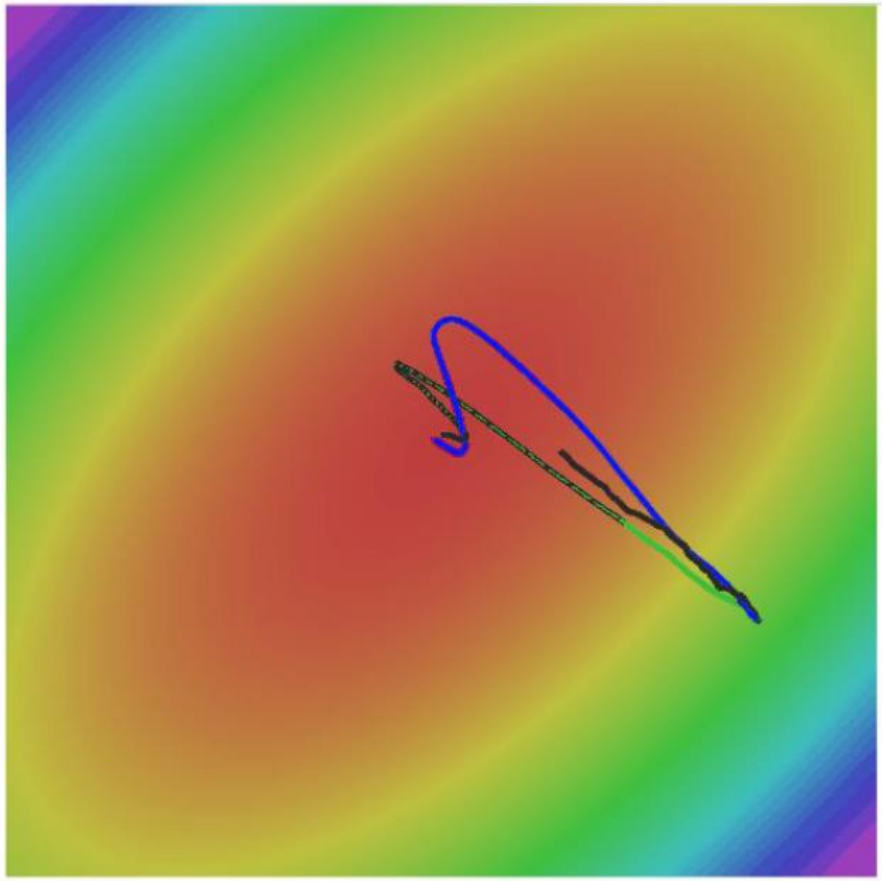






动量相关改进算法

Nesterov Momentum对比





动量相关改进算法

AdaGrad

在之前介绍的梯度降算法中，目标函数的自变量的每一个分量在某一时刻都使用同一个学习率来迭代。当不同分量的梯度值有较大差别时，需要选取较小的学习率使得梯度降保持稳定，降低发散的可能，但是牺牲了更新速度。

AdaGrad算法根据自变量在每个维度的梯度值的大小来调整各个维度上的学习率，从而避免统一的学习率难以适应所有维度的问题。



动量相关改进算法

AdaGrad引入一个小批量随机梯度g_k 按元素平方的累加变量s_k，并初始化为0 ，迭代法则为：

$$\mathbf{s}_{k+1} \leftarrow \mathbf{s}_k + \mathbf{g}_k \odot \mathbf{g}_k$$

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \frac{\lambda}{\sqrt{s_{k+1} + \varepsilon}} \odot \mathbf{g}_k$$

缺点：随着迭代次数的增加，梯度平方的累加将会使s_k 越来越大，使$\frac{\lambda}{\sqrt{s_{k+1} + \varepsilon}} \to 0$，难以收敛到最优解。



动量相关改进算法

RMSProp

为了解决AdaGrad随着迭代次数的增加，学习率趋向于零，难以收敛到最优解的问题，RMSProp对AdaGrad做了一点修改。引入了一个参数$0 \le \gamma < 1$对s_k做加权移动平均。

$$s_{k+1} \leftarrow \gamma s_k + (1-\gamma) g_k \odot g_k$$

$$x_{k+1} \leftarrow x_k - \frac{\lambda}{\sqrt{s_{k+1} + \varepsilon}} \odot g_k$$



动量相关改进算法

AdaDelta

除了RMSProp，另一个算法AdaDelta 也对AdaGrad随着迭代次数的增加学习率趋向于0，难以收敛到最优解的问题做了改进。与RMSProp类似，AdaDelta也引入了一个参数$0 \le \gamma < 1$对s_k 做加权移动平均。

$$s_{k+1} \leftarrow \gamma s_k + (1-\gamma) g_k \odot g_k$$

不同的是，AdaDelta引入一个额外的状态变量Δx_t ，初始化为0 ，用来计算学习率

$$\mathbf{g}_k^{'} \leftarrow \sqrt{\frac{\Delta \mathbf{x}_k + \epsilon}{s_{k+1} + \epsilon}} \odot \mathbf{g}_k$$

迭代法则为：

$$x_{k+1} \leftarrow x_k - \mathbf{g}_k^{'}$$



动量相关改进算法

最后，用$\Delta\mathbf{x}_{k+1}$来记录自变量变化量$\mathbf{g}_k'$ 按元素平方的加权移动平均：

$$\Delta\mathbf{x}_{k+1} \leftarrow \gamma\Delta\mathbf{x}_k + (1-\gamma)\mathbf{g}_k' \odot \mathbf{g}_k'$$

可以看到，AdaDelta与RMSProp的不同之处在于使用$\sqrt{\Delta x_k + \varepsilon}$来代替学习率。



动量相关改进算法

Adam

Adam算法结合了动量变量v_t和RMSProp 算法中引入梯度移动加权平均s_k的思想，并将v_k和s_k初始化为0，引入两个超参数$0 \le \beta_1, \beta_2 < 1$，其动量计算为小批量随机梯度的移动加权平均：

$$\mathbf{v}_{k+1} \leftarrow \beta_1 \mathbf{v}_k + (1-\beta_1)\mathbf{g}_k$$

与RMSProp类似，对小批量随机梯度元素平方也进行加权移动平均：

$$\mathbf{s}_{k+1} \leftarrow \beta_2 \mathbf{s}_k + (1-\beta_2)\mathbf{g}_k \odot \mathbf{g}_k$$



动量相关改进算法

由于我们将v_k和s_k初始化为0 ， 考察v_{k+1}的值：

$$v_1=\beta_1 v_0+(1-\beta_1)g_1$$

$$v_2=\beta_1 v_1+(1-\beta_1)g_2=\beta_1^2 v_0+\beta_1(1-\beta_1)g_1+(1-\beta_1)g_2$$

$$v_3=\beta_1 v_2+(1-\beta_1)g_3=\beta_1^3 v_0+\beta_1^2(1-\beta_1)g_1+\beta_1(1-\beta_1)g_2+(1-\beta_1)g_3$$

$$\vdots$$

$$v_{k+1}=(1-\beta_1)\sum_{i=1}^{k+1}\beta_1^{k+1-i}g_i$$

将g_i权重相加得到

$$(1-\beta_1)\sum_{i=1}^{k+1}\beta_1^{k+1-i}=\sum_{i=1}^{k+1}\beta_1^{k+1-i}-\sum_{i=1}^{k+1}\beta_1^{k+2-i}=1-\beta_1^{k+1}$$



动量相关改进算法

需要注意的是，当k较小时，过去各时间小批量随机梯度权值之和会较小，例如当$\beta_1 = 0.9$时，$\mathbf{v}_1 = 0.1g_1$。为了消除这样的影响，对任意时间k+1，将v_{k+1}除以1-β_1^{k+1}做偏差修正，从而使过去时间小批量随机梯度权值之和为1

$$\hat{\mathbf{v}}_{k+1} \leftarrow \frac{\mathbf{v}_{k+1}}{1-\beta_1^{k+1}}$$

类似地对s_t也做修正

$$\hat{s}_{k+1} \leftarrow \frac{s_{k+1}}{1-\beta_2^{k+1}}$$



动量相关改进算法

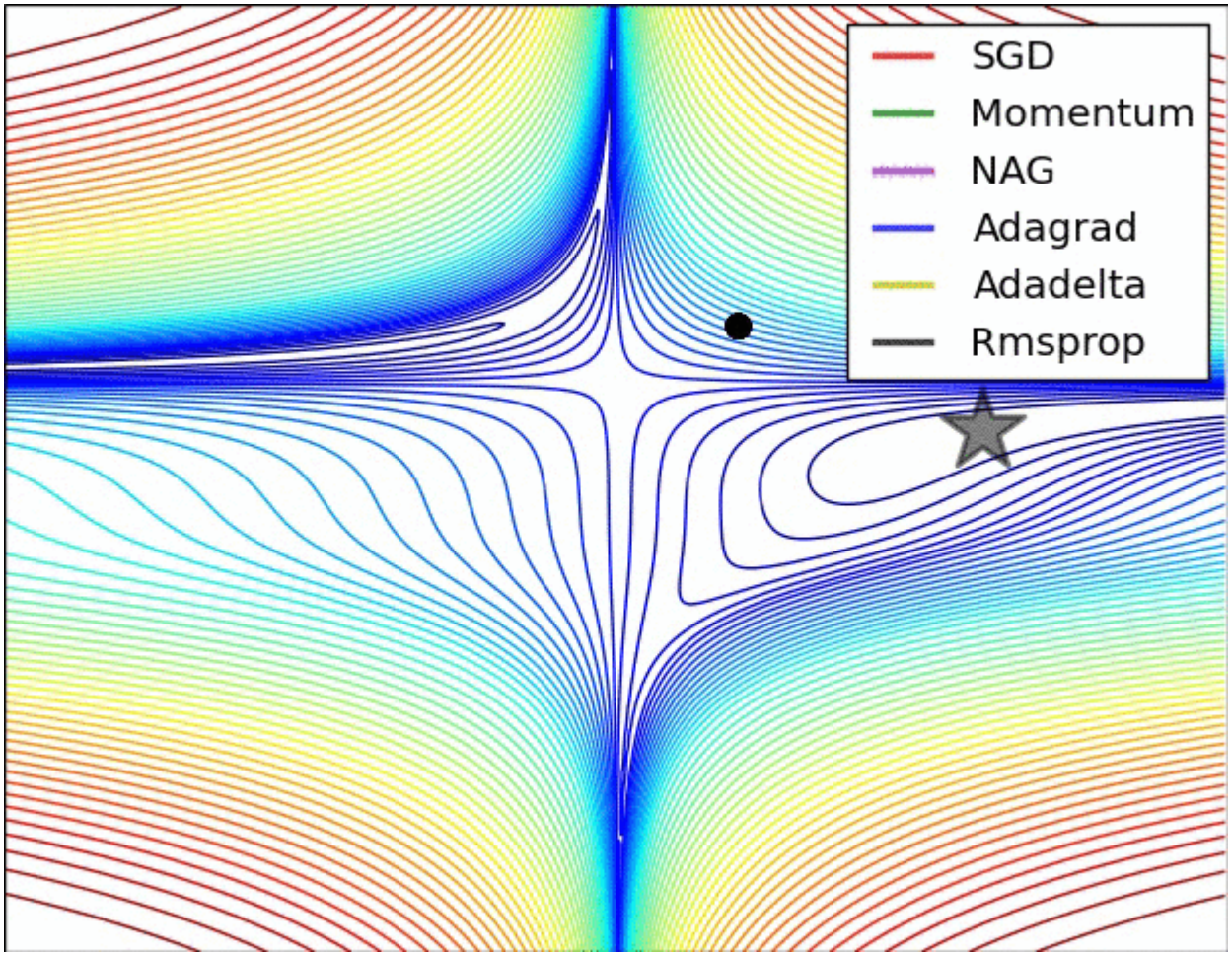
接下来类似RMSProp，用修正后的$\hat{s}_k$对动量$\hat{\mathbf{v}}_k$每个元素的学习率按照元素进行调整，并进行迭代：

$$\mathbf{g}'_{k+1} \leftarrow \frac{\lambda \mathbf{v}_{k+1}}{\sqrt{\hat{s}_{k+1} + \varepsilon}}$$

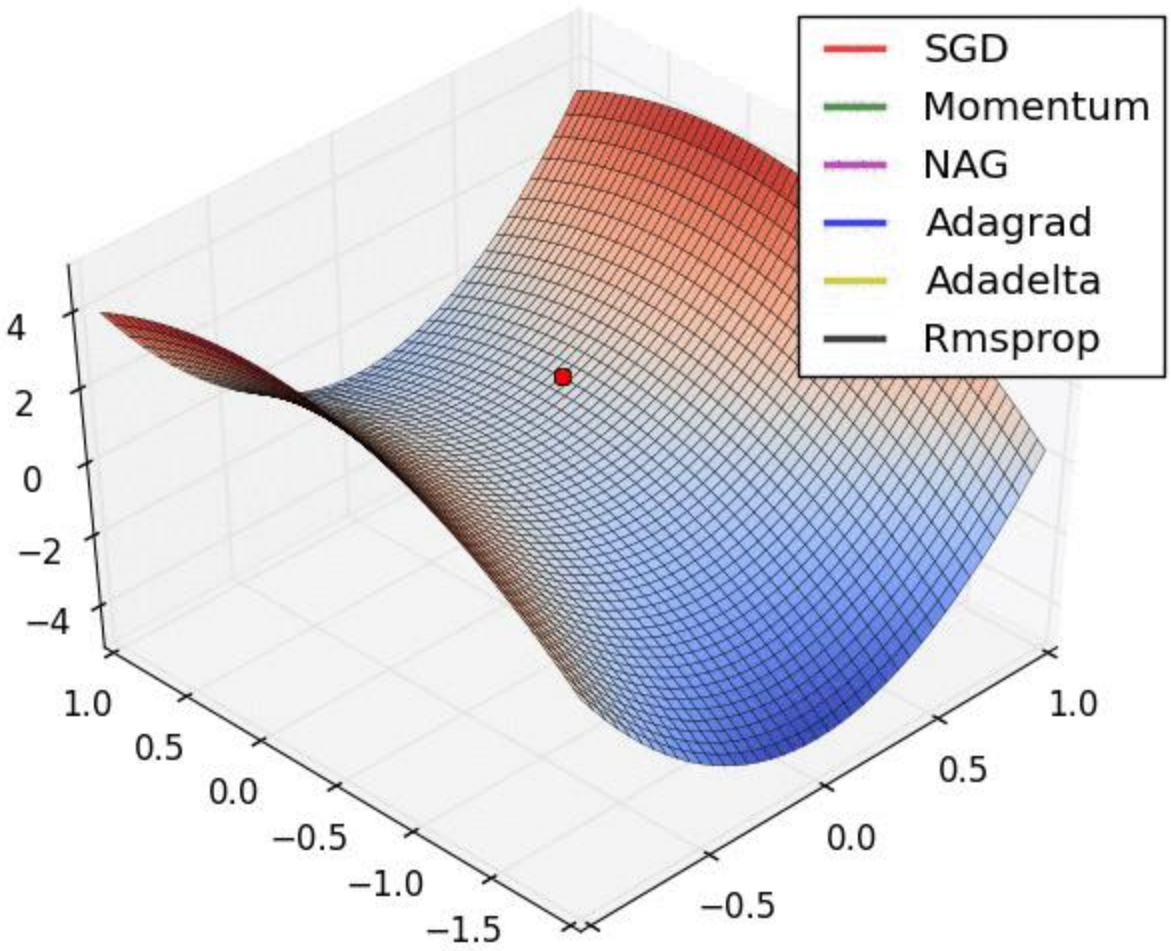
$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \mathbf{g}'_{k+1}$$



动量相关改进算法对比



动量相关改进算法对比


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5



动量相关改进算法对比

算法	优点	缺点	适用情况
批量梯度下降	目标函数为凸函数时，可以找到全局最优值	收敛速度慢，需要用到全部数据，内存消耗大	不适用于大数据集，不能在线更新模型
随机梯度下降	避免冗余数据的干扰，收敛速度加快，能够在线学习	更新值的方差较大，收敛过程会产生波动，可能落入极小值，选择合适的学习率比较困难	适用于需要在线更新的模型，适用于大规模训练样本情况
小批量梯度下降	降低更新值的方差，收敛较为稳定	选择合适的学习率比较困难	
Momentum	能够在相关方向加速SGD，抑制振荡，从而加快收敛	需要人工设定学习率	适用于有可靠的初始化参数
Nesterov	梯度在大的跳跃后，进行计算对当前梯度进行校正	需要人工设定学习率	
Adagrad	不需要对每个学习率手工地调节	仍依赖于人工设置一个全局学习率，学习率设置过大，对梯度的调节太大。中后期，梯度接近于0，使得训练提前结束	需要快速收敛，训练复杂网络时；适合处理稀疏梯度
Adadelta	不需要预设一个默认学习率，训练初中期，加速效果不错，很快，可以避免参数更新时两边单位不统一的问题。	训练后期，反复在局部最小值附近抖动	需要快速收敛，训练复杂网络时
RMSprop	解决 Adagrad 激进的学习率缩减问题	依然依赖于全局学习率	需要快速收敛，训练复杂网络时；适合处理非平稳目标 - 对于RNN效果很好
Adam	对内存需求较小，为不同的参数计算不同的自适应学习率		需要快速收敛，训练复杂网络时；善于处理稀疏梯度和处理非平稳目标的优点，也适用于大多非凸优化 - 适用于大数据集和高维空间

西安电子科技大学
XIDIAN UNIVERSITY

IPIU
智能感知与图像理解

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education

THE END

Thanks for your participation!