优化方法之一阶优化

一阶优化主要包括，GD, SGD, Momentum, Nesterov Momentum, AdaGrad, RMSProp, Adam。 其中GD,SGD,Momentum,Nesterov Momentum是手动指定学习速率的,而后面的AdaGrad, RMSProp, Adam,就能够自动调节学习速率.

1、梯度下降（GD）

梯度下降算法(GD)或者最速下降法(SD)，是求解无约束最优化问题的的一种常用的方法。特点是实现简单，梯度下降法是一种迭代法，每一步需要求解目标函数的梯度向量。

假设 $f(x)$ 是 R^n 上具有一阶连续偏导数的函数，要求解的无约束最优化问题是：

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

求解之后，x^* 表示目标函数 $f(x)$ 的极小值点。梯度下降法是一种迭代方法，选取适当的初值 $x^{(0)}$ ，不断迭代，更新 x 的值，进行目标函数的极小化，直到收敛。由于负梯度的方向是使函数值下降最快的方向，在迭代的每一步，以负梯度的方向更新 x 的值，从而达到减少目标函数值的目的。

由于 $f(x)$ 具有一阶连续偏导数，若第 k 次迭代值为 $x^{(k)}$ ，则可将 $f(x)$ 在 $x^{(k)}$ 附近进行一阶泰勒展开：

$$f(x) = f(x^{(k)}) + g_k^T(x - x^{(k)})$$

其中，$g_k = g(x^{(k)}) = \nabla f(x^{(k)})$ 为 $f(x)$ 在 $x^{(k)}$ 的梯度，梯度有正有负，通过梯度来决定是加还是减。

求出第 $k+1$ 次迭代值 $x^{(k+1)}$:

$$x^{(k+1)} \leftarrow x^{(k)} - \lambda_k g_k$$

其中 λ_k 表示步长，由一维搜索确定，但是我们通常给定一个很小的固定值。

$$f(x^{(k)} - \lambda_k g_k) = \min_{\lambda \geq 0} f(x^{(k)} - \lambda_k g_k)$$

梯度下降法
输入：目标函数 $f(x)$ ，梯度函数 $g(x) = \nabla f(x)$ ，计算精度 ε
输出：$f(x)$的极小点 x^*

- 取初始值 $x^{(0)} \in R^n$ ，设 $k = 0$
- 计算 $f(x^{(k)})$
- 计算梯度 $g_k = g(x^{(k)})$，当$\|g_k\| < \varepsilon$，停止迭代，令 $x^* = x^{(k)}$ ，否则求 λ_k 使得：

$$f(x^{(k)} - \lambda_k g_k) = \min_{\lambda \geq 0} f(x^{(k)} - \lambda_k g_k)$$

- 迭代 $x^{(k+1)} = x^{(k)} - \lambda_k g_k$，计算 $f(x^{(k+1)})$
- 如果 $x^{(k+1)} - x^{(k)} \leq \varepsilon$，停止迭代，否则继续搜索。

当目标函数是凸函数时，梯度下降算法的解是全局最优解，一般情况下，其解不保证是全局最优解。梯度下降法也不一定是很快的。

2、批梯度下降（BGD）

即batch gradient descent。在训练中，每一步迭代都使用训练集的所有内容。也就是说，利用现有参数对训练集中的每一个输入生成一个估计输出 $\hat{y}_i$，然后跟实际输出 y_i 比较，统计所有误差，求平均以后得到平均误差，以此来作为更新参数的依据。

具体实现：
输入：学习速率 ε，初始参数 θ
输出：整个模型

每步迭代过程:

- 提取训练集中的所有内容$\{x_1, \ldots, x_n\}$，以及相关的输出 y_i
- 计算梯度和误差并更新参数:

$$\hat{g} \leftarrow +\frac{1}{n}\sum_i L(f(x_i;\theta), y_i) \quad \text{表示平均损失梯度}$$
$$\hat{\theta} \leftarrow \theta - \varepsilon\hat{g}$$

优点:
由于每一步都利用了训练集中的所有数据，因此当损失函数达到最小值以后，能够保证此时计算出的梯度为**0**，换句话说，就是能够收敛。因此，使用BGD时不需要逐渐减小学习速率 ε_k

缺点:
由于每一步都要使用所有数据，因此随着数据集的增大，运行速度会越来越慢。

3、随机梯度下降（SGD）

SGD全名 stochastic gradient descent， 即随机梯度下降。不过这里的**SGD**其实跟**MBGD(minibatch gradient descent)**是一个意思，即随机抽取一批样本，以此为根据来更新参数。

具体实现：
输入：学习速率 ε，初始参数 θ
输出：整个模型

每步迭代过程:

- 提取训练集中随机抽取一批容量为 m 的样本 $\{x_1, \ldots, x_m\}$，以及相关的输出 y_i

- 计算梯度和误差并更新参数:

$$\hat{g} \leftarrow +\frac{1}{m}\sum_{i} L(f(x_i;\theta),y_i) \quad \text{表示平均损失梯度}$$

$$\hat{\theta} \leftarrow \theta - \varepsilon\hat{g}$$

优点:
训练速度快，对于很大的数据集，也能够以较快的速度收敛。

缺点:
由于是抽取，因此不可避免的，得到的梯度肯定有误差。因此学习速率需要逐渐减小，否则模型无法收敛。因为误差，所以每一次迭代的梯度受抽样的影响比较大，也就是说梯度含有比较大的噪声，不能很好的反映真实梯度。

学习速率该如何调整:
那么这样一来，ε 如何衰减，就成了问题。如果要保证SGD收敛，应该满足如下两个要求:

$$\sum_{k=1}^{\infty}\varepsilon_k = \infty$$

$$\sum_{k=1}^{\infty}\varepsilon_k^2 < \infty$$

而在实际操作中，一般是进行线性衰减:

$$\varepsilon = (1-\alpha)\varepsilon_0 + \alpha\varepsilon_\tau$$

$$\alpha = \frac{k}{\tau}$$

其中 ε_0 是初始学习率，ε_τ 是最后一次迭代的学习率。τ 代表迭代次数。一般来说，ε_τ 设为 ε_0 的 **1%** 比较合适。而 τ 一般设为让训练集中的每个数据都输入模型上百次比较合适。那么初始学习率 ε_0 怎么设置呢?书上说，你先用固定的学习速率迭代**100**次，找出效果最好的学习速率，然后 ε_0 设为比它大一点就可以了。

4、Momentum

上面的SGD有个问题，就是每次迭代计算的梯度含有比较大的噪音。 而Momentum方法可以比较好的缓解这个问题，尤其是在面对小而连续的梯度，但是含有很多噪声的时候，可以很好的加速学习。Momentum借用了物理中的动量概念，即前几次的梯度也会参与运算。为了表示动量，引入了一个新的变量 $v(velocity)$，v 是之前的梯度的累加，但是每回合都有一定的衰减。

具体实现:
需要：学习速率 ε，初始参数 θ， 初始速率 v， 动量衰减参数 α
每步迭代过程:

1. 从训练集中的随机抽取一批容量为 m 的样本 $\{x_1,\ldots,x_m\}$，以及相关的输出 y_i

2. 计算梯度和误差，并更新速度 v 和参数 θ:

$$\hat{g} \leftarrow +\frac{1}{m}\sum_i L(f(x_i;\theta),y_i) \quad \text{表示平均损失梯度}$$

$$v \leftarrow \alpha v - \varepsilon\hat{g}$$

$$\hat{\theta} \leftarrow \theta + v$$

其中参数 α 表示每回合速率 v 的衰减程度。同时也可以推断得到，如果每次迭代得到的梯度都是 g，那么最后得到的 v 的稳定值为:

$$\frac{\epsilon\|g\|}{1-\alpha}$$

也就是说，Momentum最好情况下能够将学习速率加速 $\frac{1}{1-\alpha}$ 倍。一般 α 的取值有0.5，0.9，0.99这几种。当然，也可以让 α 的值随着时间而变化，一开始小点，后来再加大。不过这样一来，又会引进新的参数。

特点:
前后梯度方向一致时，能够加速学习，前后梯度方向不一致时，能够抑制震荡。

5、Nesterov Momentum

这是对之前的Momentum的一种改进，大概思路就是，先对参数进行估计，然后使用估计后的参数来计算误差

具体实现:
需要：学习速率 ε，初始参数 θ， 初始速率 v， 动量衰减参数 α
每步迭代过程:

1. 从训练集中的随机抽取一批容量为 m 的样本 $\{x_1,\ldots,x_m\}$，以及相关的输出 y_i
2. 计算梯度和误差，并更新速度 v 和参数 θ:

$$\hat{g} \leftarrow +\frac{1}{m}\sum_i L(f(x_i;\theta+\alpha v),y_i) \quad \text{表示先对参数进行了估计}$$

$$v \leftarrow \alpha v - \varepsilon\hat{g}$$

$$\hat{\theta} \leftarrow \theta + v$$

注意在估算 $\hat{g}$ 的时候,参数变成了 $\theta+\alpha v$ 而不是之前的 θ

6、AdaGrad方法

AdaGrad可以自动变更学习速率，只是需要设定一个全局的学习速率 ε，但是这并非是实际学习速率，实际的速率是与以往参数的模之和的开方成反比的。也许说起来有点绕口，不过用公式来表示就直白的多:

$$\epsilon_n = \frac{\epsilon}{\delta + \sqrt{\sum_{i=1}^{n-1} g_i \odot g_i}}$$

其中 δ 是一个很小的常量，大概在 10^{-7}，防止出现除以0的情况。

具体实现:
需要:全局学习速率 ε，初始参数 θ，数值稳定量 δ
中间变量: 梯度累计量r (初始化为0)
每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本 $\{x_1, \ldots, x_m\}$，以及相关的输出 y_i
2. 计算梯度和误差，更新 r，再根据 r 和梯度计算参数更新量

$$\hat{g} \leftarrow + \frac{1}{m} \nabla_\theta \sum_i L(f(x_i; \theta), y_i)$$
$$r \leftarrow r + \hat{g} \odot \hat{g}$$
$$\Delta\theta = - \frac{\epsilon}{\delta + \sqrt{r}} \odot \hat{g}$$
$$\theta \leftarrow \theta + \Delta\theta$$

优点:
能够实现学习率的自动更改。如果这次梯度大，那么学习速率衰减的就快一些，如果这次梯度小，那么学习速率衰减的就慢一些。

缺点:
仍然要设置一个变量 ε 经验表明，在普通算法中也许效果不错，但在深度学习中，深度过深时会造成训练提前结束。

7、RMSProp

RMSProp通过引入一个衰减系数，让 r 每回合都衰减一定比例，类似于Momentum中的做法。

具体实现:
需要:全局学习速率 ε，初始参数 θ，数值稳定量 δ，梯度累计量衰减速率 ρ
中间变量: 梯度累计量r (初始化为0)
每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本 $\{x_1, \ldots, x_m\}$，以及相关的输出 y_i
2. 计算梯度和误差，更新 r，再根据 r 和梯度计算参数更新量

$$\hat{g} \leftarrow +\frac{1}{m}\nabla_\theta \sum_i L(f(x_i;\theta),y_i)$$
$$r \leftarrow \rho r + (1-\rho)\hat{g}\odot\hat{g}$$
$$\Delta\theta = -\frac{\epsilon}{\delta+\sqrt{r}}\odot\hat{g}$$
$$\theta \leftarrow \theta + \Delta\theta$$

优点：
相比于AdaGrad，这种方法很好的解决了深度学习中过早结束的问题，适合处理非平稳目标，对于RNN效果很好

缺点：
又引入了新的超参，衰减系数 ρ，依然依赖于全局学习速率

8、RMSProp with Nesterov Momentum

当然，也有将RMSProp和Nesterov Momentum结合起来的

具体实现**:**
需要:全局学习速率 ε，初始参数 θ，初始速率 v，动量衰减系数 α，梯度累计量衰减速率 ρ
中间变量: 梯度累计量r (初始化为0)
每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本 $\{x_1,\ldots,x_m\}$，以及相关的输出 y_i
2. 计算梯度和误差，更新 r，再根据 r 和梯度计算参数更新量

$$\tilde{\theta} \leftarrow \theta + \alpha v$$
$$\hat{g} \leftarrow +\frac{1}{m}\nabla_{\tilde{\theta}} \sum_i L(f(x_i;\tilde{\theta}),y_i)$$
$$r \leftarrow \rho r + (1-\rho)\hat{g}\odot\hat{g}$$
$$v \leftarrow \alpha v - \frac{\epsilon}{\sqrt{r}}\odot\hat{g}$$
$$\theta \leftarrow \theta + v$$

9、Adam

Adam(Adaptive Moment Estimation)本质上是带有动量项的**RMSprop**，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

具体实现**:**
需要:全局学习速率 ε，初始参数 θ，数值稳定量 δ，一阶动量衰减系数 ρ_1，二阶动量衰减系数 ρ_2。
其中：几个取值一般为：$\delta = 10^{-8}$，$\rho_1 = 0.9$，$\rho_2 = 0.999$

中间变量: 一阶动量 s，二阶动量r (初始化都为0)
每步迭代过程:

1. 从训练集中的随机抽取一批容量为m的样本 $\{x_1, \ldots, x_m\}$，以及相关的输出 y_i

2. 计算梯度和误差，更新 r 和 s，再根据 r 和 s 以及梯度计算参数更新量

$$
\begin{aligned}
g &\leftarrow +\frac{1}{m} \nabla_\theta \sum_i L(f(x_i;\theta), y_i) \\
s &\leftarrow \rho_1 s + (1-\rho_1) g \\
r &\leftarrow \rho_2 r + (1-\rho_2) g \odot g \\
\hat{s} &\leftarrow \frac{s}{1-\rho_1} \\
\hat{r} &\leftarrow \frac{r}{1-\rho_2} \\
\Delta\theta &= -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta} \\
\theta &\leftarrow \theta + \triangle\theta
\end{aligned}
$$