* **概述**

在神经网络的训练中，有两个重要的概念，一个是损失函数，一个是优化函数，简单来说损失函数是评价指标，优化函数是网络的优化策略。

为了更高效的优化网络结构（损失函数最小），我们需要最恰当的优化函数来优化网络，常用的优化函数有 SGD、BGD、MBGD、Momentum、NAG、Adagrad、Adadelta，RMSprop、Adam，其中：

1. SGD、BGD、MBGD 三个算法都是使用梯度下降算法对网络权重进行更新，只是每次更新使用的样本数量不同。（ SGD属于离线学习，BGD、MBGD属于在线学习）
2. Momentum和NAG算法是根据局部历史梯度对当前梯度进行平滑。
3. Adagrad，Adadelta，RMSprop算法都是自适应学习率的优化算法，对于不同参数使用不同的自适应学习率；Adagrad使用梯度平方和、Adadelta，RMSprop使用梯度一阶指数平滑（一阶指数平均，局部加权历史梯度）解决了Adagrad后期梯度非常小的问题；RMSprop是Adadelta的一种特殊形式。
4. Adam算法吸收了Momentum和RMSprop算法的优点，同时改进了梯度计算方式和学习率，本质上是带有动量项的RMSprop。

* **SGD（Stochastic Gradient Descent 随机梯度下降）**

SGD随机梯度下降参数更新原则：单条数据就可对参数进行一次更新。每个epoch参数更新M（样本数）次，

这里的随机是指每次选取哪个样本是随机的，每个epoch样本更新的顺序是随机的

* 优点：参数更新速度快。
* 缺点：由于每次参数更新时采用的数据量很小，造成梯度更新时震荡幅度大，容易受到异常值的影响，在最优解附近会有加大波动，但大多数情况都是向着梯度减小的方向。

注：离线学习就是使用随机梯度下降算法对模型进行更新，对于每一个样本都计算一次梯度并更新模型

* **BGD（Batch Gradient Descent 批量梯度下降）**

BGD批量梯度下降参数更新原则：每次将所有样本的梯度求和，然后根据梯度和对参数进行更新，每个epoch参数更新1次。

* 优点：由于每次参数更新时采用的数据量很大，梯度更新时很平滑。
* 缺点：由于每次参数更新时采用的数据量很大，造成参数更新速度慢，内存消耗大，因为梯度更新平滑随机性差，容易陷入局部最优解。
* **MBGD（Mini-Batch Gradient Descent 小批量梯度下降）**

MBGD小批量梯度下降参数更新原则：只用所有数据的一部分进行参数的每一次更新。本质上就是在每个batch内部使用BGD策略，在batch外部使用SGD策略。

优点：相比于SGD，由于每次参数更新时采用的数据量相对较大，梯度更新时相对平滑；相比于BGD，由于每次参数更新时采用的数据量相对较小，参数更新速度相对较快。

缺点：没有考虑数据集的稀疏度和模型的训练时间对参数更新的影响。

注：MBGD优化器对每个batch内所有样本的梯度求均值，然后使用梯度的均值更新模型，因此使用MBGD优化器进行更新属于离线学习，同理BGD也属于离线学习

* **Momentum（动量梯度下降）**

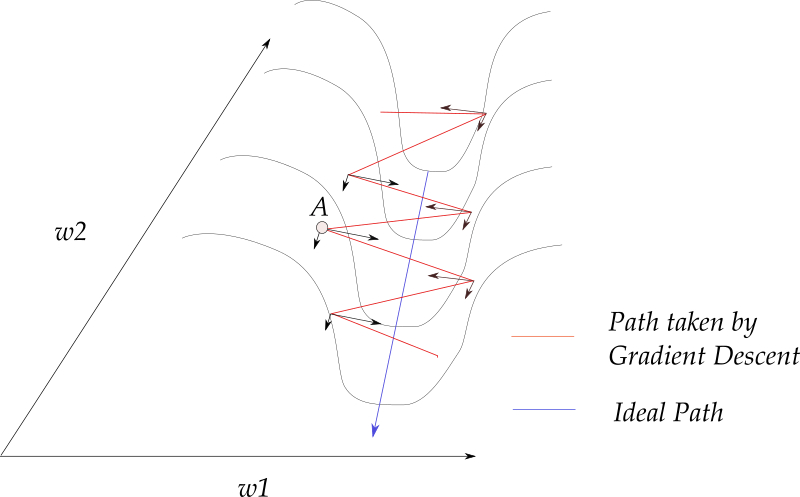
Momentum梯度下降算法在与原有梯度下降算法的基础上，引入了动量的概念，使网络参数更新时的方向会受到前一次梯度方向的影响，换句话说，每次梯度更新都会带有前几次梯度方向的惯性，使梯度的变化更加平滑，这一点上类似一阶马尔科夫假设；Momentum梯度下降算法能够在一定程度上减小权重优化过程中的震荡问题。

引入动量的具体方式是：通过计算梯度的指数加权平均数来积累之前的动量，进而替代真正的梯度，Momentum的优化函数的权重更新公式如下：



通过上述公式可以在，动量参数v本质上就是到目前为止所有历史梯度值的加权平均，距离越远，权重越小。

通过下面这张图片来理解动量法如何在优化过程中减少震荡



上图中每次权重更新的方向可以分解成W1方向上的w1分量和w2方向上的w2分量，红线是传统的梯度下降算法权重更新的路径，在w1方向上会有较大的震荡；

如果使用动量梯度下降算法，算法每次会累计之前的梯度的值。以A点为例，在A点的动量是A点之前所有点的梯度的加权平均和，这样会很大程度抵消A点在w1上的分量，使A点在w2方向上获得较大的分量，从而使A点沿蓝色路径（理想路径）进行权重更新。

* 优点：加入的这一项，可以使得梯度方向不变的维度上速度变快，梯度方向有所改变的维度上的更新速度变慢，这样就可以加快收敛并减小震荡。
* 缺点：这种情况相当于小球从山上滚下来时是在盲目地沿着坡滚，如果它能具备一些先知，例如快要到坡底时，就知道需要减速了的话，适应性会更好。
* **NAG（Nesterov Accelerated Gradient 牛顿动量梯度下降）**

Nesterov Accelerated Gradient （牛顿动量梯度下降） 算法是Momentum算法的改进，原始公式如下：

在这里插入图片描述

通过变换我们可以得到上边公式的等价形式：

在这里插入图片描述

NAG算法会根据此次梯度（i-1）和上一次梯度（i-2）的差值对Momentum算法得到的梯度进行修正，如果两次梯度的差值为正，证明梯度再增加，我们有理由相信下一个梯度会继续变大；相反两次梯度的差值为负，我们有理由相信下一个梯度会继续变小。

如果说Momentum算法是用一阶指数平滑，那么NGA算法则是使用了二阶指数平滑；Momentum算法类似用已得到的前一个梯度数据对当前梯度进行修正（类似一阶马尔科夫假设），NGA算法类似用已得到的前两个梯度对当前梯度进行修正（类似二阶马尔科夫假设），无疑后者得到的梯度更加准确，因此提高了算法的优化速度。

* 应用：NAG 可以使 RNN 在很多任务上有更好的表现。
* 缺点：不能根据参数的重要性而对不同的参数进行不同程度的更新。
* **Adagrad（Adaptive gradient algorithm 自适学习率应梯度下降）**

Adagrad 算法解决的问题：算法不能根据参数的重要性而对不同的参数进行不同程度的更新的问题。

在基本的梯度下降法优化中，有个一个常见问题是，要优化的变量对于目标函数的依赖是各不相同的。Adagrad算法能够在训练中自动的对学习率进行调整，对于出现频率较低参数采用较大的α更新（出现次数多表明参数波动较大，）；相反，对于出现频率较高的参数采用较小的学习率更新，具体来说，每个参数的学习率反比于其历史梯度平方值总和的平方根。因此，Adagrad非常适合处理稀疏数据。

Adagrad其实是对学习率进行了一个约束即：

文本, 信件

描述已自动生成

注：符号E[] 表示期望

特点：

* 前期 gt 较小的时候， 分母较大，能够放大梯度
* 后期 gt 较大的时候， 分母较小，能够约束梯度
* 适合处理稀疏梯度

缺点：

* 由公式可以看出，仍依赖于人工设置一个全局学习率
* η 设置过大的话，会使分母过于敏感，对梯度的调节太大
* 中后期，分母上梯度平方的累加将会越来越大，使 Δθt ->0，使得训练提前结束
* **RMSprop（root mean square prop）**

RMSprop解决的问题：解决Adagrad分母会不断积累，这样学习率就会收缩并最终会变得非常小的问题。  
RMSprop参数更新原则：RMSprop在Adagrad算法的基础上对进行了一阶指数平滑处理  
RMSprop算法的公式如下：

文本

描述已自动生成

我们可以使用均方根（RMS）公式简化上述过程：

图片包含 表格

描述已自动生成

特点：

* RMSprop依然依赖于全局学习率
* RMSprop算是Adagrad的发展和Adadelta的变体，效果趋于二者之间
* 适合处理非平稳目标 - 对于RNN效果很好
* **AdaDelta**

RMSprop优化器虽然可以对不同的权重参数自适应的改变学习率，但仍要指定超参数η，AdaDelta优化器对RMSProp算法进一步优化：AdaDelta算法额外维护一个状态变量，并使用代替 RMSprop 中的学习率参数η，使AdaDelta优化器不需要指定超参数η

AdaDelta 的计算公式如下：

文本

中度可信度描述已自动生成

文本

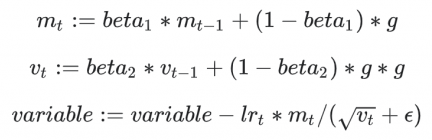
描述已自动生成

特点：

* 训练初中期，加速效果不错，很快
* 训练后期，反复在局部最小值附近抖动
* **Adam**

Adma吸收了Adagrad（自适应学习率的梯度下降算法）和动量梯度下降算法的优点，既能适应稀疏梯度（即自然语言和计算机视觉问题），又能缓解梯度震荡的问题。

Adma的公式如下，这里我们主要分析红色的标记的这3行公式：



公式1：计算 **历史梯度的一阶指数平滑值**，用于得到带有动量的梯度值  
公式2：计算 **历史梯度平方的一阶指数平滑值**，用于得到每个权重参数的学习率权重参数  
公式3：计算变量更新值，由公式3可知，**变量更新值正比于历史梯度的一阶指数平滑值，反比于历史梯度平方的一阶指数平滑值；**

* **历史梯度的一阶指数平滑**

历史梯度的一阶指数平滑（也叫一阶矩加权均值）可以理解为求历史梯度的加权平均，距离当前时刻距离越远权重越小，这里类似于求历史梯度的L1范数，这个思想来自于**动量梯度下降算法**

如果权重参数更新时，某一维度存在较大震荡（梯度更新值正负交替），那么对当前梯度使用一阶指数平滑（对历史梯度进行正负抵消），可以得到一个较小的梯度值

如果变量在某一维度更新时不存在较大震荡，且在在某一方向上存在稳定较大的梯度更新，那么通过一阶指数平滑后会得到一个较大的值。

总的来说，历史梯度的一阶指数平滑就是使用历史梯度信息对当前梯度进行一个修正（消除变量更新时的震荡），从而得到一个稳定的梯度更新值

* **历史梯度平方的一阶指数平滑**

历史梯度平方的一阶指数平滑（也叫二阶矩加权均值）可以理解为求历史梯度平方的加权平均，距离当前时刻距离越远权重越小，这里类似于求历史梯度的L2范数，这个思想来自于自适应学习率梯度下降算法

因为历史梯度平方的一阶指数平滑的作用就是使每个参数的学习率不同，因此这里我们将学习率参数η和看作一个整体，这里我们简单称他为自适应学习率

在迭代过程中，如果某一维度一直以很小的梯度进行更新，证明此方向梯度变换较为稳定，因此可以加大学习率，以较大的学习率在此维度更新，体现在公式上就是：对历史梯度平方进行一阶指数平滑后，公式2会得到一个很小的值，公式3中的自适应学习率会相对较小

相反，某一维度在迭代过程中一直以很大的梯度进行更新，明此方向梯度变换较为剧烈（不稳定），因此可减小学习率，以较小的学习率在此维度更新 体现在公式上就是：对历史梯度平方进行一阶指数平滑后，公式2则会得到一个很大的值，公式3中的自适应学习率会相对较大

**历史梯度平方的一阶指数平滑**也可以解决**梯度稀疏**的问题；频繁更新的梯度将会被赋予一个较小的学习率，而稀疏的梯度则会被赋予一个较大的学习率，通过上述机制，在数据分布稀疏的场景，能更好利用稀疏梯度的信息，比标准的SGD算法更有效地收敛。

**综上所述，Adam 优化器可以根据历史梯度的震荡情况和过滤震荡后的真实历史梯度对变量进行更新**