# 一个框架看懂优化算法

https://zhuanlan.zhihu.com/p/32230623

https://zhuanlan.zhihu.com/p/32338983

### 一个框架回顾优化算法

SGD -> SGDM -> NAG ->AdaGrad -> AdaDelta -> Adam -> Nadam

首先定义：待优化参数: w, 目标函数f(w),初始化学习率.

然后，开始迭代优化，每个epoch t：

1. 计算目标函数关于当前参数的梯度
2. 根据历史梯度计算一阶动量和二阶动量：

；

1. 计算当前时刻的下降梯度：
2. 根据下降梯度进行更新：

### SGD

下降梯度：

SGD最大的缺点是下降速度慢，而且可能会在沟壑的两边持续震荡，停留在一个局部最优点。

### SGD with momentum

为了抑制SGD的震荡，SGDM认为梯度下降过程可以加入惯性。下坡的时候，如果发现是陡坡，那就利用惯性跑的快一些。SGDM全称是SGD with momentum，在SGD基础上引入了一阶动量：;

一阶动量是各个时刻梯度方向的指数移动平均值，约等于最近 个时刻的梯度向量和的平均值。

### SGD with Nesterov Acceleration

SGD 还有一个问题是困在局部最优的沟壑里面震荡。想象一下你走到一个盆地，四周都是略高的小山，你觉得没有下坡的方向，那就只能待在这里了。可是如果你爬上高地，就会发现外面的世界还很广阔。因此，我们不能停留在当前位置去观察未来的方向，而要向前一步、多看一步、看远一些。

NAG全称Nesterov Accelerated Gradient，是在SGD、SGD-M的基础上的进一步改进，改进点在于步骤1。我们知道在时刻t的主要下降方向是由累积动量决定的，自己的梯度方向说了也不算，那与其看当前梯度方向，**不如先看看如果跟着累积动量走了一步，那个时候再怎么走。**因此，NAG在步骤1，不计算当前位置的梯度方向，而是计算如果按照累积动量走了一步，那个时候的下降方向：; 然后用下一个点的梯度方向，与历史累积动量相结合，计算步骤2中当前时刻的累积动量。

### AdaGrad

此前我们都没有用到二阶动量。二阶动量的出现，才意味着“自适应学习率”优化算法时代的到来。SGD及其变种以同样的学习率更新每个参数，但深度神经网络往往包含大量的参数，这些参数并不是总会用得到。对于经常更新的参数，我们已经积累了大量关于它的知识，不希望被单个样本影响太大，希望学习速率慢一些；对于偶尔更新的参数，我们了解的信息太少，希望能从每个偶然出现的样本身上多学一些，即学习速率大一些。

怎么样去度量历史更新频率呢？那就是二阶动量——该维度上，迄今为止所有梯度值的平方和：;

此时实质上的学习率由  变成了  。 一般为了避免分母为0，会在分母上加一个小的平滑项。因此 是恒大于0的，而且参数更新越频繁，二阶动量越大，学习率就越小。

这一方法在稀疏数据场景下表现非常好。但也存在一些问题：因为 是单调递的，会使得学习率单调递减至0，可能会使得训练过程提前结束，即便后续还有数据也无法学到必要的知识。

### AdaDelta / RMSProp

由于AdaGrad单调递减的学习率变化过于激进，我们考虑一个改变二阶动量计算方法的策略：不累积全部历史梯度，而只关注过去一段时间窗口的下降梯度。这也就是AdaDelta名称中Delta的来历。

修改的思路很简单。前面我们讲到，指数移动平均值大约就是过去一段时间的平均值，因此我们用这一方法来计算二阶累积动量：

这就避免了二阶动量持续累积、导致训练过程提前结束的问题了。

### Adam

谈到这里，Adam和Nadam的出现就很自然而然了——它们是前述方法的集大成者。我们看到，SGD-M在SGD基础上增加了一阶动量，AdaGrad和AdaDelta在SGD基础上增加了二阶动量。把一阶动量和二阶动量都用起来，就是Adam了——Adaptive + Momentum。

SGD的一阶动量：

AdaDelta的二阶动量：

* **指数移动平均值的偏差修正：**前面公式当初始化时候，初期，都接近0，所以我们进行修正：



# 优化算法的选择与使用策略

### 不同优化算法的核心差异：下降方向

在第三步执行的下降方向是不同算法的核心：



这个式子中，**前半部分**是实际的学习率（也即**下降步长**），**后半部分**是实际的**下降方向**。**SGD算法**的下降方向就是该位置的梯度方向的**反方向**，带**一阶动量的SGD**的下降方向则是该位置的**一阶动量方向**。自适应学习率类优化算法为每个参数设定了不同的学习率，在不同维度上设定不同步长，因此其下降方向是缩放过（scaled）的一阶动量方向。

### Adam+SGD 组合策略

不同优化算法的优劣依然是未有定论的争议话题。据我在paper和各类社区看到的反馈，主流的观点认为：**Adam**等自适应学习率算法对于**稀疏数据**具有优势，且收敛速度很快；但**精调**参数的**SGD**（+Momentum）往往能够取得更好的最终结果。

**两个问题**

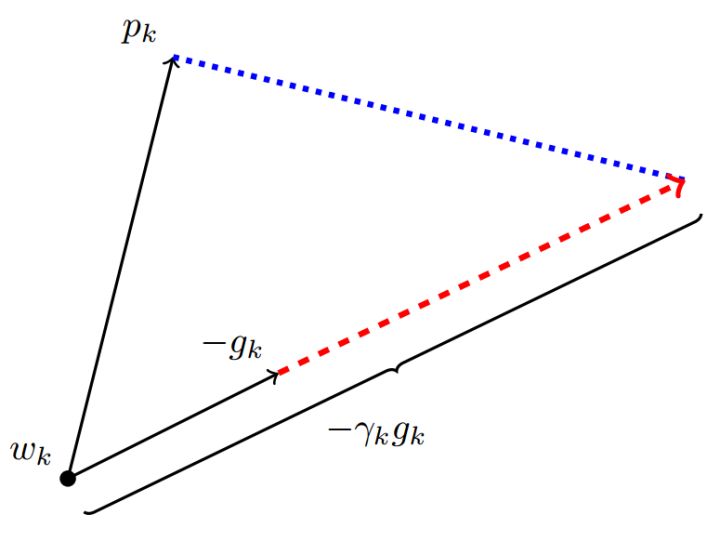
**（1） 什么时候切换优化算法？**——如果切换太晚，Adam可能已经跑到自己的盆地里去了，SGD再怎么好也跑不出来了。

**（2） 切换算法以后用什么样的学习率？**——Adam用的是自适应学习率，依赖的是二阶动量的累积，SGD接着训练的话，用什么样的学习率？

**首先来看第二个问题，切换之后用什么样的学习率**。Adam的下降方向是

而SGD：，

必定可以分解为所在方向及其正交方向之和，那么其在方向上的投影就意味着SGD在Adam算法决定的下降方向上前进的距离，而在的正交方向上的投影是SGD在自己选择的修正方向上前进的距离。



P为Adam下降方向，g为梯度方向，r为SGD的学习率

如果SGD要走完Adam未走完的路，那就首先要接过Adam的大旗——沿着  方向走一步，而后在沿着其正交方向走相应的一步。

这样我们就知道该如何确定SGD的步长（学习率）了——S**GD在Adam下降方向上的正交投影，应该正好等于Adam的下降方向（含步长）**。也即：

解这个方程，我们就可以得到接续进行SGD的学习率：

为了减少噪声影响，作者使用移动平均来修正对学习率的估计





**然后来看第一个问题，何时进行算法的切换。**的时候；

### 优化算法的常用tricks

* **各大算法孰优孰劣并无定论**，如果刚入门，选择SGD+Nesterov Momentum或者Adam。
* **选择你熟悉的算法**——这样你可以更加熟练地利用你的经验调参
* **充分了解你的数据**——如果模型是非常稀疏的，那么优先考虑自适应学习率算法。
* **根据你的需求来选择**——在模型设计实验过程中，要快速验证新模型的效果，可以先用Adam进行快速实验优化，在模型上线或者结果发布前，可以用精调的SGD进行模型极致的优化
* **先用小数据即进行实验。**有论文指出，随机梯度下降算法的收敛速度和数据集的大小关系不大。因此先用一个具有代表性的小数据进行实验，测试一下最好的优化算法，并通过参数所有来寻找最优的训练参数；
* **考虑不同算法的组合。**先用Adam进行快速下降，而后再换SGD进行充分的调优。
* **数据集一定要充分打散。**这样使用自适应学习率算法的时候，可以避免某些特征集中出现，而导致的有时学习过度，有时学习不足，使得梯度方向出现偏差问题。
* **训练过程中持续监控训练数据和验证数据上的目标函数值以及精度或者AUC等指标的变化情况**。对训练数据的监控室要保证模型进行了充分的训练——梯度下降方向，且学习率足够高；对验证数据的监控是为了避免过拟合。
* **制定一个合适的学习率衰减策略**。可以使用定期衰减策略，比如美国多少个epoch就衰减一次，或者利用精度或者AUC等性能指标来控制，当测试集上的指标不变或者下跌时，就降低学习率。