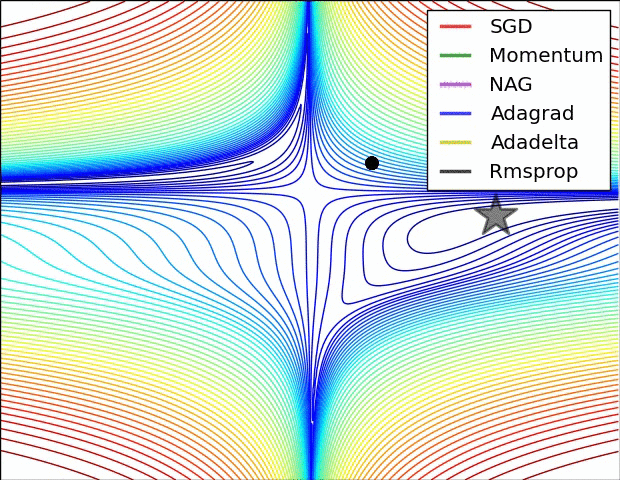
**最清晰的讲解各种梯度下降法原理与Dropout**



一、梯度法思想

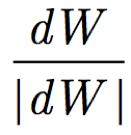
梯度法思想的三要素：出发点、下降方向、下降步长。

机器学习中常用的权重更新表达式为

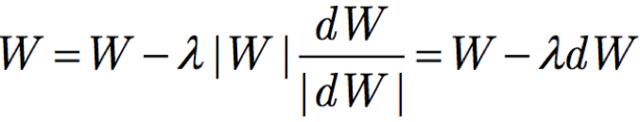
：，这里的λ就是学习率，本文从这个式子出发来把机器学习中的各种“梯度”下降法阐释清楚。

机器学习目标函数，一般都是凸函数，什么叫凸函数？限于篇幅，我们不做很深的展开，在这儿我们做一个形象的比喻，凸函数求解问题，可以把目标损失函数想象成一口锅，来找到这个锅的锅底。非常直观的想法就是，我们沿着初始某个点的函数的梯度方向往下走（即梯度下降）。在这儿，我们再作个形象的类比，如果把这个走法类比为力，那么完整的三要素就是步长（走多少）、方向、出发点，这样形象的比喻，让我们对梯度问题的解决豁然开朗，出发点很重要，是初始化时重点要考虑的，而方向、步长就是关键。事实上不同梯度的不同就在于这两点的不同！

梯度方向是



，步长设为常数Δ，这时就会发现，如果用在梯度较大的时候，离最优解比较远，W的更新比较快；然而到了梯度较小的时候，也就是较靠近最优解的时候，W的更新竟然也保持着跟原来一样的速率，这样会导致W很容易更新过度反而远离了最优解，进而出现在最优解附近来回震荡。所以，既然在远离最优解的时候梯度大，在靠近最优解的时候梯度小，我们让步长随着这个律动，于是我我们就用λ|W|来代替Δ，最后得到了我们熟悉的式子：



所以说这时的λ是随着坡度的陡缓而变化的，别看它是个常数。

二、全量梯度下降法（Batch gradient descent）

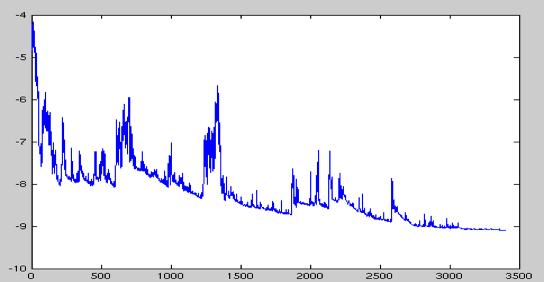
全量梯度下降法每次学习都使用整个训练集，因此每次更新都会朝着正确的方向进行，最后能够保证收敛于极值点，凸函数收敛于全局极值点，非凸函数可能会收敛于局部极值点，缺陷就是学习时间太长，消耗大量内存。

第二、随机梯度下降法（Stochastic Gradient Descent）

SGD一轮迭代只用一条随机选取的数据，尽管SGD的迭代次数比BGD大很多，但一次学习时间非常快。

SGD的缺点在于每次更新可能并不会按照正确的方向进行，参数更新具有高方差，从而导致损失函数剧烈波动。不过，如果目标函数有盆地区域，SGD会使优化的方向从当前的局部极小值点跳到另一个更好的局部极小值点，这样对于非凸函数，可能最终收敛于一个较好的局部极值点，甚至全局极值点。

缺点是，出现损失函数波动，如下图所示，并且无法判断是否收敛。



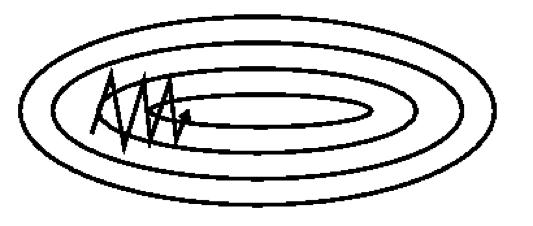
第三、小批量梯度下降法（Mini-Batch Gradient Descent）

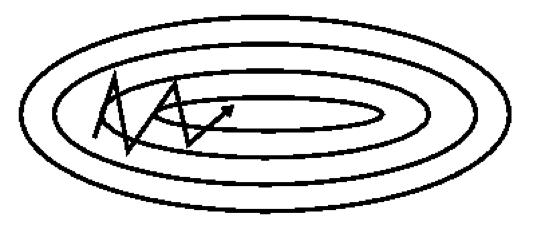
SGD相比BGD收敛速度快，然而，它也的缺点，那就是收敛时浮动，不稳定，在最优解附近波动，难以判断是否已经收敛。这时折中的算法小批量梯度下降法，MBGD就产生了，道理很简单，SGD太极端，一次一条，为何不多几条？MBGD就是用一次迭代多条数据的方法。

并且如果Batch Size选择合理，不仅收敛速度比SGD更快、更稳定，而且在最优解附近的跳动也不会很大，甚至得到比Batch Gradient Descent 更好的解。这样就综合了SGD和Batch Gradient Descent 的优点，同时弱化了缺点。总之，Mini-Batch比SGD和Batch Gradient Descent都好。

第四、Momentum梯度下降法

SGD、BSGD两种改进方法都存在不同程度的震荡，如何避免震荡？或者说震荡是怎么产生的？震荡，从可视图表现来看，就是频繁更改方向，所以，如果能够把之前下降的方向考量进来，那么将会减少振荡。





上边第二个图是带动量的梯度下降法。

下面推导动量下降法。

在普通的梯度下降法W -= V中，每次W的更新量V为V = dW \* λ;

当使用冲量时，V考虑为本次的梯度下降量与部分上次更新量的矢量和，即-dW\*λ与上次x的更新量V乘以一个介于[0, 1]的系数momemtum的和，即:

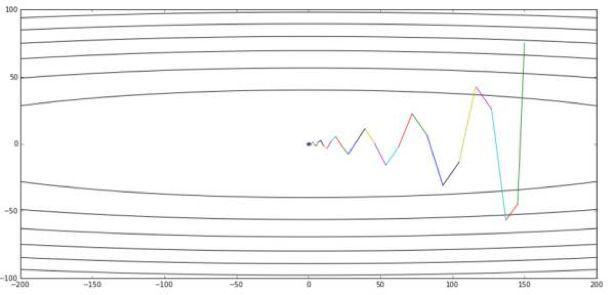
V = dW \* λ+ V\*momemtum。

当本次梯度下降方向与上次更新量的方向相同时，上次的更新量能够对本次的搜索起到一个正向加速的作用。当本次梯度下降方向与上次更新量的方向相反时，上次的更新量能够对本次的搜索起到一个减速的作用。下面两个图，第一图是没有动量的梯度下降法，第二图是有动量的梯度下降法，很明显减少了震荡。

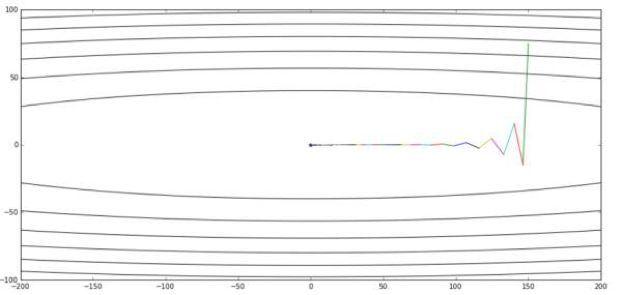
既减少震荡，又大方向不变（本次梯度下降方向），从而保证了效率和正确的收敛。

第五、NAG梯度下降法

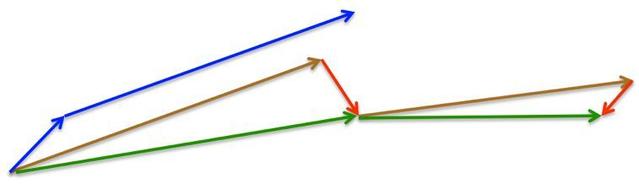
NAG（Nesterov Accelerated Gradient）不仅仅把SGD梯度下降以前的方向考虑，还将Momentum梯度变化的幅度也考虑了进来。



上图是Momentum的优化轨迹，下图是NAG的优化轨迹:



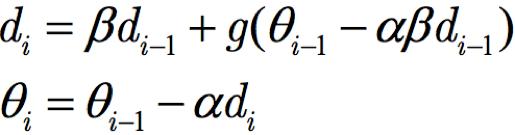
在给出公式之前，先看一个向量图：



这个图来源于 G. Hinton’s的文章，网上对这个图的解释大都来源于《An overview of gradient descent optimizationalgorithms》中文翻译，其实并没有真正讲解清楚。本文严格按照愿意解释一下：

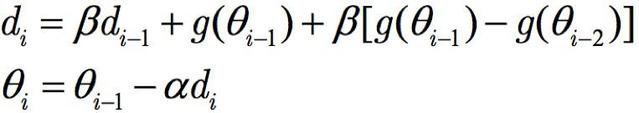
Momentum梯度法首先计算的是当前的梯度（图中的小蓝色向量）然后沿着更新的累积梯度的方向来一个大的跳跃（图中大蓝色向量），而NAG梯度法首先沿着先前的累积梯度方向（棕色向量）实现一个大的跳跃，然后加上一个小的按照动量梯度法计算的当前梯度（上图红色向量）进行修正得到上图绿色的向量。此处我抛出一个问题，上图为什么画了两个三角形？如果能理解第二个矢量三解形的意义，才能正在理解NAG。注意第二个矢量三角形的棕色向量与前一个的绿色向量方向一致，因为上一个矢量三角形的结果是绿色向量，而棕色代表的是先前的累积梯度，方向就应该和绿色的一样。然后，再加上当前按照动量梯度法计算出的梯度，就得到第二个三角形的绿色向量。

给出公式表达：



我们先给出类似生活体验的通俗的解释：我们要让算法要前瞻性，提前看到前方的地形梯度，如果前面的梯度比当前位置的梯度大，那我就可以把步子迈得比原来大一些，如果前面的梯度比现在的梯度小，那我就可以把步子迈得小一些。这个大一些、小一些，都是相对于原来不看前方梯度、只看当前位置梯度的情况来说的。

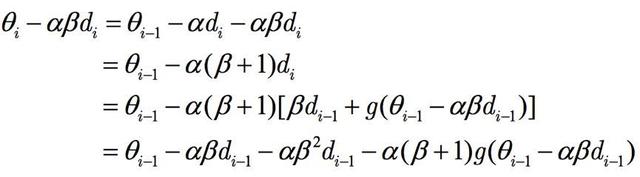
NAG的另一个等效形式为（下文会详细推导）：



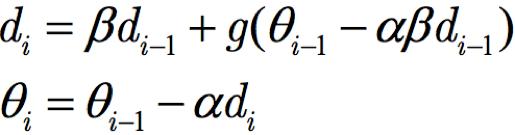
这个NAG的等效形式与Momentum的区别在于，本次更新方向多加了一个，注意到没有，这项其实就是目标函数的二阶导数！所以NAG本质上是多考虑了目标函数的二阶导信息，读过《一篇中矣：彻底理解XGBoost！》一文的读者，是不是觉得很面熟，二阶导数再次发挥作用！其实所谓“往前看”的说法，在牛顿法这样的二阶方法中经常提到，形象说“往前看”，数学思维则是利用了目标函数的二阶导信息。

现在我们由NAG原始形式来推导NAG的等效形式：

为了出现二阶形式，将原始形式按照递推展开：



我们观察这个等式，并对应原始公式



观察对比这个长等式：

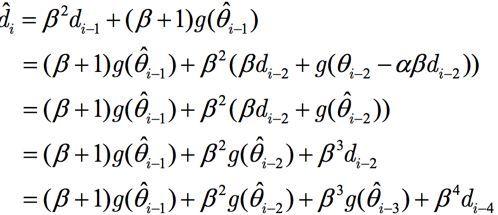
等式相当于参数在更新量更新基础上的更新，所以为了看上去更有意义，对应的形式，我们作如下记号：

我们在猜想一个形式：

则可作记号：

也就是有：

为了变换形式，我们作如下展开：



继续推导：

到此为止NAG证明就结束，逻辑、意义非常完美！这就是数学之美！

限于篇幅其他非常重要的梯度法AdaGrad、AdaDelta、RMSProp、Adam等在后续文章在详加阐述。

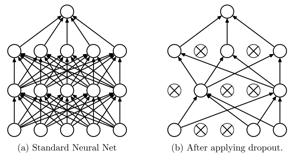
Dropout的正则化原理

1、dropout 如何工作

典型的神经网络其训练流程是将输入通过网络进行正向传导，然后将误差进行反向传播。Dropout就是针对这一过程之中，随机地删除隐藏层的部分单元，进行上述过程。步骤如下：

随机删除网络中的一些隐藏神经元，保持输入输出神经元不变将输入通过修改后的网络进行前向传播，然后将误差通过修改后的网络进行反向传播对于另外一批的训练样本，重复上述操作 训练阶段使用Dropout，测试阶段把Dropout屏蔽。

所以，在训练的时候为每个存活下来的神经元做同采样比例的放大，或者，另外一种选择就是把测试结果根据采样比例缩小。



Dropout之所以具有正则化能力的两个解释：

强迫网络拥有冗余的表示；Dropout在训练一个大规模的网络组合（ensemble）强迫网络拥有冗余的表示：在每轮迭代中，总是随机屏蔽一定比例的神经元，因此输出并不知道它正在组合哪些特征，比起没有Dropout时过于专注于某些特征（比如拿识别猫来说，过于专注于猫的眼睛），现在模型把专注力分散到每个特征，使得这些特征也能具备比较好的预测能力。由于原本具有很高权重的那些特征现在被压缩，达到了正则化的效果。 Dropout相当于模型平均、模型组合：看作是模型平均的一种，平均一个大量不同的网络。不同的网络在不同的情况下过拟合，但是共用一个损失函数，相当于同时进行了优化，取了平均；相比之前使用所有的数据在一个模型里面训练，Dropout相当于在训练阶段每次针对一批数据就生成一个独特的小模型，然后再将这些模型组合起来的一种方法。我们在《透彻理解集成算法及其思想》一文中对于集成模型的优势有很好的阐述。

减少复杂依赖性：由于每次用输入网络的样本进行权值更新时，隐含节点都是以一定概率随机出现，因此不能保证某两个隐含节点每次都同时出现，这样就减少了神经元之间的依赖性，即每个神经元不能依赖于某几个其他的神经元（指层与层之间相连接的神经元），阻止了某些特征仅在有其他特征存在才有效的情况，减少神经元之间复杂的依赖性。