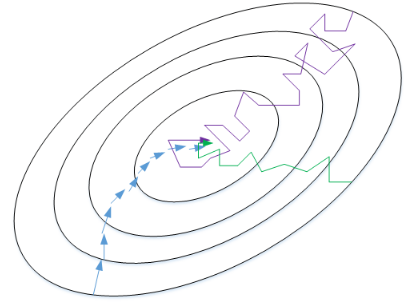
深度学习优化算法

http://blog.csdn.net/red\_stone1/article/details/78348753

Mini-batch gradient，相比batch gradient，能够提高运算速度，兼顾收敛速度，是一种平衡选择

Stachastic-batch，每次训练只有一个样本，虽然运算速度最快（样本处理时间短），但是收敛速度反而比mini-batch底，因为梯度的振荡太大。样本太少，意味着单次训练获取梯度的局部性太强。

每次epoch，最好是将总体训练数据重新打乱、重新分成T组mini-batches，这样有利于训练出最佳的神经网络模型



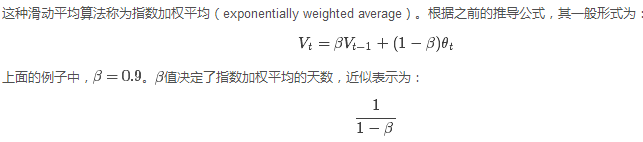
蓝色为batch，运算慢，但是振荡小。完整样本空间获得梯度最为准确。收敛最优需要很多步，是因为学习率限制。蓝色每一步，都是学习率的几倍或者百分之。

绿色为mini-batch，运算速度和收敛速度均衡。

紫色为stachastic-batch，运算速度快，收敛折腾多

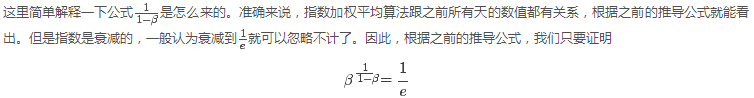
三种优化算法的本质，都是抑制梯度振荡。梯度振荡是影响收敛速度的主要因素

指数加权平均（Exponentially weighted averages）



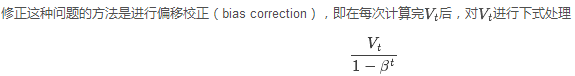
Beta越大，当前数值对于整体平均的影响越小，整体曲线越平滑。反之……

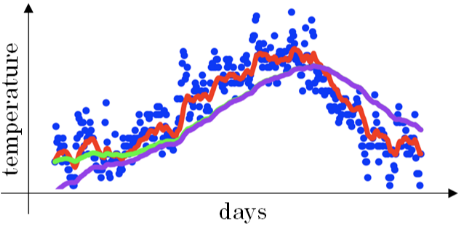
指数加权平均公式的得出过程。关键在于确定多少天之前的数据会衰减到忽略不计



理解指数加权平均。一段时间的数据乘以指数衰减的权重，时间越久，影响越小

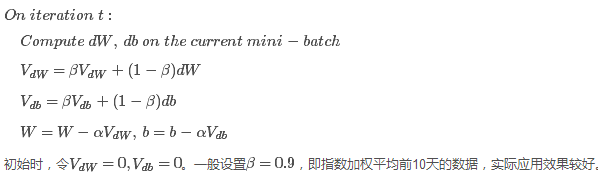
初始阶段，指数加权平均的偏差修正





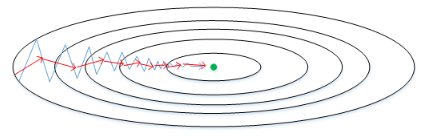
紫色为修正前的，绿色为修正后的

动量梯度下降算法，momentum gradient



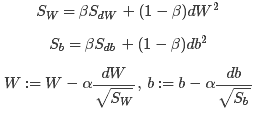
Dw才是当前计算出的梯度。参数的改变，由梯度直接决定，变为梯度的移动平均决定

理解动量梯度算法。有多种说法，但本质都是某次迭代的梯度变化值，不仅与当次计算的梯度有关，还与之前的梯度有关。梯度是渐变而非瞬变的，保证了梯度下降的平稳和准确，减少了震荡



蓝色为普通梯度下降，红色为动量梯度下降

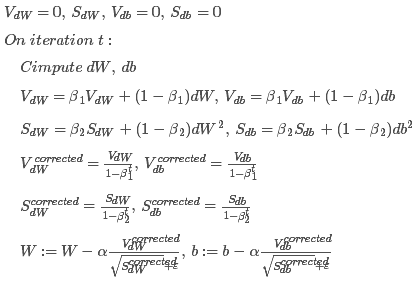
RMSprop，root mean square prop。个人认为SRMprop表述更合理，毕竟平方决定了算法作用



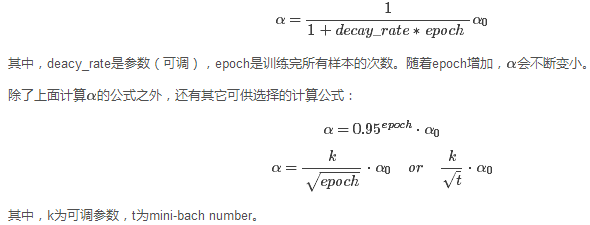
Dw才是当前计算出的梯度。平方针对dw整体

理解RMSprop。梯度振荡是影响收敛速度的主要原因，如果某个参数方向梯度振荡明显，那么其dw平方肯定明显大于其他参数方向。注意只有平方能够表明这一特性，没有平方则来回振荡正负抵消。（借助上图折线，更好认识这一点。将振荡图像化为折线的反复，靠谱）振荡越大分母越大，震荡大的减小明显，振荡小的减小微弱。整体效果上抑制了梯度振荡，加快了收敛速度。

Adam算法，momentum和RMSprop的混合



学习率衰减，随着迭代次数增加，学习因子逐渐减小，避免最优点附近振荡。



局部最优解问题

低维度看，局部最优解是一个个浅坑。而从高维度看，则是一个个鞍面。曲面就只有凹凸两种，局部最优解是凸面上的凹部聚集区，自然就是鞍面。因此，局部最优的实质问题，是鞍面梯度接近于0，难以逃离。RMSprop和adam都能有效解决。

