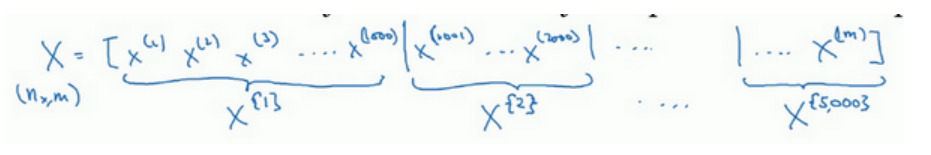
1. mini-batch梯度下降

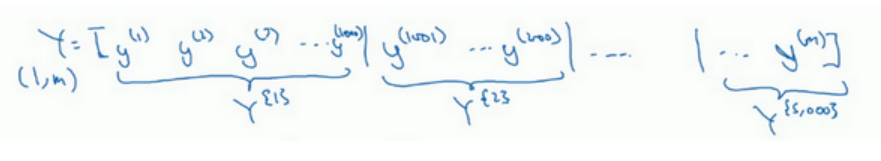
将维数为（n,m）的X划分为小一点的子集，并取名为mini-batch。

假设小一点的子集中有1000个元素，并将其记为

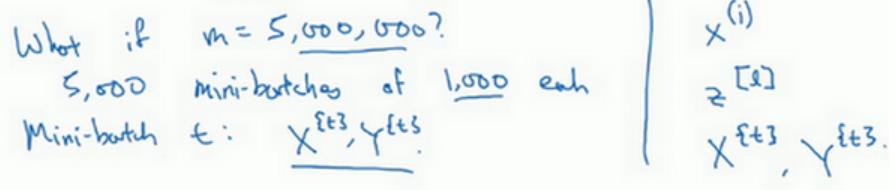
对于整个X，则有



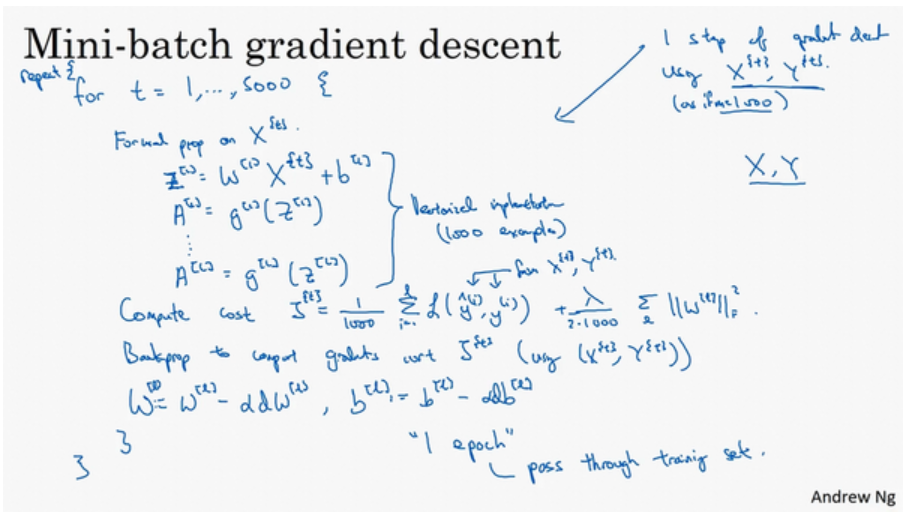
同样将对应的Y划分为同样大小的mini-batch，如下图所示：



mini-batch的数量组成了和，其中分别包含1000个训练样本

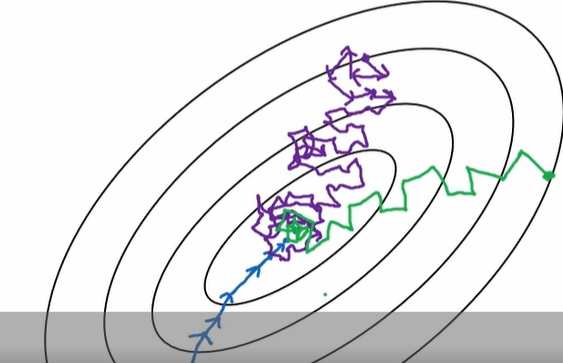


对于mini-batch的单次运算如下图所示：



使用batch梯度下降法，一次遍历训练集只能让你做一个梯度下降，使用mini-batch梯度下降法，一次遍历训练集，能完成5000个梯度下降。在此基础上可以一直遍历训练集，直到找到一个合适的精度。

mini-batch梯度下降与batch梯度下降与随机梯度下降的比较：



1. 指数加权平均数

对于计算指数加权平均数的方程：



vt-1为t-1时刻的估计数值，为t时刻的实际数据，β表示权重

对于示例中给出的这一公式，可以展开并表示为：



上述式子表示第100天的数据是结合了前99天的数据，但这99天的数据所占的权重各不相同，并随着天数的增加而增加。

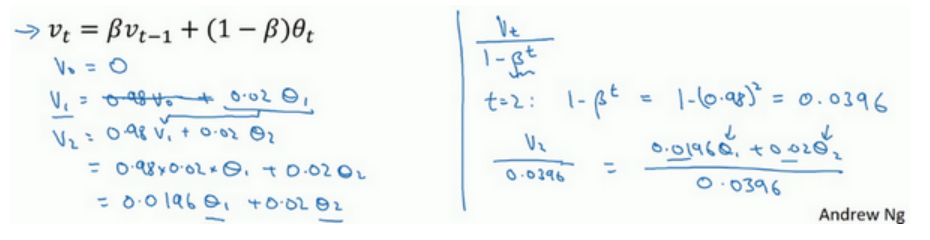
事实上（0.9）^10≈0.35，大约为1/e，这就意味着10天前的数据对于当下数据的影响仅相当于当日权重的1/e，因此，认为当β=0.9时，该指数加权平均数只关注了过去10天的数值。

而对于β=0.98，有（0.98）^50≈1/e，我们认为此时的指数加权平均数关注了过去50天的数值。

1. 偏差矫正

对于计算指数加权平均数的方程，令v0=0，会导致初期的计算值偏差较大，偏差矫正主要是为了解决这一问题。

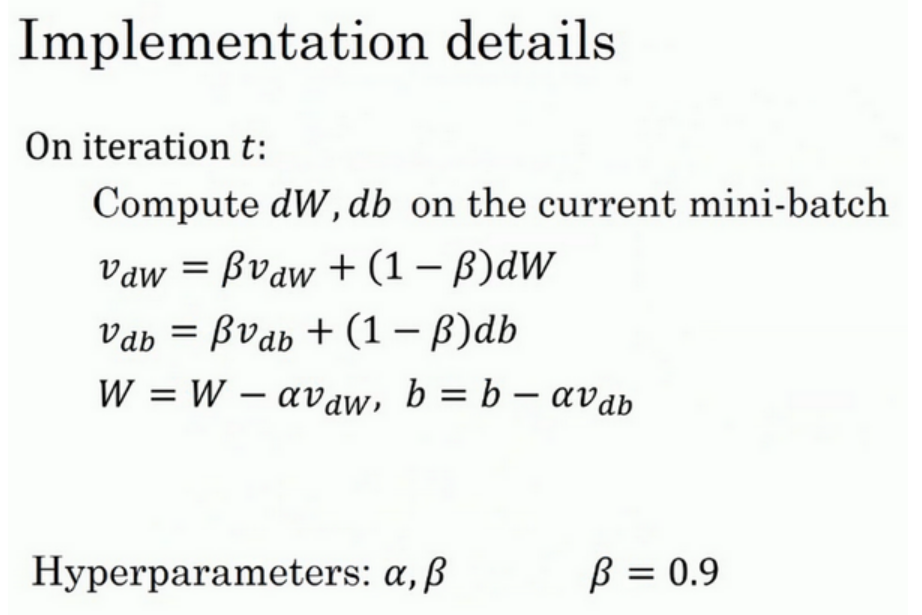
在初期并不采用vt，而是采用vt/(1-β)^t，如下图所示：



1. 动量梯度下降法

运行速度总是快于标准的梯度下降方法，其基本的想法就是计算梯度的指数加权平均数，并利用该梯度更新权重。为了记录方便，暂时将数据维度分为主要方向（有利于梯度下降较快的方向）和非主要方向（对梯度快速下降无关的方向），动量算法会使非主要方向上的变化趋于平缓，主要方向上的变化不受影响。

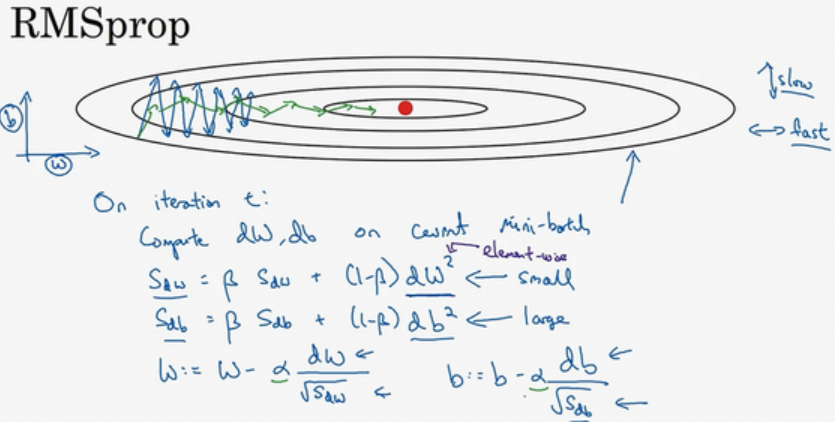
算法伪代码：



其中α为学习率，β为指数加权平均数，一般情况下β=0.9

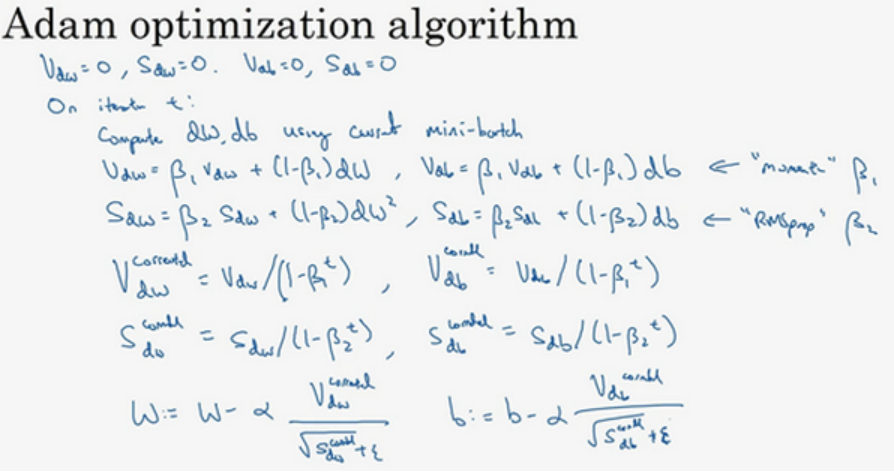
1. RMSprop

与动量算法类似，会减缓非主要方向上的参数变化，而主要方向将不受影响。与动量算法的差异如下图所示：

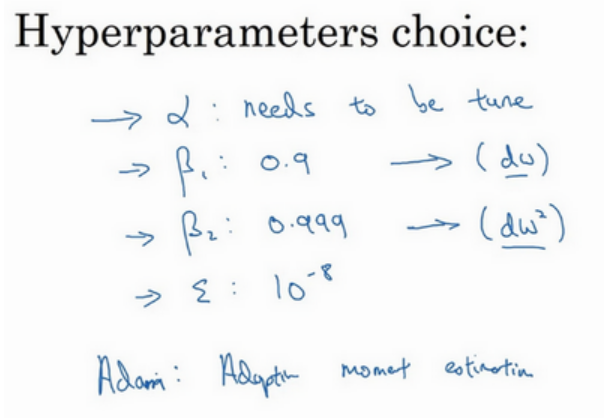


1. adam优化算法

结合了动量算法和RMSprop算法，有非常广泛的实用性。（充分证明了科研就是搭积木）



对于α、β1、β2的参数设置：



1. 学习率衰减

随着时间的推移使得学习率逐渐减少这将有利于代价曲线在最小值附近的更小范围内摆动。如下图所示：

