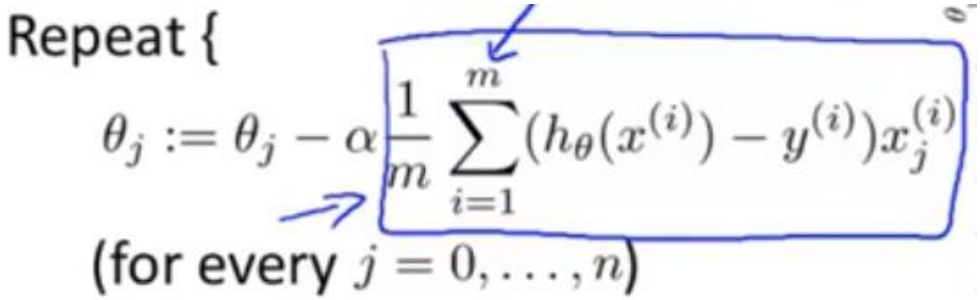
1、引言

现如今的机器学习效果比以往任何时候都要好，原因除了算力的提升，还因为有海量的数据帮助优化目标。然而，当今数据量呈爆发式增长，以往的batch算法性能越发捉襟见肘了，仅在梯度下降算法中的优化目标这一项，就要依次遍历所有的训练样本，数以



亿计的数据是常有的事，所以老旧的算法性能越发的差劲了。

2、随机梯度下降算法

先来介绍一下它的过程，随机意味着要在拿到训练集的时候，将它们打乱顺序，然

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

后进行单样本的梯度下降算法，这是原来的代价函数，下方是随机算法中的代价函数，每

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

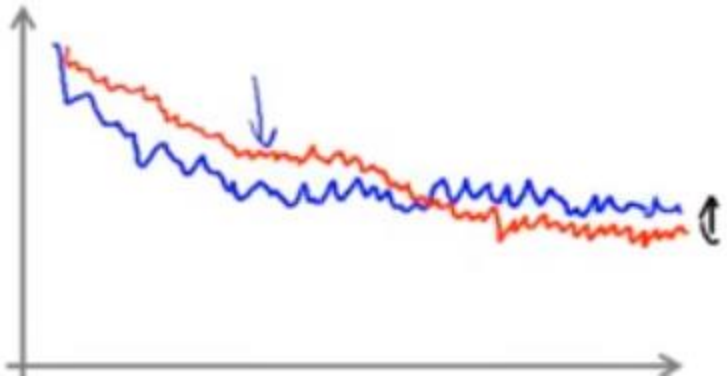
一个样本都会优化一次参数，所以参数更新率远高于普通的梯度下降算法。另外，该算法内部相当于进行了样本容量个优化次数，总体循环次数视样本容量大小而定，比如数据达到1亿了，那么外层循环一次就够了。

3、mini-batch算法

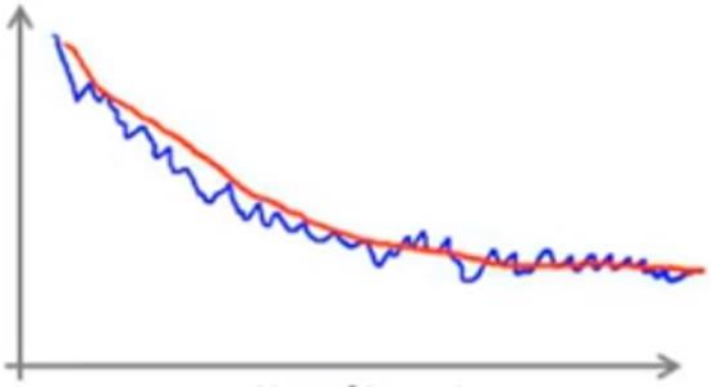
这是一种介于两者之间的算法，一次更新使用10个样本，由于cpu/gpu有良好的向量化计算方式，所以其效率更高。

4、检测结果

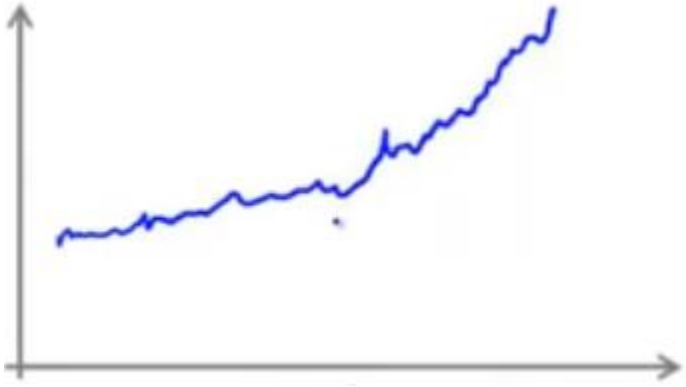
如何判断随机梯度下降算法已经收敛了，或是判断其是否进步了，一个好的做法是，每进行1次优化时，先计算该单样本的代价值，然后又输出优化后的单样本代价值，进行纵向比较。判断是否收敛的做法是，比如每进行1000次优化后，输出最后1000个样本的评价代价值，然后画出图形来。



蓝线的学习率比红线的大，
导致最后收敛的值相对较大



蓝线代表每 1000 次计算 cost，
红线代表每 5000 次计算 cost，
所以红线相比更加平滑



学习率太大，导致代价函数发散

最后讨论关于学习率的问题，在随机梯度下降算法影响下，代价函数呈震荡式的接近于全局最小值，要想达到真正的全局最小值，需要让学习率随迭代次数增加而减小。

随机梯度下降算法还适用于一个特殊场景，就是数据会源源不断进来，比如大型在线网站，每时每刻会有用户做出新的行为，那么通过使用在线学习机制，一次一个样本，过后丢弃，可以使整个预测保持“新鲜”。