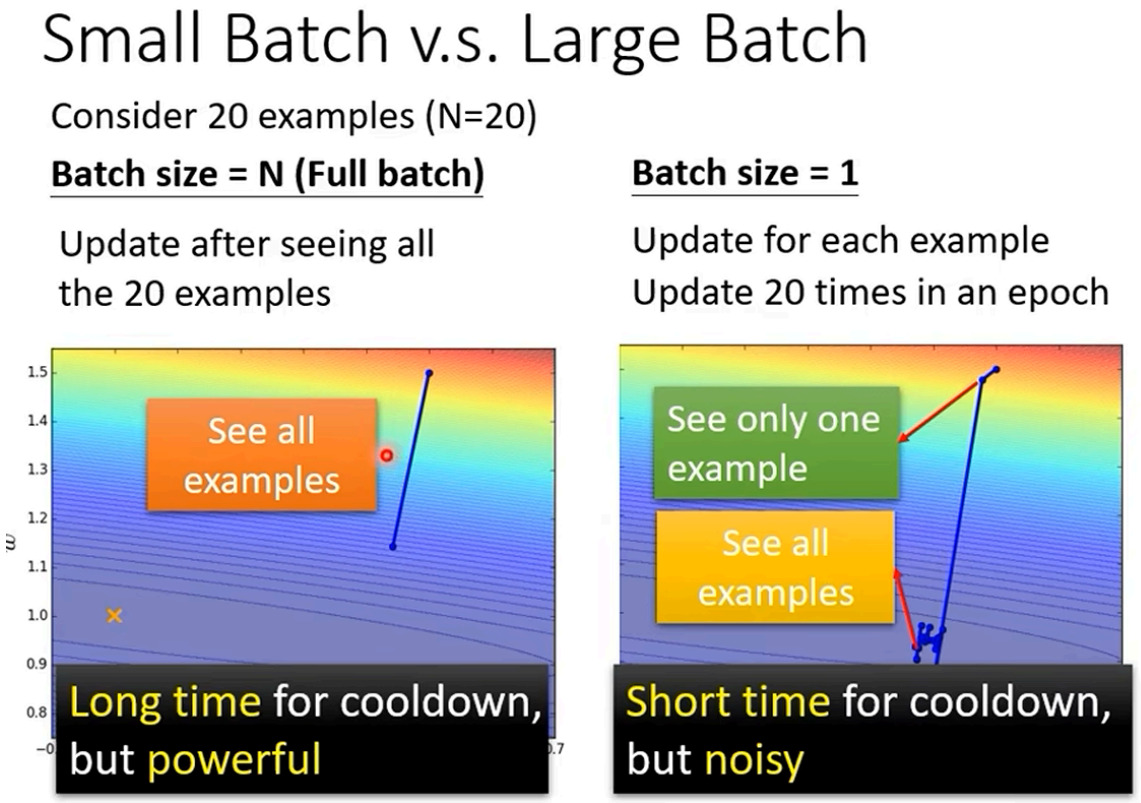
类神经网络训练不起来怎么办（二）批次与动量

一个epoch就是过一遍所有的batch，而shuffle是在每一次epoch之前进行对batch重新分组。

▼ 为什么要用batch：



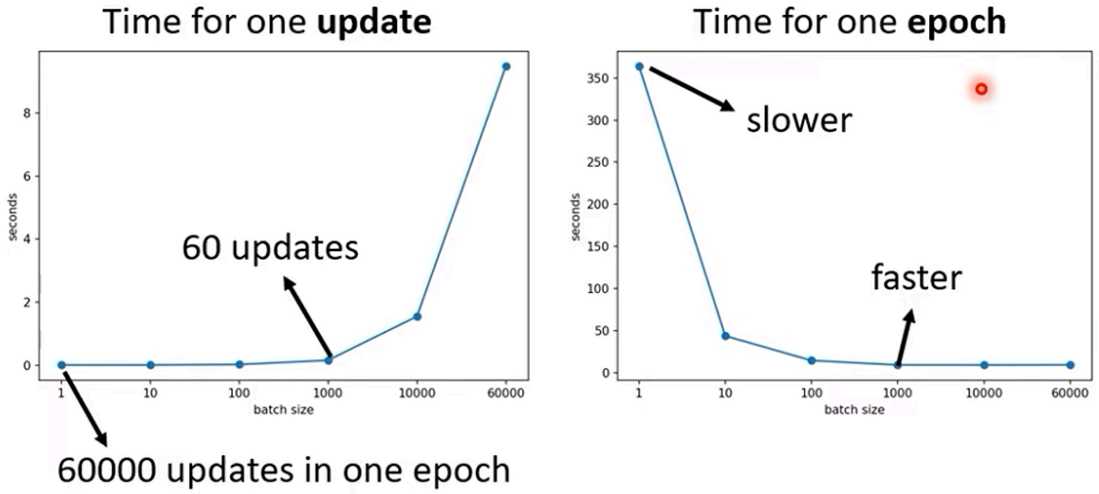
可以看到，左边full batch，他的更新一次参数时间长，但是有力，右边时间短，但是不稳定。但是这样说时间长短是没有考虑并行计算的问题。

考虑上并行计算，大的batch计算所需要的时间不一定比小的batch所需时间短。

但是考虑一个epoch的时间，也是大的batch所需时间更少。

Small Batch v.s. Large Batch

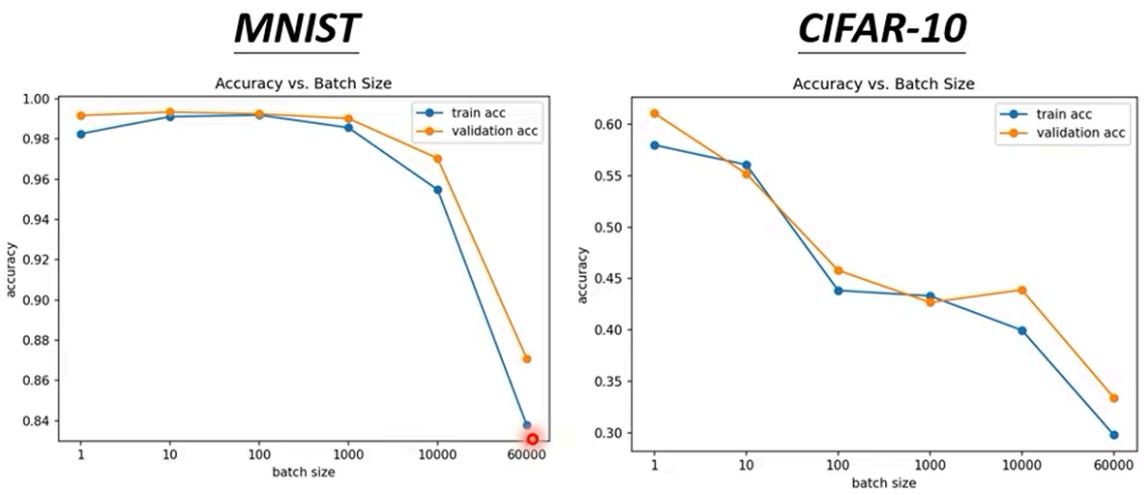
- Smaller batch requires longer time for one epoch (longer time for seeing all data once)



可以看到，update和epoch的时间趋势图正好是相反的。

但是小的batch的noisy可以帮助learning

Small Batch v.s. Large Batch



- Smaller batch size has better performance
- What's wrong with large batch size? Optimization Fails

可以看到，小的batch size的准确率更高一些，这里是优化的问题。同时小的batch在test上表现也更好。

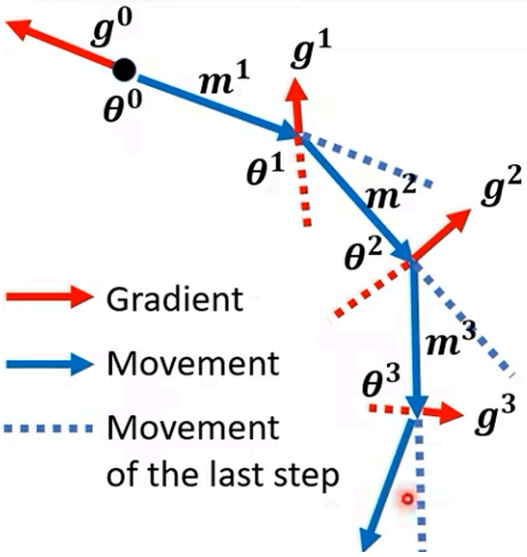
Small Batch v.s. Large Batch

	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster
Gradient	Noisy	Stable
Optimization	Better	Worse
Generalization	Better	Worse

▼ Momentum（动量）

Gradient Descent + Momentum

Movement: **movement of last step** minus **gradient** at present



Starting at $\boldsymbol{\theta^0}$

Movement $\boldsymbol{m^0 = 0}$

Compute gradient $\boldsymbol{g^0}$

Movement $\boldsymbol{m^1} = \lambda \boldsymbol{m^0} - \eta \boldsymbol{g^0}$

Move to $\boldsymbol{\theta^1 = \theta^0 + m^1}$

Compute gradient $\boldsymbol{g^1}$

Movement $\boldsymbol{m^2} = \lambda \boldsymbol{m^1} - \eta \boldsymbol{g^1}$

Move to $\boldsymbol{\theta^2 = \theta^1 + m^2}$

Movement not just based on gradient, but previous movement.

这是我们加上动量之后的参数变化情况，不止只考虑梯度下降的方向，并且要考虑动量。但其实mi可以当作为之前的梯度加权之和。

$\boldsymbol{m^i}$ is the weighted sum of all the previous gradient: $\boldsymbol{g^0}, \boldsymbol{g^1}, \ldots, \boldsymbol{g^{i-1}}$

$$\boldsymbol{m^0} = \mathbf{0}$$

$$\boldsymbol{m^1} = -\eta \boldsymbol{g^0}$$

$$\boldsymbol{m^2} = -\lambda\eta \boldsymbol{g^0} - \eta \boldsymbol{g^1}$$

$$\vdots$$

动量加大了跳过local minima的概率。