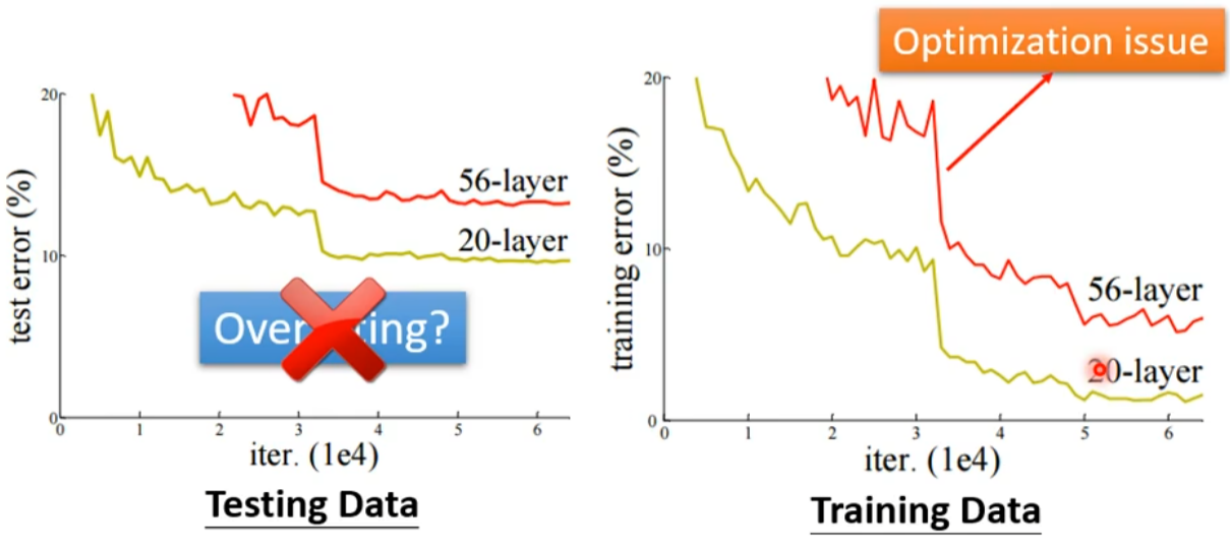
训练模型攻略

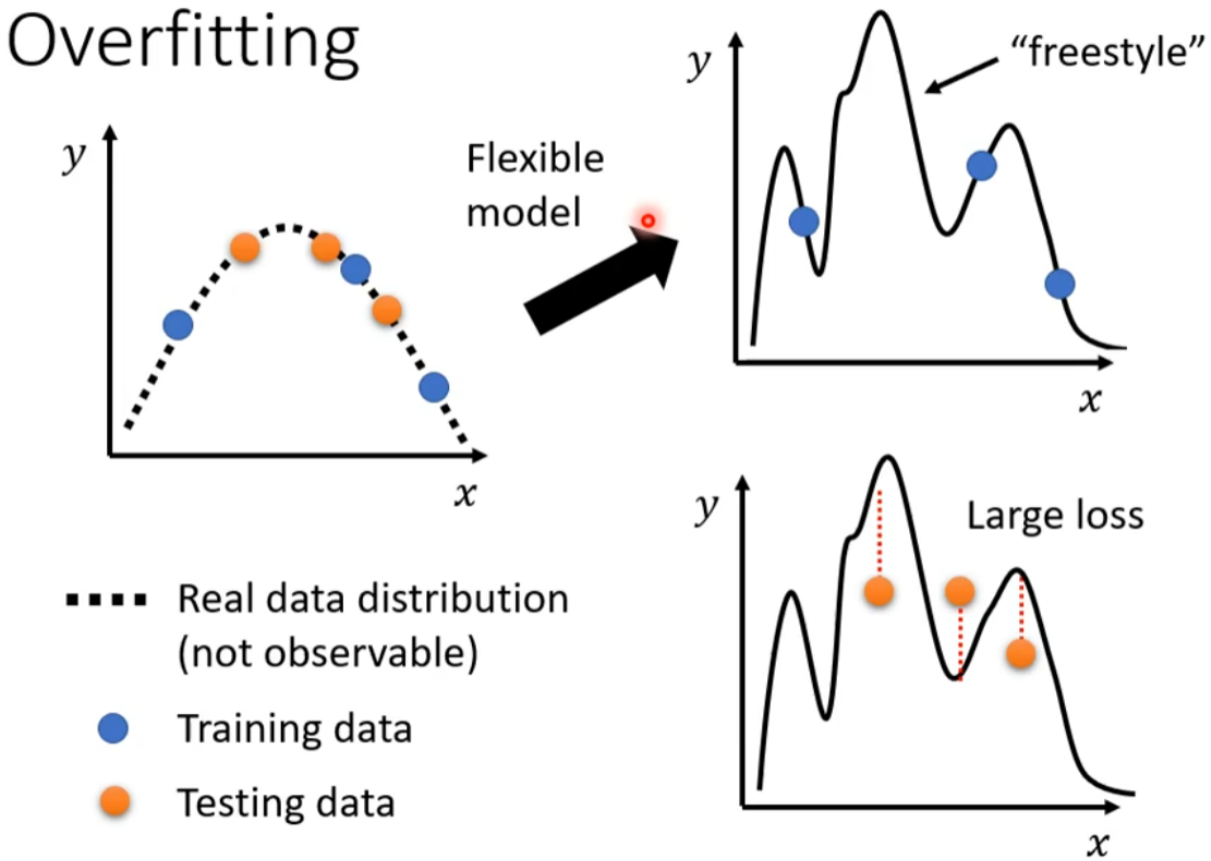
回顾：

- step1：定义含有未知参数函数 $y = f_\theta(x)$
- step2：从训练集中定义损失 $L(\theta)$
- step3：优化 $\theta^* = argmin_\theta L$

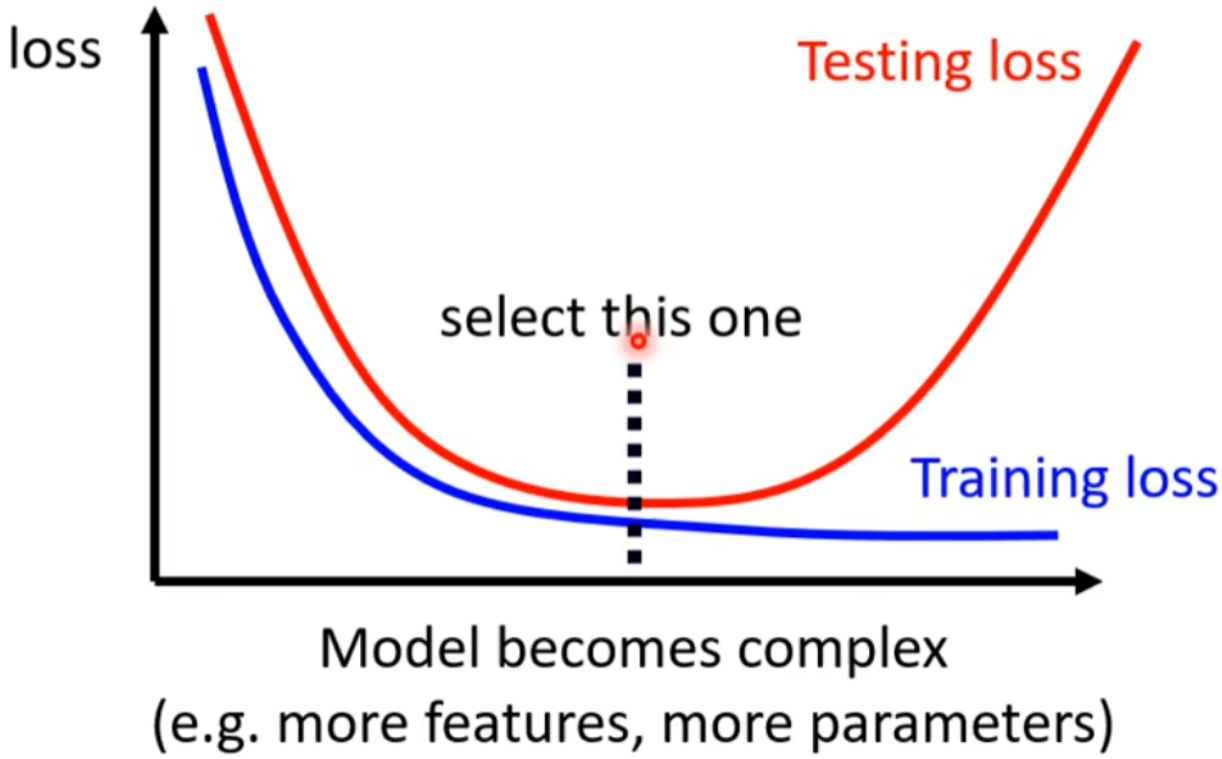
模型训练策略

- 训练集的损失过大
  - 原模型无法描述问题 -> **复杂化模型**（增加更多特征x，增加弹性（激活函数））
  - **优化问题（optimization issue）**：没有找到全局最好
  - 更多的层数堆叠引入了更多的未知数，理论上深的模型可以做到浅模型的功能，但更多参数optimization也会更难（**判断：通过深浅的模型对比**，如果深的表现更差就存在optimization issue）
  - 

- 训练集的损失小 -> 测试集的损失大
  - **过拟合**



- 解决方法：增加数据量，数据增强，减少弹性**适当**限制模型（更少的参数/共享参数，更少的特征，早停，正则，Dropout）



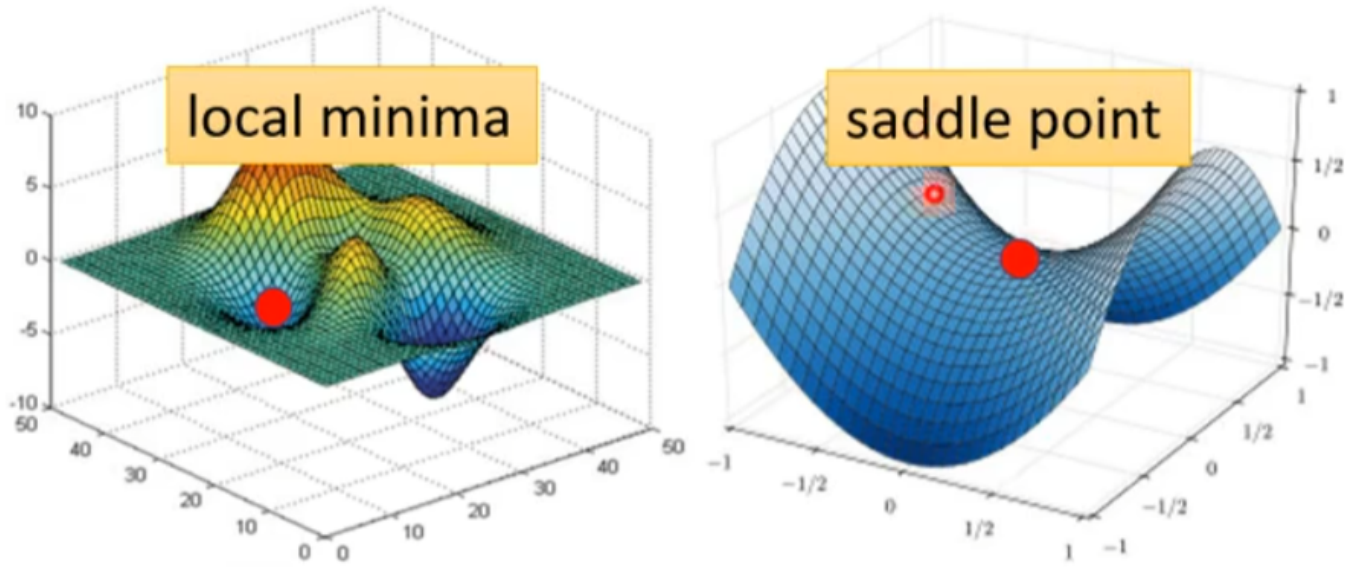
- 如何选择模型 -> 交叉验证：训练集、验证集、测试集

- mismatch：训练集和测试集是不同的分布

如何解决优化问题

存在问题：当梯度为0，不再更新了，训练停止 -> 局部问题。

梯度为0的情况（零界点）：（1）局部极小值，周边没办法走 （2）鞍部（局部极小值+极大值），可以往两边走



如果是saddle point还可以梯度更新，那如何根据数学判断两种情况：

Tayler Series Approximation

$L(\boldsymbol{\theta})$ around $\boldsymbol{\theta} = \boldsymbol{\theta}'$ can be approximated below

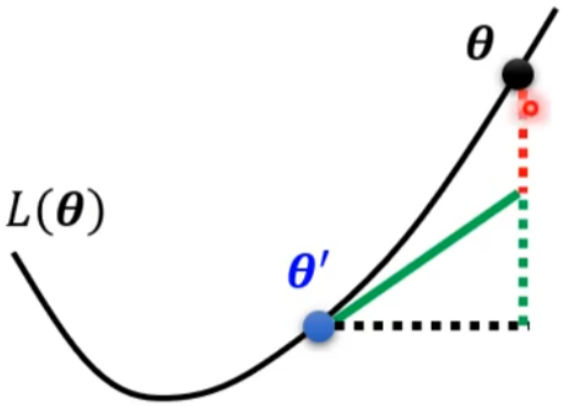
$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')$$

Gradient $\boldsymbol{g}$ is a *vector*

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}') \qquad \boldsymbol{g}_i = \frac{\partial L(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}_i}$$

Hessian H is a *matrix*

$$H_{ij} = \frac{\partial^2}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} L(\boldsymbol{\theta}')$$



当走到临界点时 <=> 梯度$g = 0$，此时用H判断临界点种类：

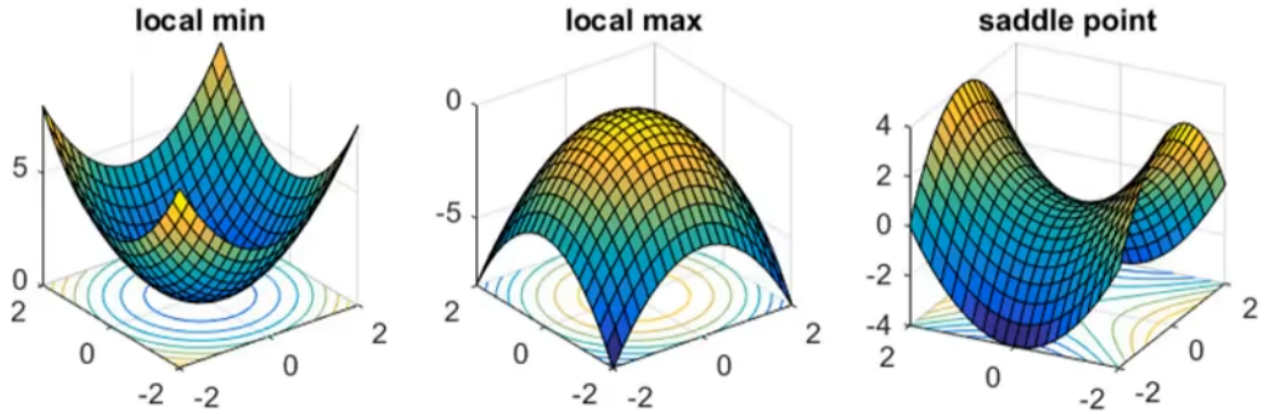
Hessian

$L(\boldsymbol{\theta})$ around $\boldsymbol{\theta} = \boldsymbol{\theta}'$ can be approximated below

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')$$

At critical point

telling the properties of critical points



令$v = \theta - \theta'$，我们可以根据推论用H的值判断：

For all $\boldsymbol{v}$

$\boldsymbol{v}^T H \boldsymbol{v} > 0$ ➡ Around $\boldsymbol{\theta}'$: $L(\boldsymbol{\theta}) > L(\boldsymbol{\theta}')$ ➡ **Local minima**

= H is positive definite = All eigen values are positive. ⬆

For all $\boldsymbol{v}$

$\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ Around $\boldsymbol{\theta}'$: $L(\boldsymbol{\theta}) < L(\boldsymbol{\theta}')$ ➡ **Local maxima**

= H is negative definite = All eigen values are negative. ⬆

Sometimes $\boldsymbol{v}^T H \boldsymbol{v} > 0$, sometimes $\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ **Saddle point**

Some eigen values are positive, and some are negative. ⬆

可以找一个向量λ让L变小

Don't afraid of saddle point?

At critical point: $L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^T H(\boldsymbol{\theta}-\boldsymbol{\theta}')$, where $\boldsymbol{v}^T H \boldsymbol{v}$

Sometimes $\boldsymbol{v}^T H \boldsymbol{v} > 0$, sometimes $\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ Saddle point

H may tell us parameter update direction!

$\boldsymbol{u}$ is an eigen vector of H

λ is the eigen value of $\boldsymbol{u}$

$\lambda < 0$

➡ $\boldsymbol{u}^T H \boldsymbol{u} = \boldsymbol{u}^T(\lambda \boldsymbol{u}) = \lambda \|\boldsymbol{u}\|^2$

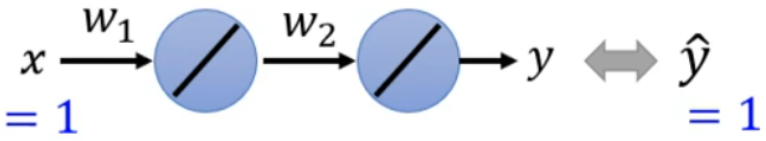
< 0 $\quad$ < 0

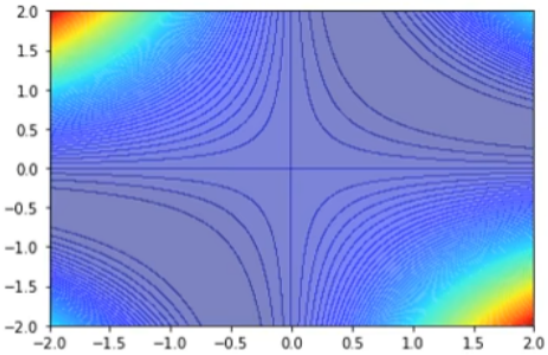
$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}')^T H(\boldsymbol{\theta}-\boldsymbol{\theta}')$ ➡ $L(\boldsymbol{\theta}) < L(\boldsymbol{\theta}')$

$\boldsymbol{u}$ $\quad$ $\boldsymbol{u}$

$\boldsymbol{\theta} - \boldsymbol{\theta}' = \boldsymbol{u}$ $\quad$ $\boldsymbol{\theta} = \boldsymbol{\theta}' + \boldsymbol{u}$ $\quad$ Decrease L

e.g.（计算H）：（0，0）点是saddle point





$$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$$

$$g = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \end{bmatrix} \quad \frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2) = 0 \quad \frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2)(-w_1) = 0$$

Critical point: $w_1 = 0, w_2 = 0$

$$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad \lambda_1 = 2, \lambda_2 = -2$$

Saddle point

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial w_1^2} & \frac{\partial^2 L}{\partial w_1 \partial w_2} \\ \frac{\partial^2 L}{\partial w_2 \partial w_1} & \frac{\partial^2 L}{\partial w_2^2} \end{bmatrix}$$

$$\frac{\partial^2 L}{\partial w_1^2} = 2(-w_2)(-w_2) = 0 \quad \frac{\partial^2 L}{\partial w_1 \partial w_2} = -2 + 4 w_1 w_2 = -2$$

$$\frac{\partial^2 L}{\partial w_2 \partial w_1} = -2 + 4 w_1 w_2 = -2 \quad \frac{\partial^2 L}{\partial w_2^2} = 2(-w_1)(-w_1) = 0$$

$\lambda_2 = -2$ Has eigenvector $\boldsymbol{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Update the parameter along the direction of $\boldsymbol{u}$

You can escape the saddle point and decrease the loss.

但这种二次微分的方法运算复杂，目前实际没有用这个方法逃离。

- 但考虑到维度的问题，路可能不止一条，**局部极小值可能很稀少**。实际中，即使趋向收敛也可能会让梯度继续下降。

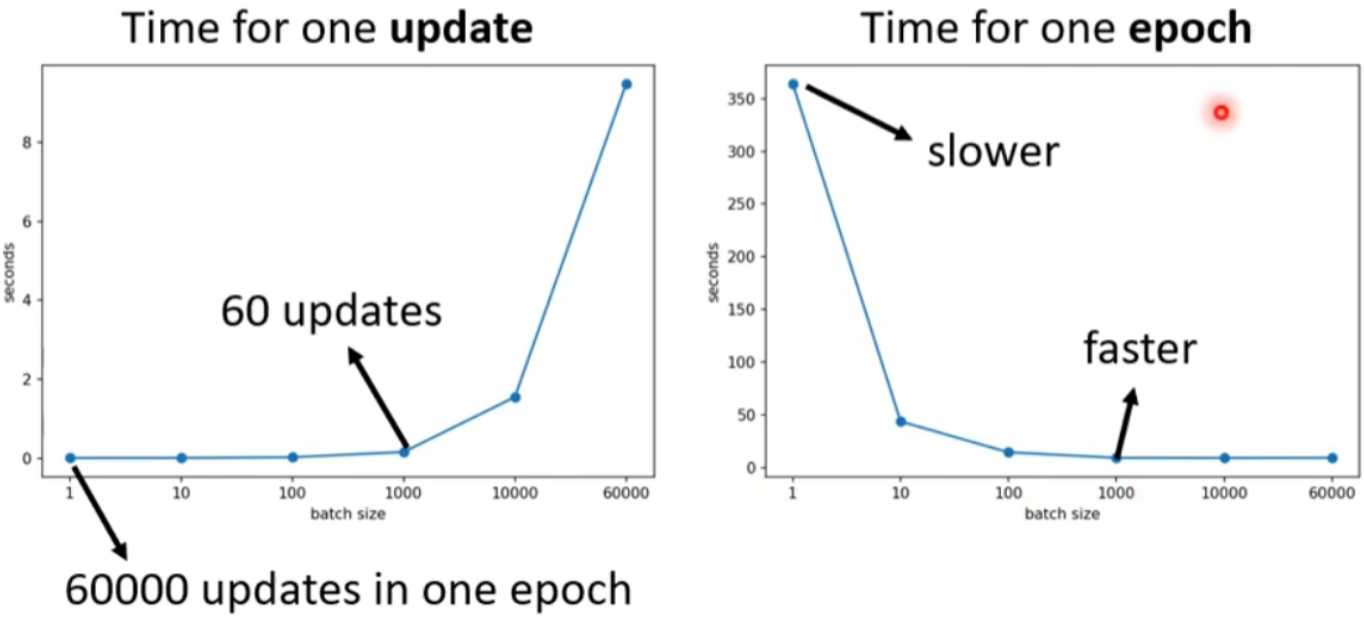
(1) Batch

在没有平行运算的情况下：

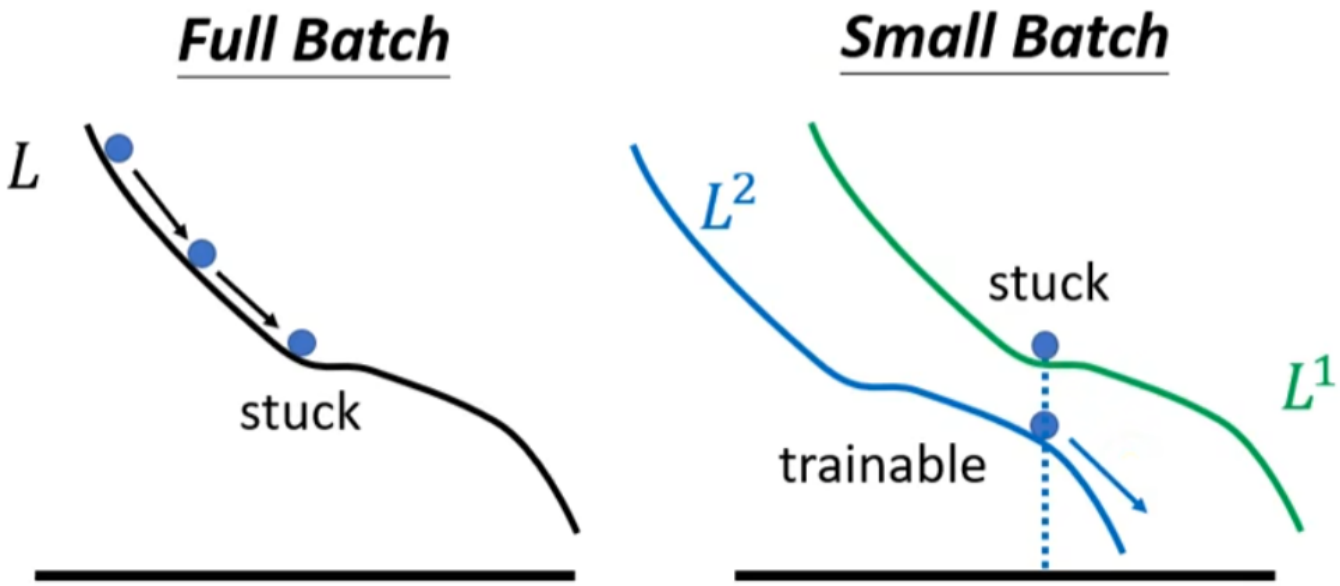
- 大Batch Size：全部数据看完才更新 -> 耗时长但是有用（稳）

- 小Batch size：分组多次跟新 -> 耗时少但是噪声影响大

但又由于GPU并行运算的存在，在一定范围内大Batch Size似乎表现更好：



但小Batch size有噪声的存在，可能会解决梯度下降卡住的问题



(2) Momentum

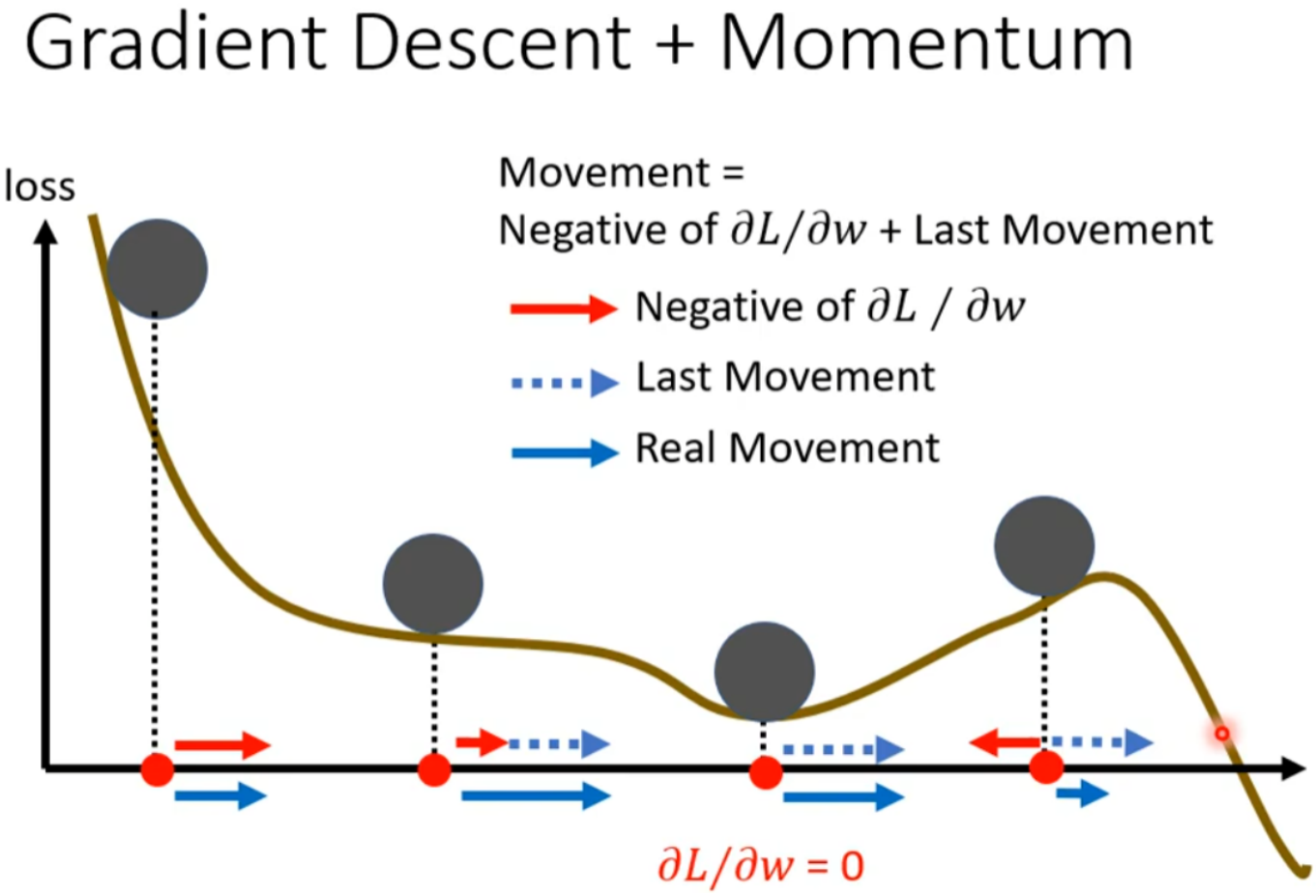
一般的梯度下降：$\theta^{i+1} = \theta^i - \eta g^i$

不是只往梯度的反方向移动，还加入前一步的Momentum：

$m^{i+1} = \lambda m^i - \eta g^i$

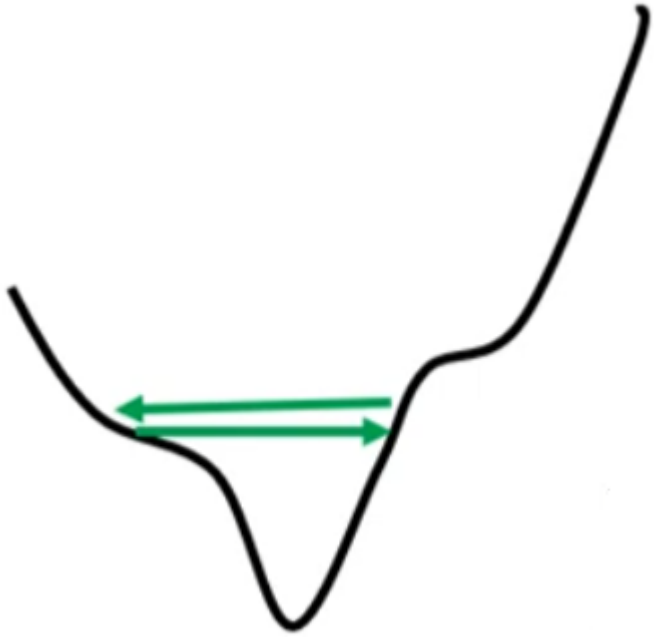
$\theta^{i+1} = \theta^i + m^{i+1}$

不仅仅和当前梯度有关，还和之前所有梯度有关



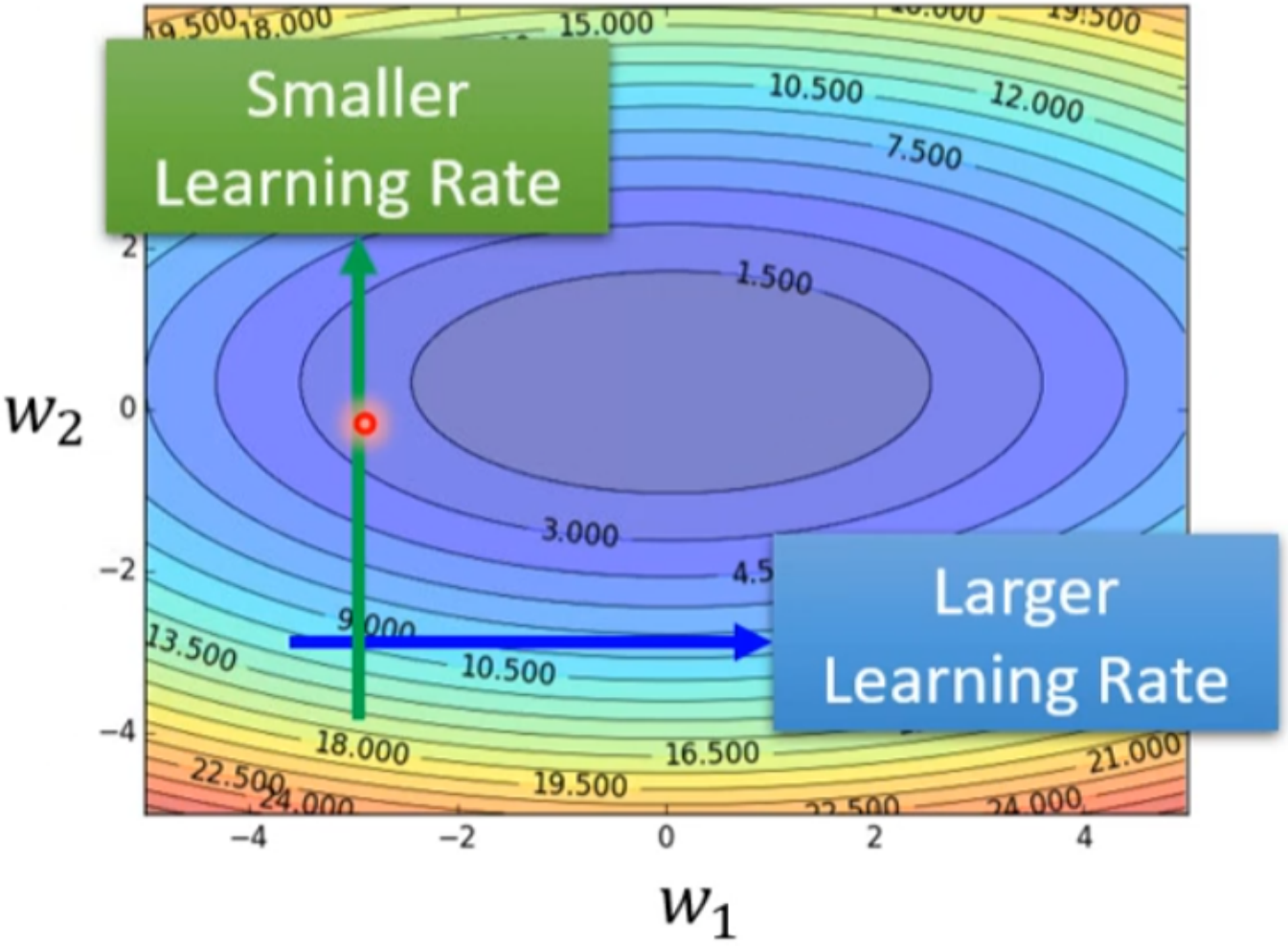
(3) 学习率

loss变小 != gradient变小，下面这个图就是左右来回走，一直没有到最低点

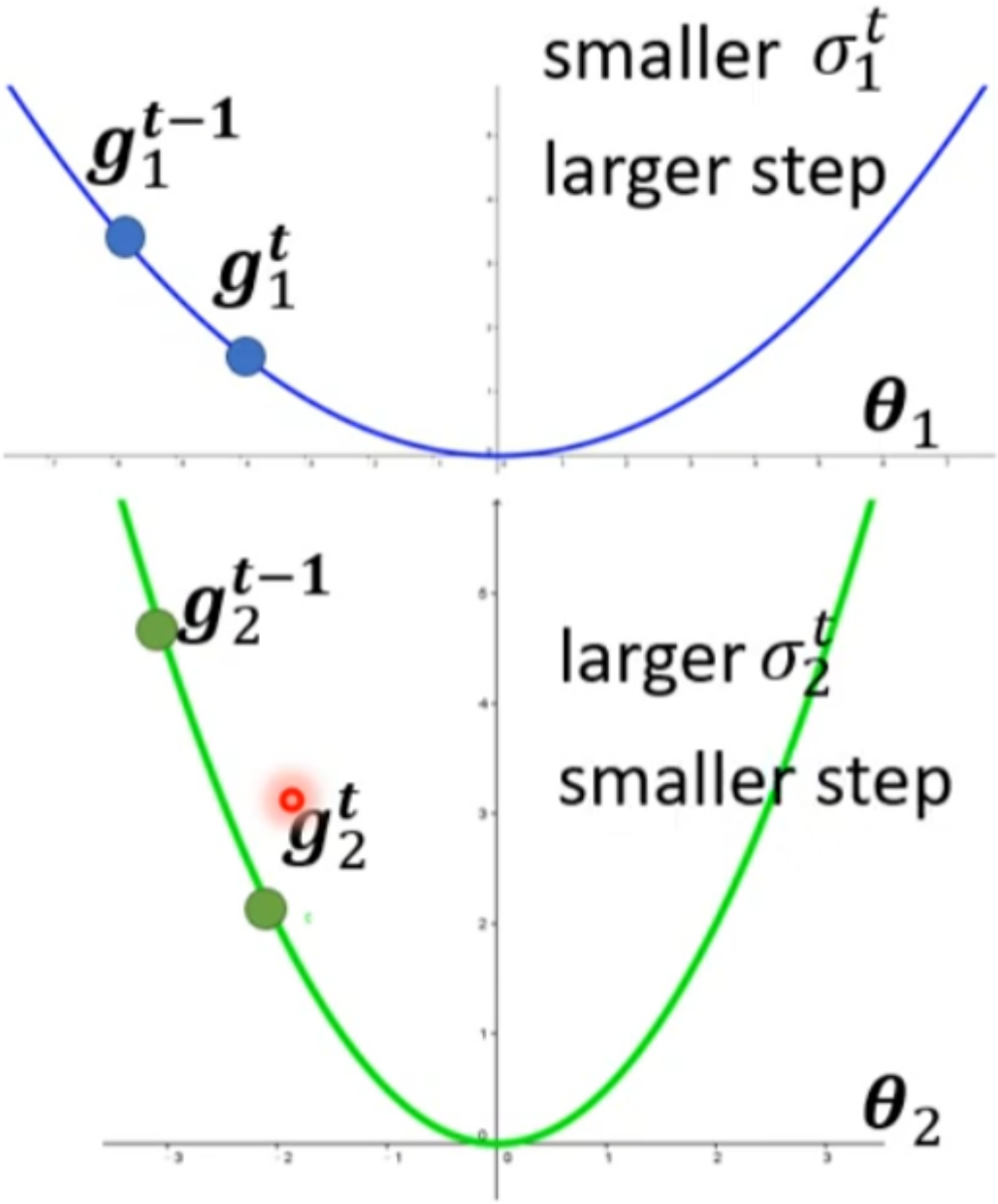


期望的η是多次走到谷底，one-size-fits-all的学习率是不能满足的 -> 特质化（期望：陡的时候小一点，缓的时候大一点）

使用$\theta^{i+1} \leftarrow \theta^i - \eta g^i$ 变成 $\theta^{i+1} \leftarrow \theta^i - \frac{\eta}{\sigma_i^t} g^i$，让 $\frac{\eta}{\sigma_i^t}$ 具有参数依赖



- 在Adagrad中$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=1}^{n}(g_i^t)^2}$



- RMSProp中，$\sigma_i^t = \sqrt{\alpha(\sigma_t^{t-1})^2 + (1-\alpha)(g_i^t)^2}$

RMSProp

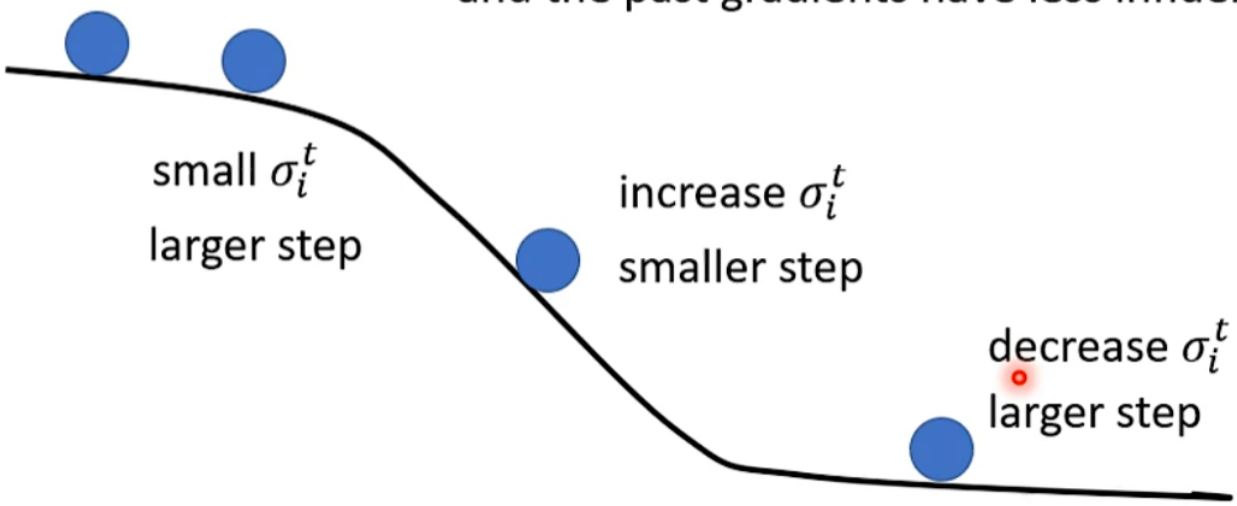
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\eta}{\sigma_i^t}\boldsymbol{g}_i^t$$

$\boldsymbol{g}_i^1\ \boldsymbol{g}_i^2 \cdots\cdots \boldsymbol{g}_i^{t-1}$

$0 < \alpha < 1$

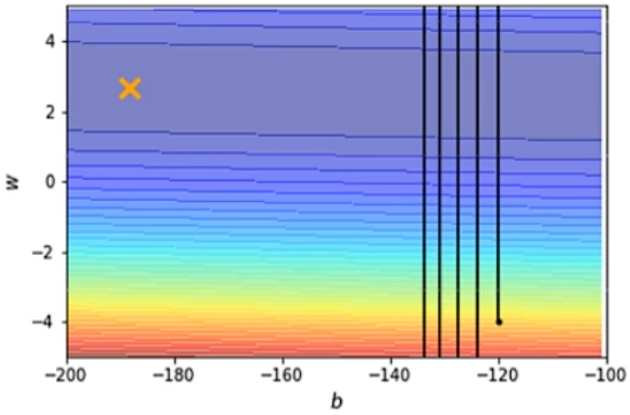
$$\sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^t\right)^2}$$

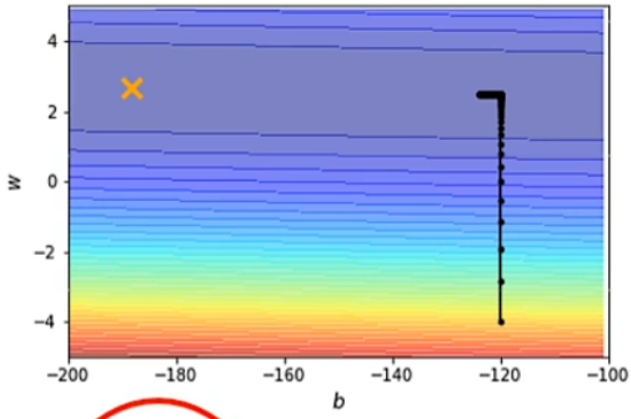
The recent gradient has larger influence, and the past gradients have less influence.



- Adam采用RMSProp+Momentum

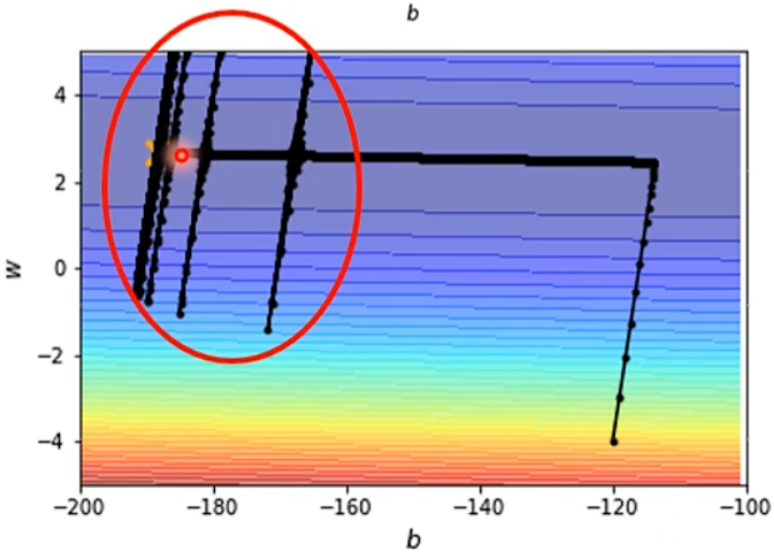
Without Adaptive Learning Rate





$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$$

$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}\left(g_i^t\right)^2}$$



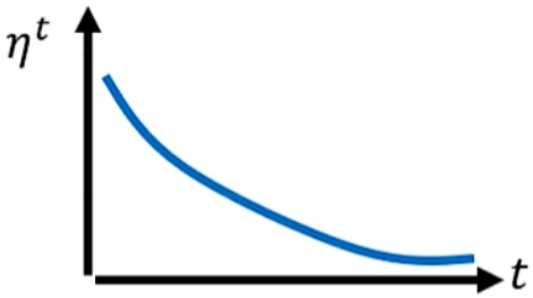
为了解决最后波动的问题，需要减小η，采用Learning Rate Scheduling -> 把η和时间t关联起来

（1）Learning Rate Decay：逐渐变小

（2）Warm up：先变大再变小（目前没有理论原因，解释是先变大可以有一个比较好的探索）

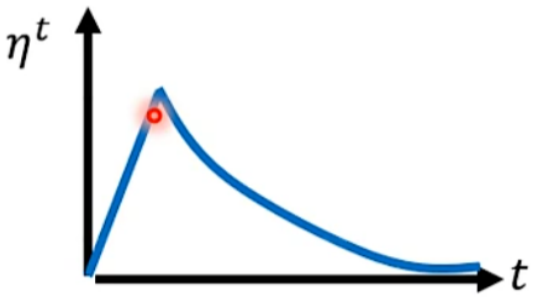
Learning Rate Scheduling

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} g_i^t$$



Learning Rate Decay

After the training goes, we are close to the destination, so we reduce the learning rate.



Warm Up

Increase and then decrease?

At the beginning, the estimate of σ_i^t has large variance.

总结各种梯度下降的提升计算方法：

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

Learning rate scheduling

Momentum: weighted sum of the previous gradients

Consider direction

root mean square of the gradients

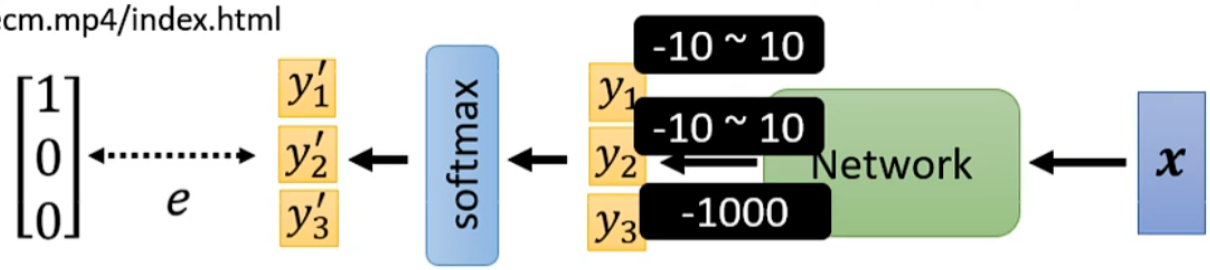
only magnitude

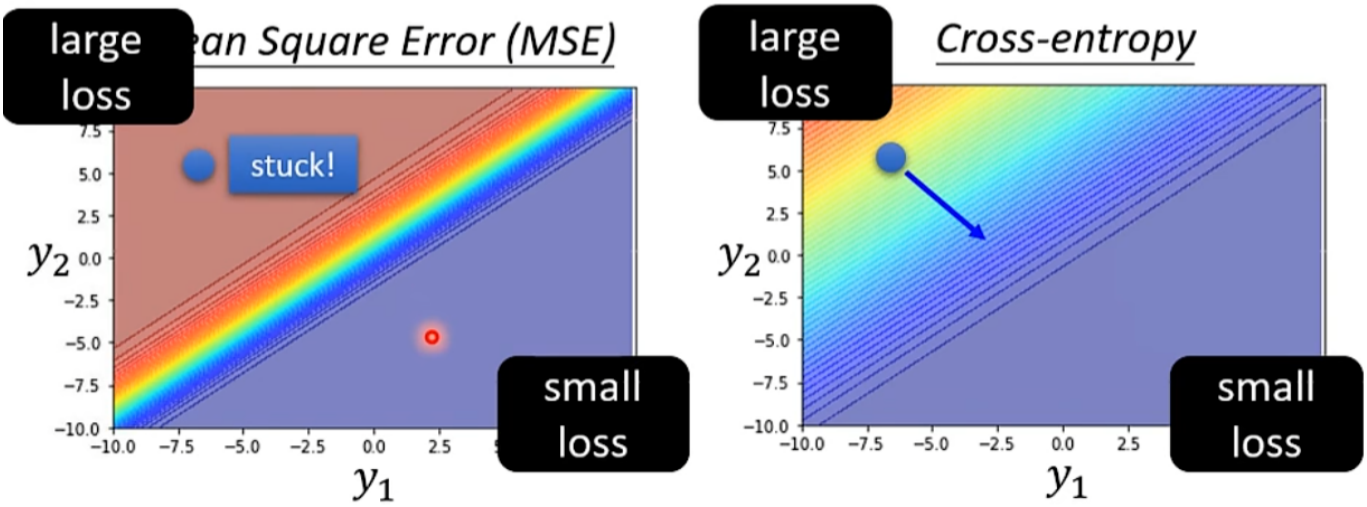
（4）损失函数

修改损失函数让梯度变得简单好找最小点，要选择更合适的损失函数。

例：在分类任务中，选择MSE还是交叉熵。用MSE在左上角更新的就比较小，不如交叉熵。

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Deep%20More%20(v2).ecm.mp4/index.html





个人总结

深度学习的核心是选择学习拟合函数和未知参数的确定。

功能复杂函数的函数有较强的学习能力，但在优化中由于较多的未知参数可能无法较快的确定较优的值（梯度下降面临的困难核心是无法在临界点"走出来"），因此衍生了一系列在保持神经网络结构框架前提下，优化梯度下降的方法。

（1）使用Batch会引入噪声，在一定程度上可以跨越一些零界点。

（2）使用Momentum向量可以在一定程度上影响梯度下降的方向，而不是固定的往微分的反方向移动。

（3）自适应调整学习率，在梯度陡时移动的少点，缓时移动的多点。

（4）选择合适的损失函数，可以从根本上减小梯度下降的难度。