Lecture 7:

Training Neural Networks,

Part 2

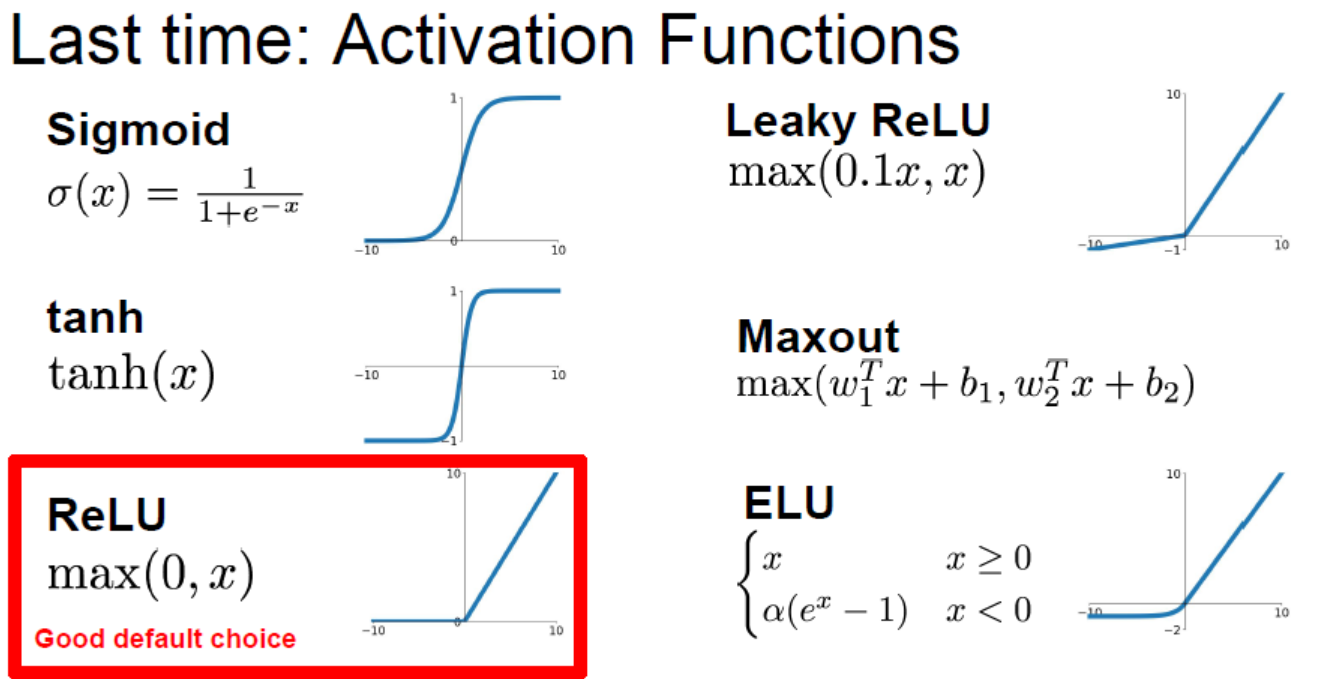
更好的优化

正则化

迁移学习

## 上节课回顾：

# 激活函数一般用ReLU



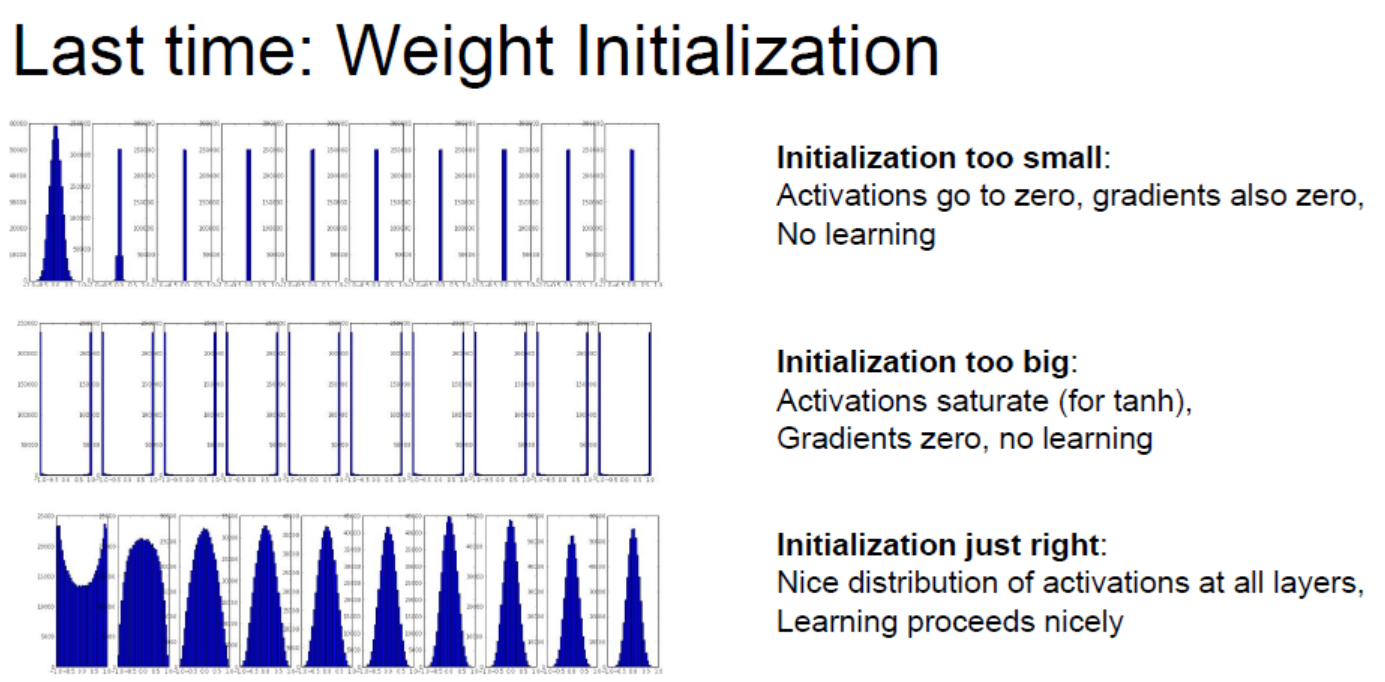
### W 的初始化：

太小的话 后面层无法激活

太大的话 后面的层饱和，导致激活函数为 -1， +1

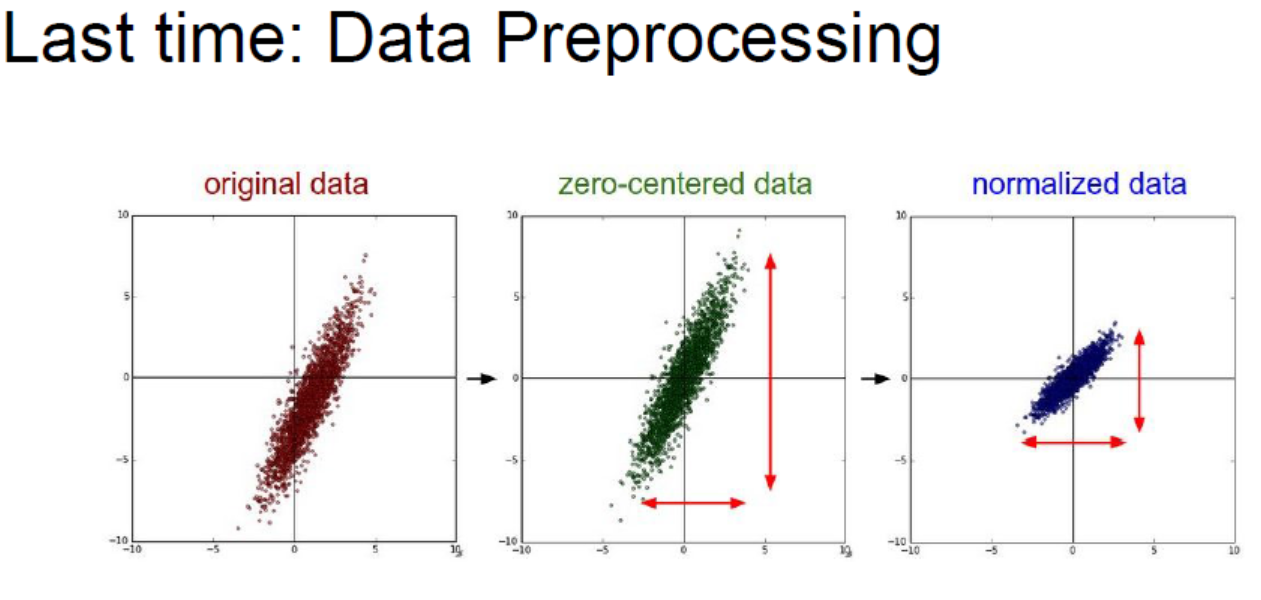
不大不小，gaussian分布最好

Xavier, MSRA 法。。



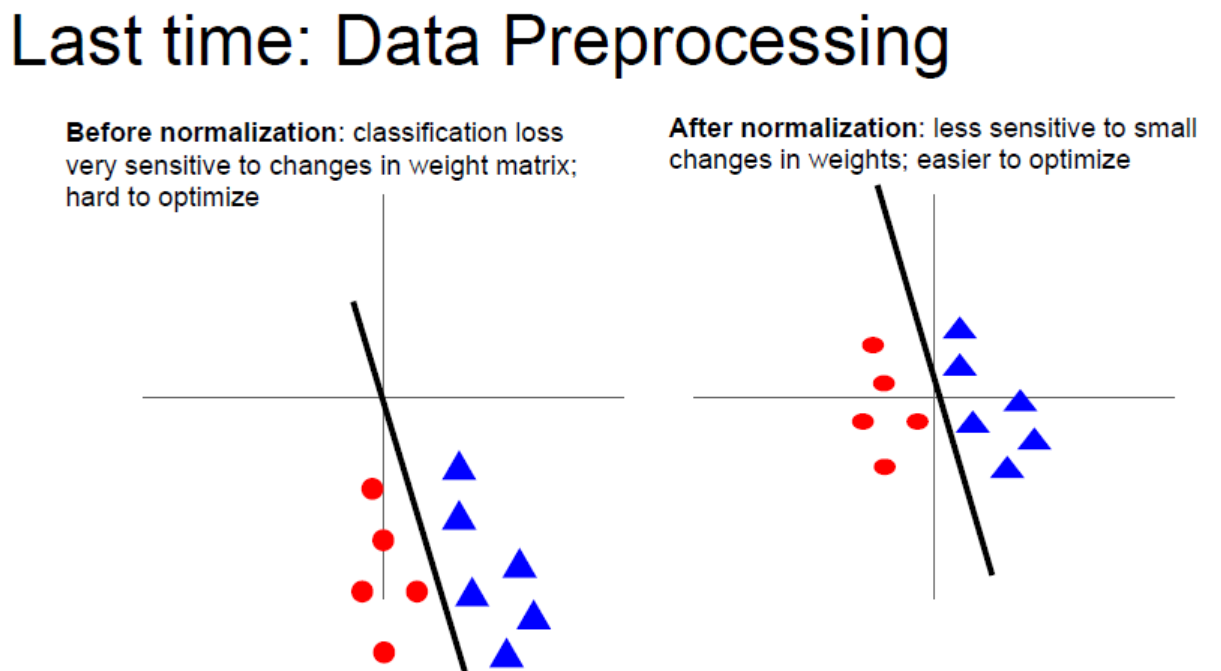
# 预处理

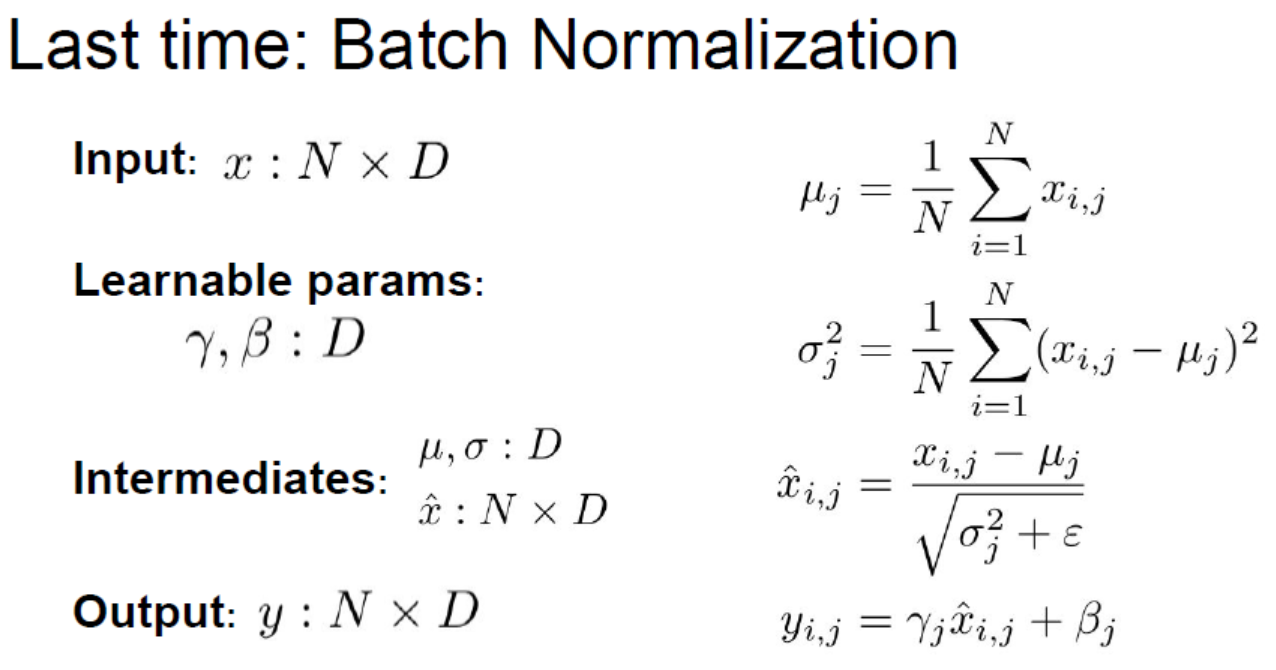
### Batch Normalization ：0 mean, std 标准差，方差为1



为什么要预处理

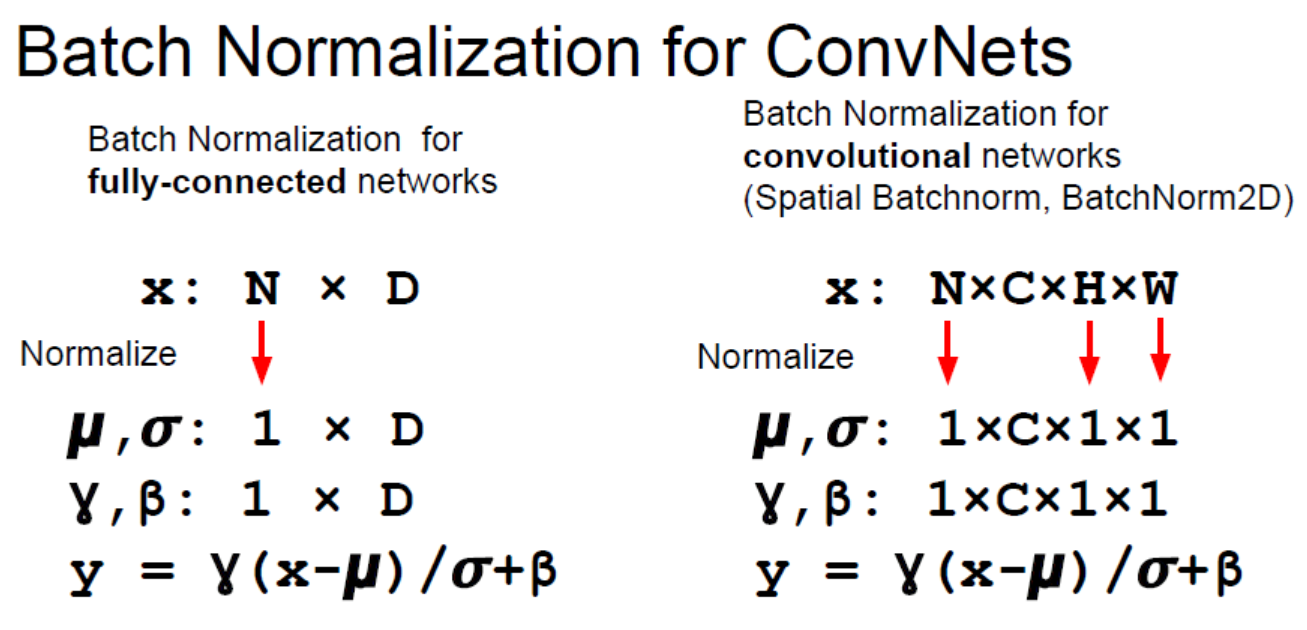
没有归一化对分类线的变动比较敏感

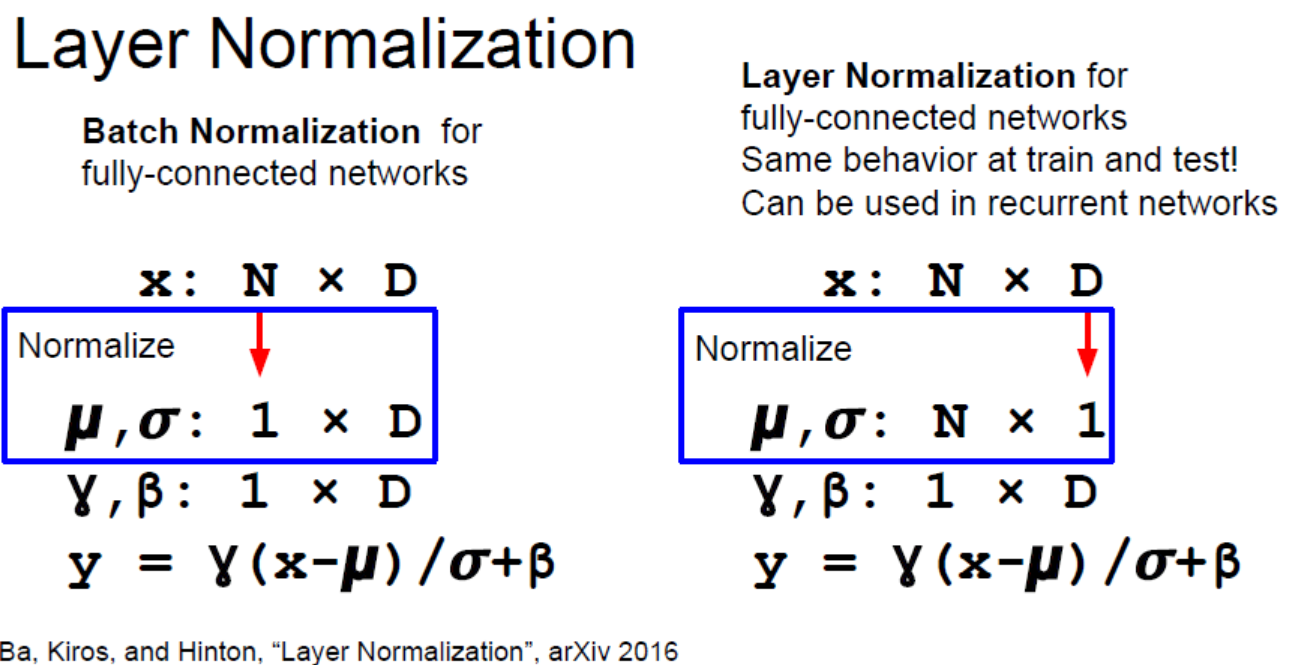


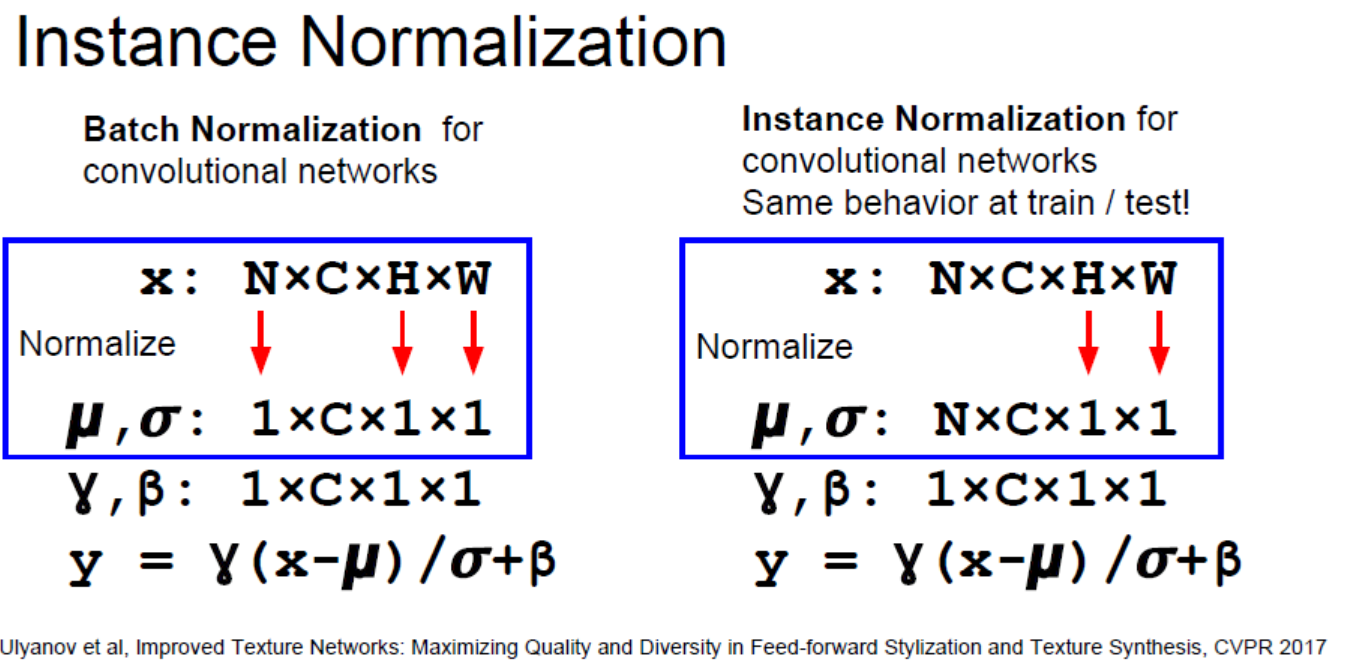


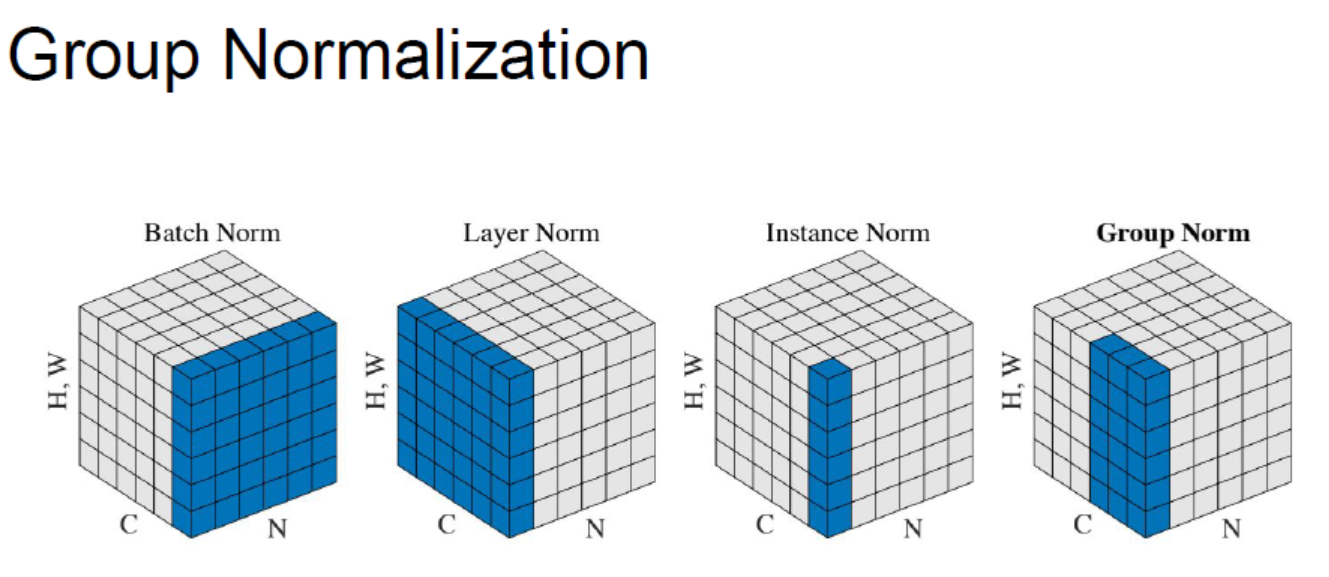
对卷积层的归一化：

不同的归一化方法



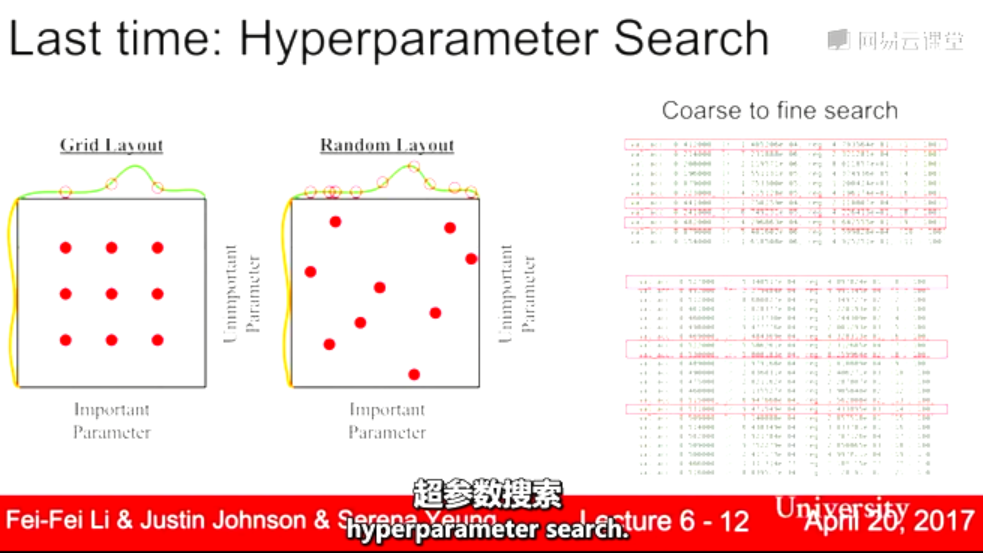






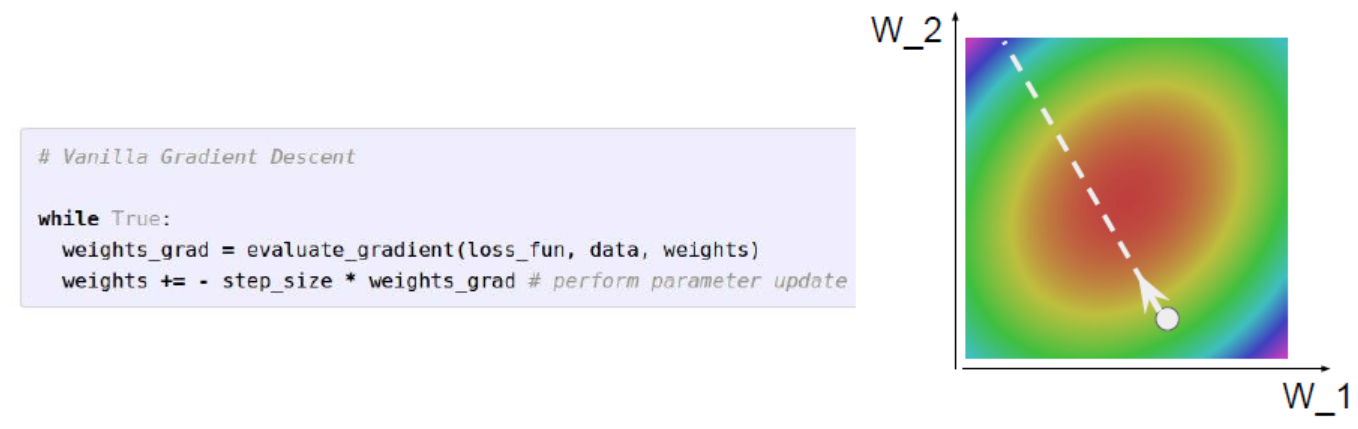
# Babysitting Learning Process:

# 超参数的搜索（一开始需要确定范围）



1. 更好的优化

# 简单逻辑：



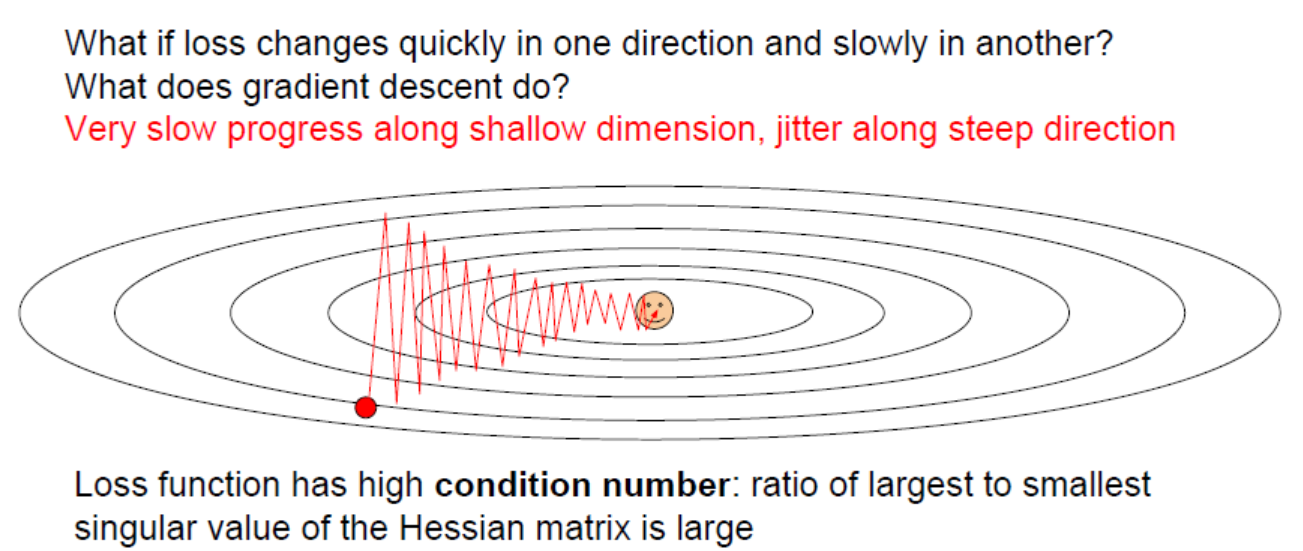
# 产生如下问题：

1. 如下图，W1,W2, 两个参数，改 变一个，loss改变很小， 而改变另一个，loss改变就很大，如下，水平方向变动时loss变动很小, 在垂直方向变动时，loss变动很大。

在这点是，

Hessian矩阵最大很最小奇异值的比值很大，loss function 的条件数很大



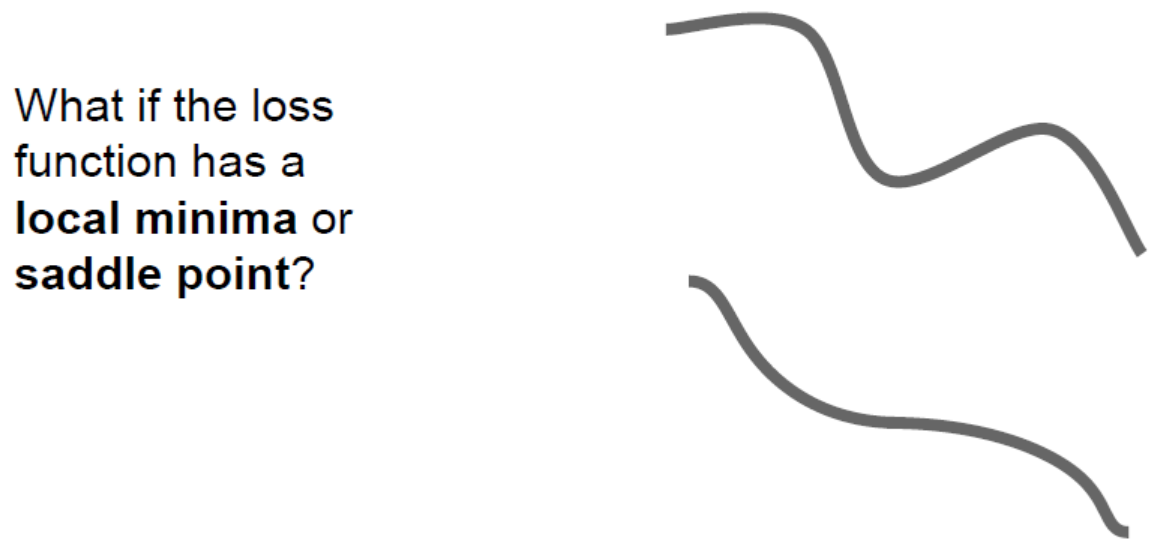


这时用SGD 梯度下降时，就会出现方向并不是和梯度最低点在一个直线上，而是以‘之’字形下降，这种情况在高维空间上更普遍。

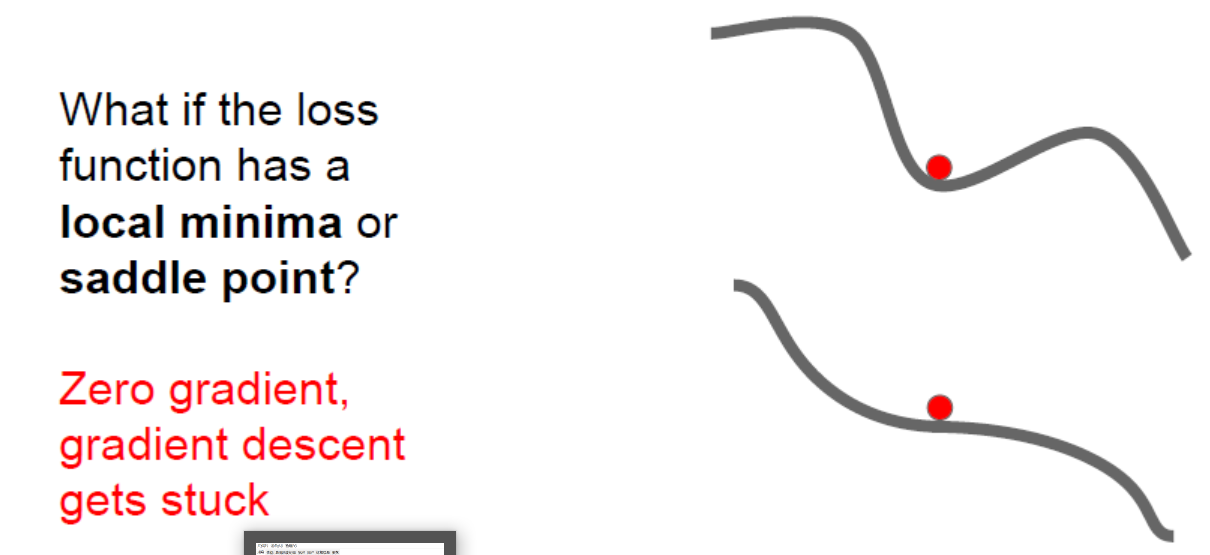
1. Saddle 鞍点 （局部最小点）

下图两个图像，横轴对应两个参数 W1,W2,

Y轴代表loss值



这时SGD会卡在中间



在高维度中会出现很多鞍点，但很少的可能性会出现局部最小值

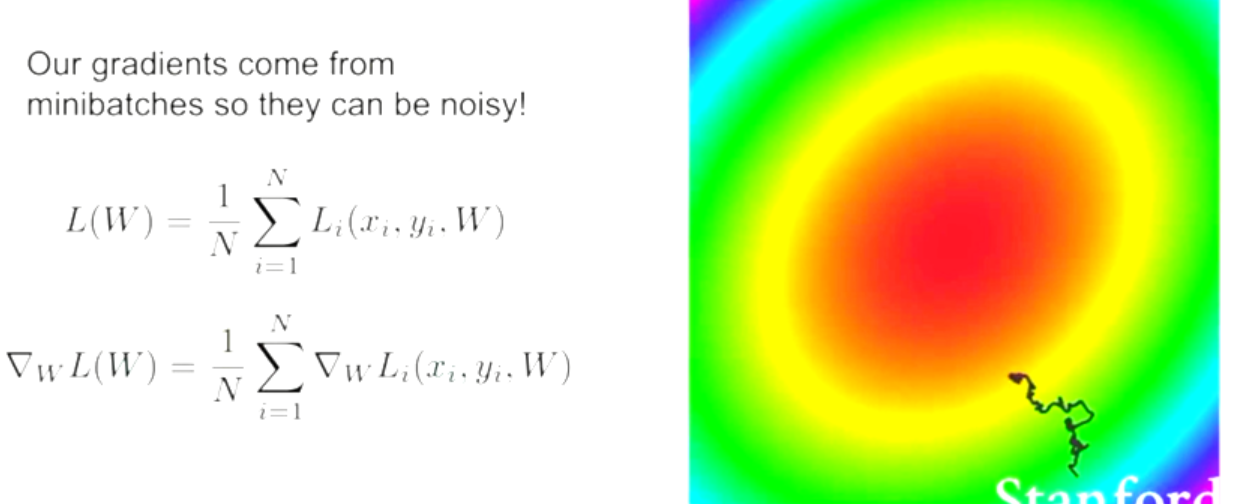
会出现在鞍点附近更新的很慢

1. 随机; stochastic

Loss计算是对训练集的数据进行全部计算的，要，如果这个N是整个训练集那么计算量是很大的

所以实际中在对一部分样本进行梯度计算的，所以实际上不会每一步都计算真正的loss 和 梯度

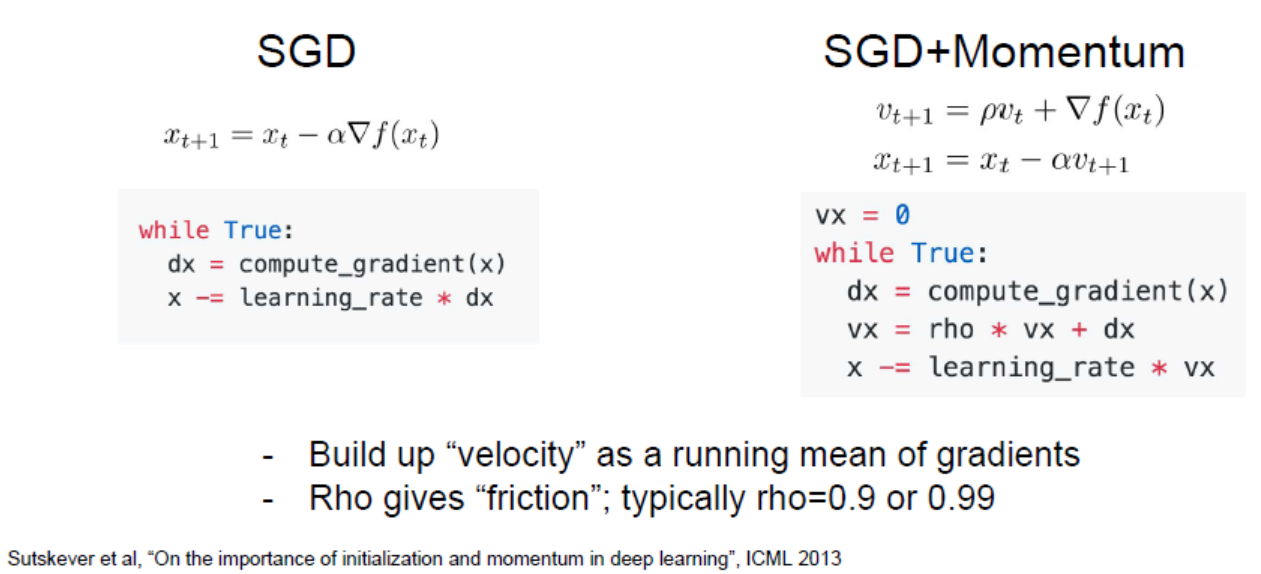
也就是说根据 一部分数据计算的梯度可能是干扰，而不是真正的梯度



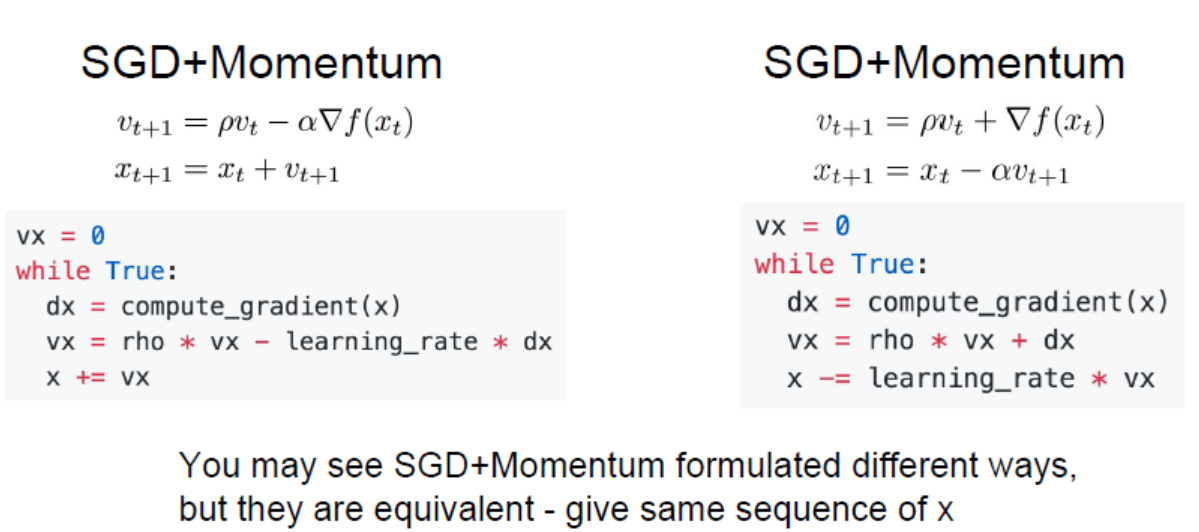
## 解决方法：SGD + Momentum

不是在SGD上求梯度，而是在SGD上求梯度后再加上一个Momentum，

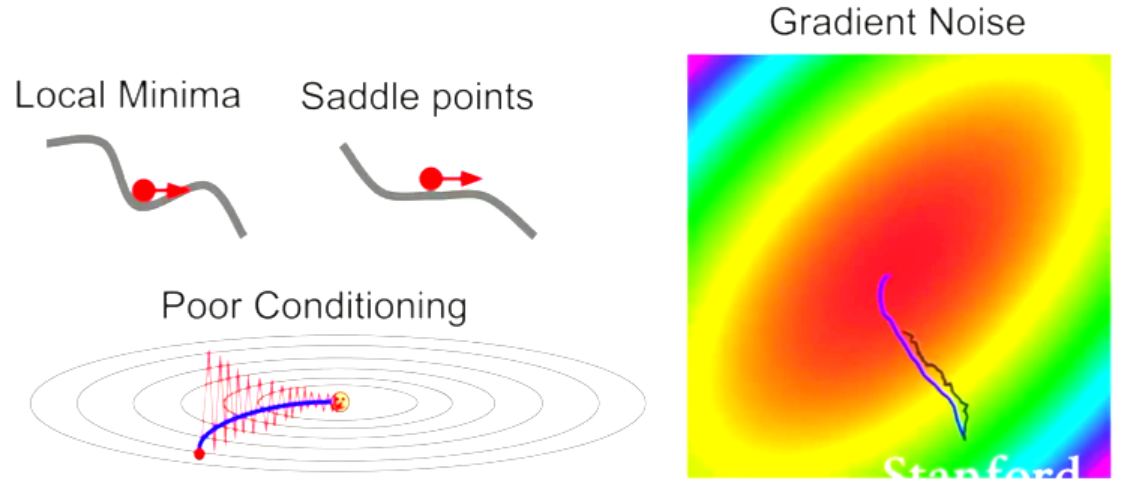
Rho是一个摩擦系统，让Momentum衰减



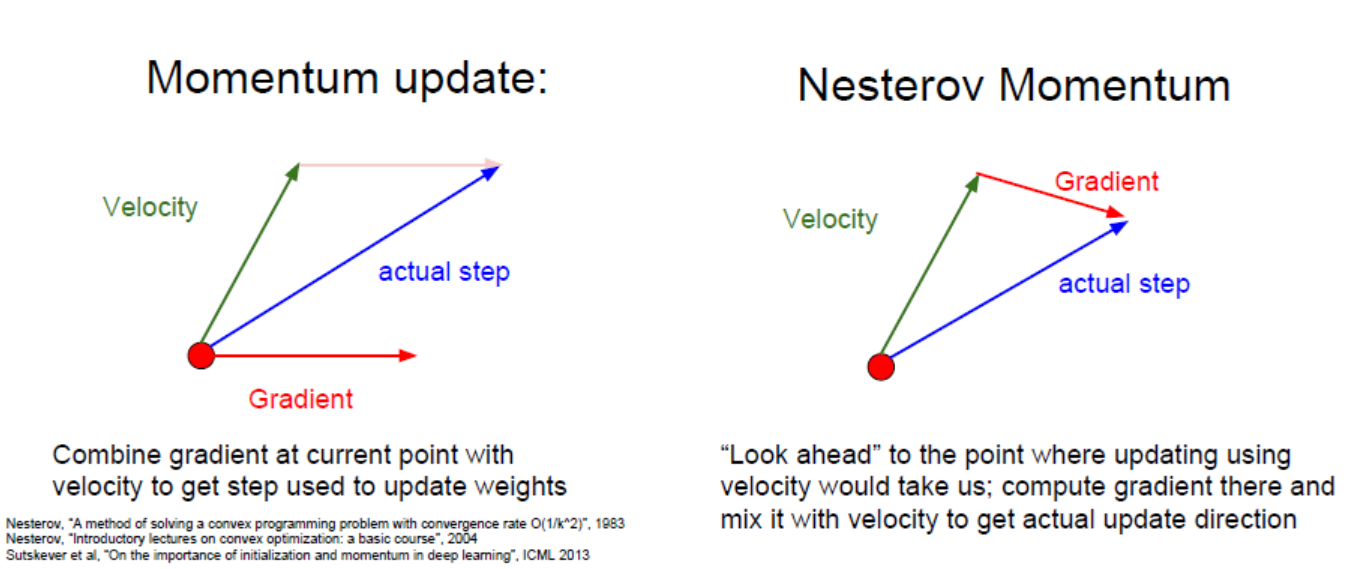
不同的计算方法：

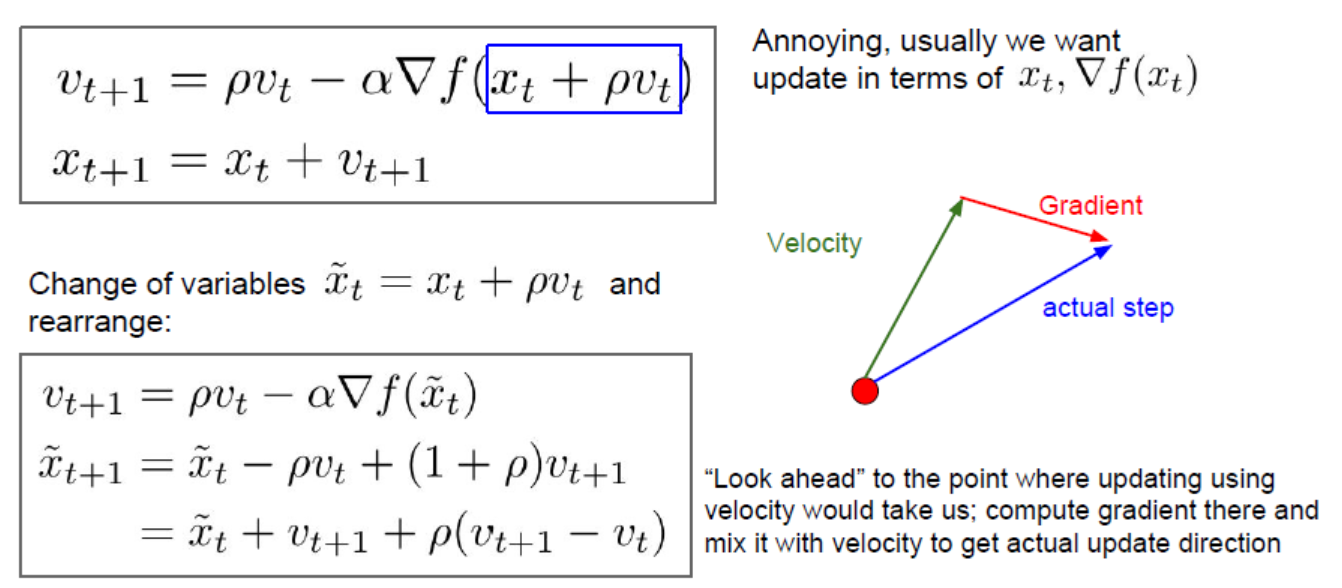


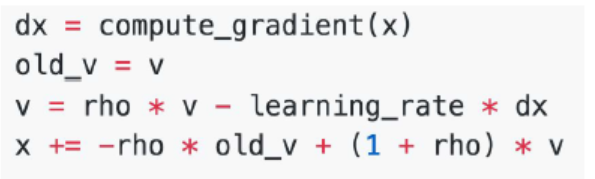
另外当某一点没有梯度时，因为这里有速度，所以可以继续下降，也就避免在鞍点卡住



## 注意有一种 Nesterov Momentum 的东西



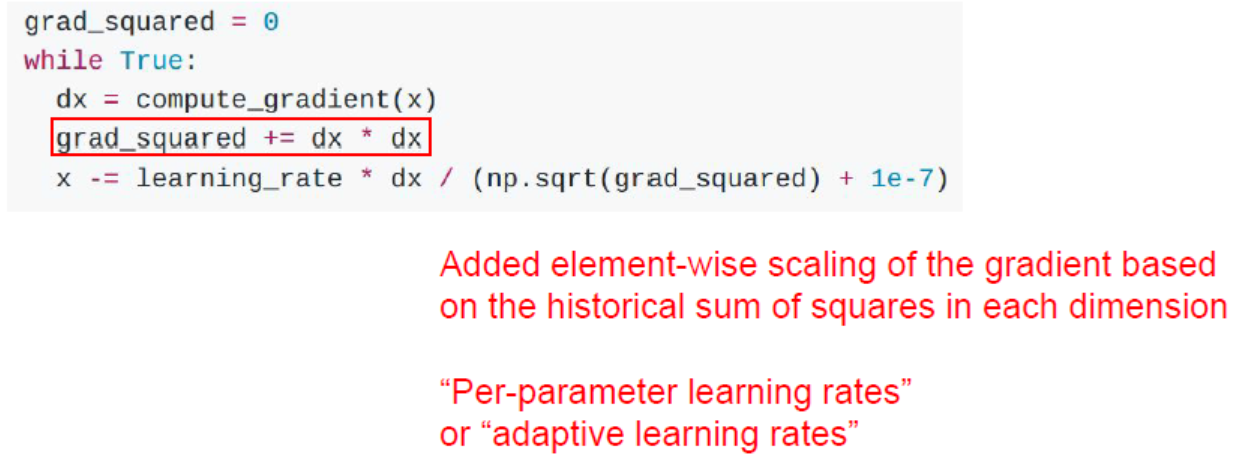




下图中可以看到，SGD+Momntum 和 Nesterov都跳过了局部最值，而SGD卡在那里没有更新了



## AdaGrad – 梯度平方

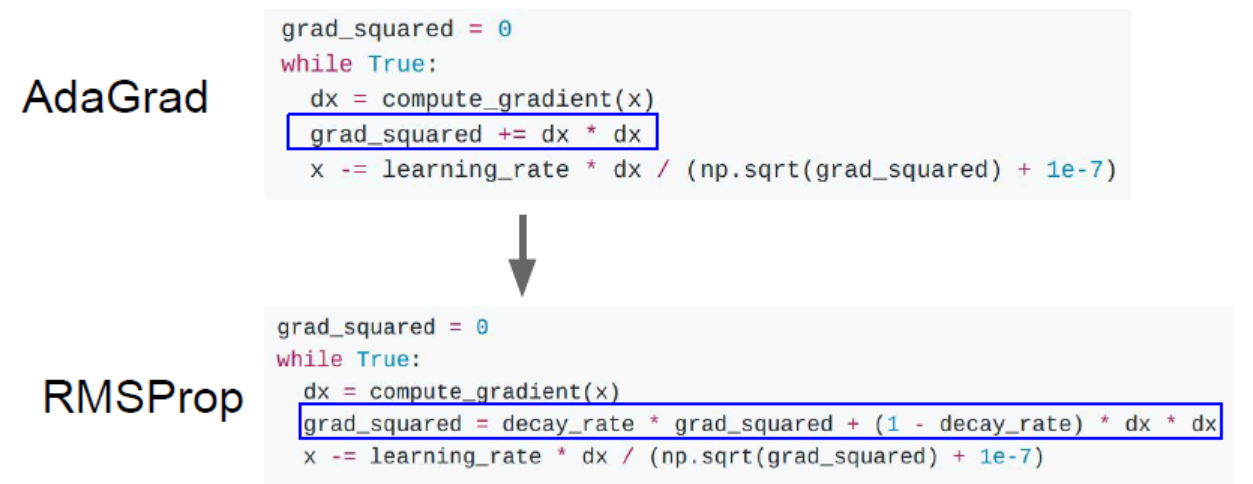


但是有个问题，就是当t越来越大时， 上式最后一步的分母会越来大，最后的步长会越来越小。

在凸函数的下，这点是比较好的特性

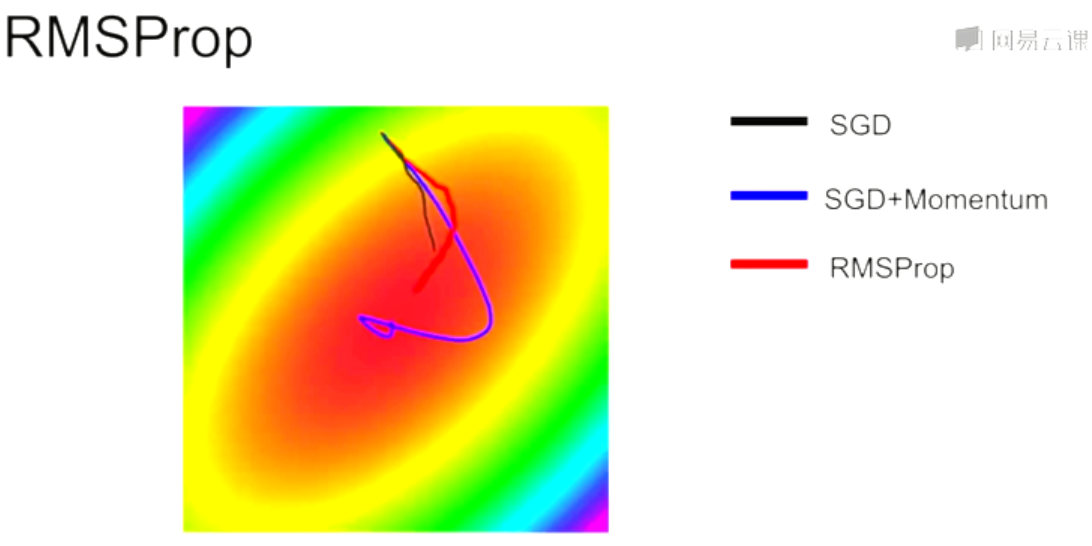
但是在非凸的情况 下就不是了，会卡在一个鞍点

## RMSProp : 对AdaGrad的梯度添加一个decay(0.9 0.99)



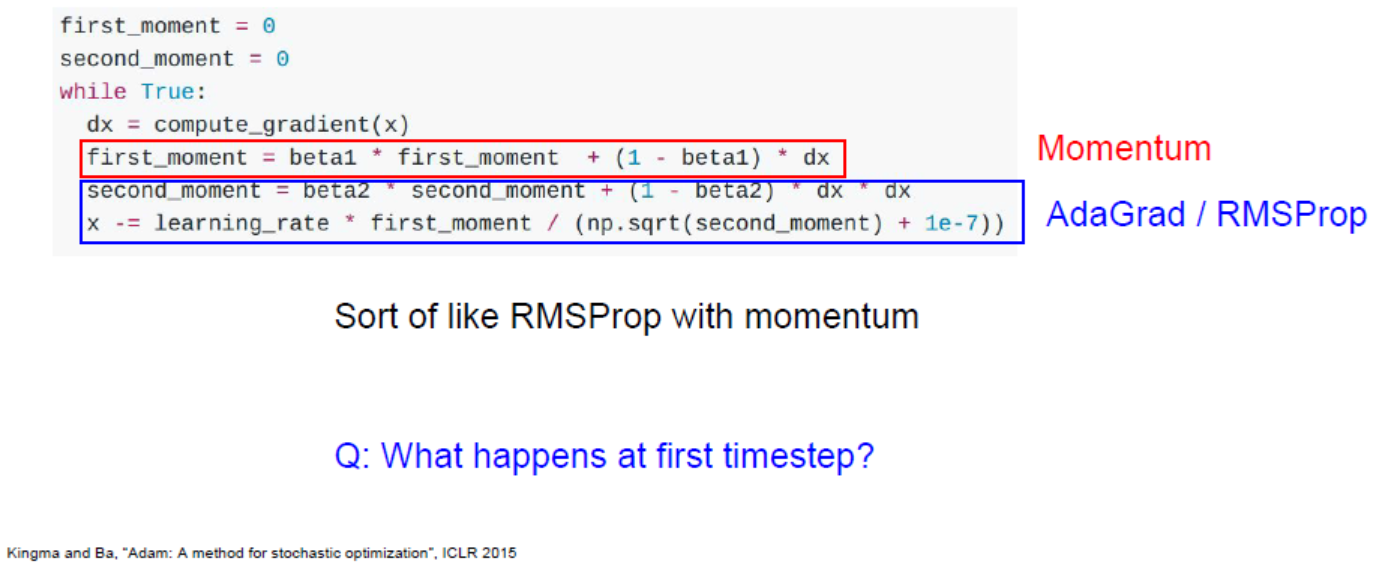
这样可以克服非凸情况下卡在鞍点的缺点，但是又会有下降变慢的问题。。。

可以看到 SGD+Momentum 和 RMSProp 都比SGD好，而SGD+Momentum会先有一个过头再转回来，而RMSProp会和SGD类似的下降，但不会卡在鞍点



一般次会使用AdaGrad

## Momentum 与 AdaGrad & RMSProp的结合： Adam(almost)



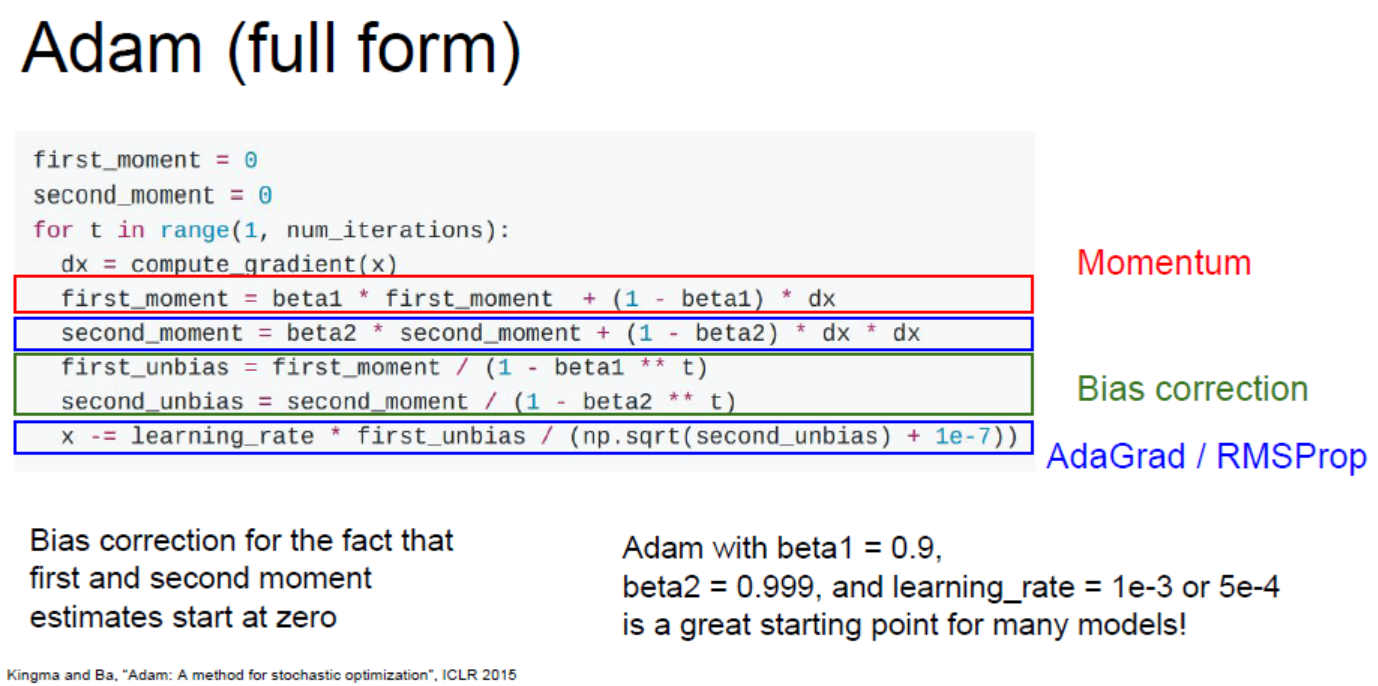
First momentum : grad

Second momentum: AdaGrad + RMSProp

避免初期比较大的步长：

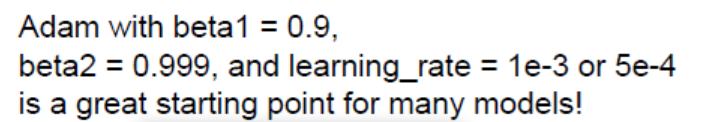
First momentum -> first unbias

Second momentum -> second unbias

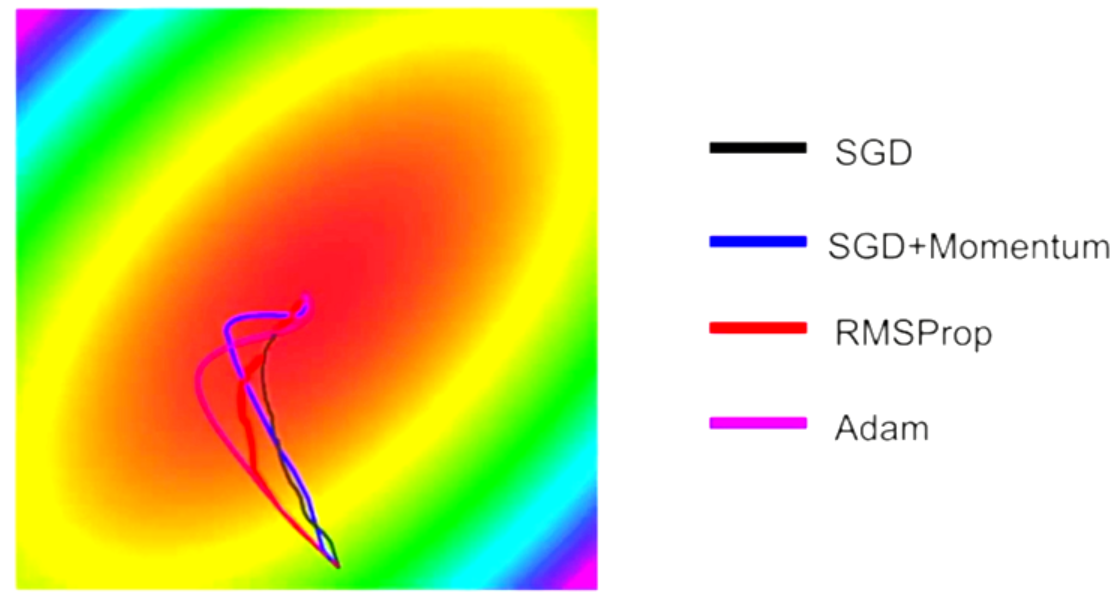


Adam 一般是处理更新梯度下降的默认算法，效果比较好

经验：

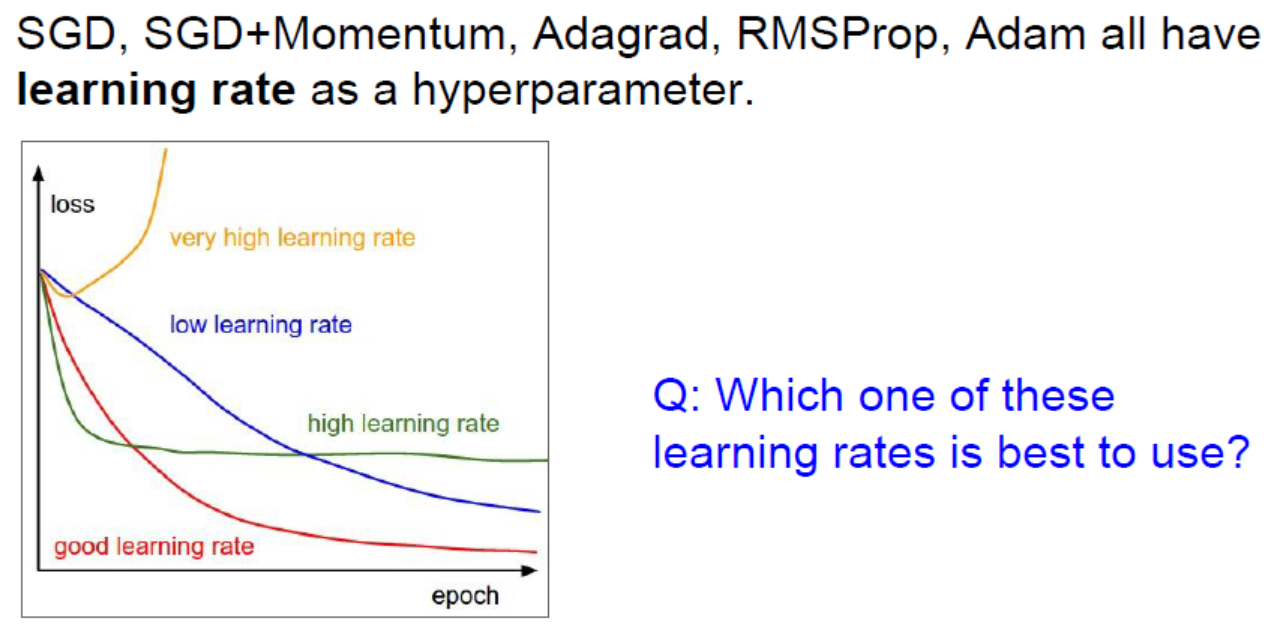


互相之间的比较：



## Learning Rate

一般学习率不是固定的，而是衰减的 （decay）



有不事的衰减方式：

