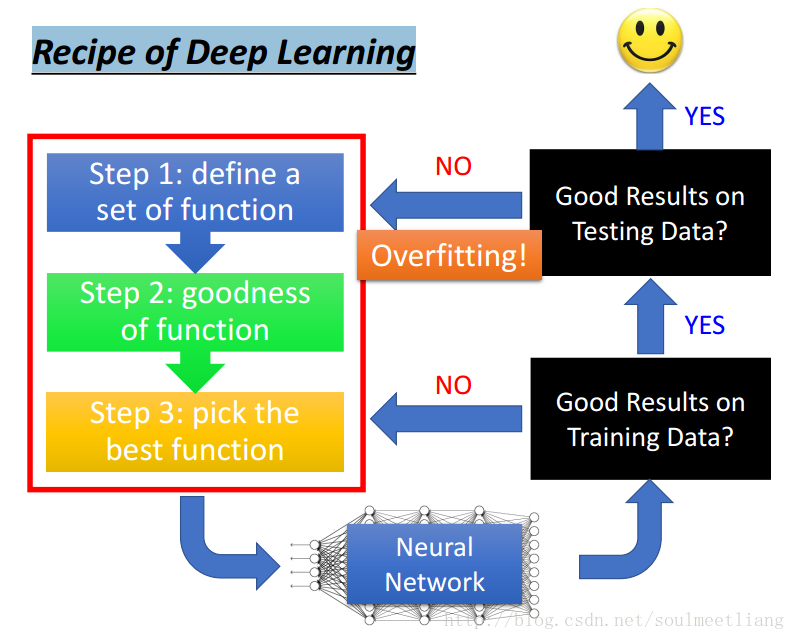
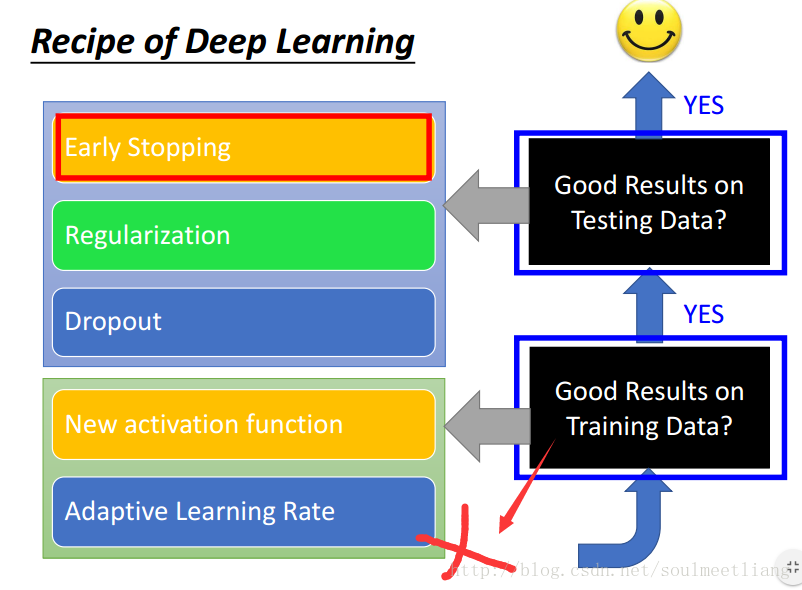
Recipe of Deep Learning

Three steps





Learning rate 与overfitting

学习率主要影响 收敛速度 Vs. 局部最优 的问题（为了达到更好的训练效果），和过拟合问题不在一个维度。如果对训练集都拟合不好，还何谈过拟合。

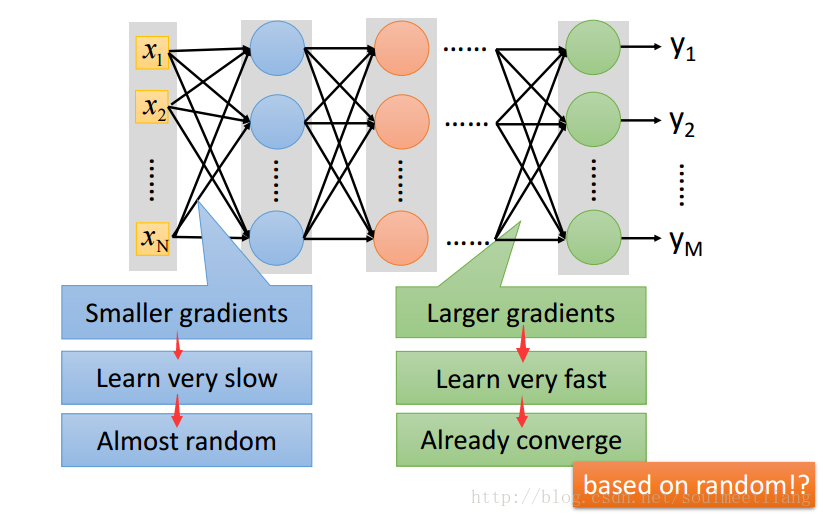
Dropout: A Simple Way to Prevent Neural Networks from Overfitting

解决training data上performance表现不好的方法

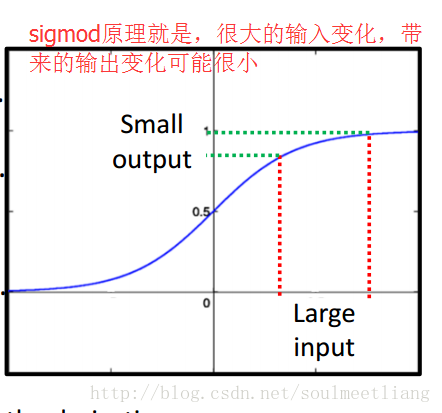
1. 更换更好的activation function

为什么有的时候neural net work层数增多，效果反而不好？

1. vanishing gradient problem

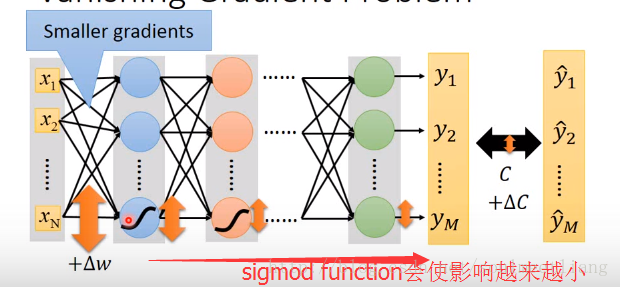


后面layer的输入是基于前面layer的output，所以前面的gradient会比较小，而后面的gradient会比较大，所以在前面layer的参数还在变化时，后面layer的参数就基本已经收敛了，就会导致，后面layer的参数是基于random的前面layer的参数，结果变差。



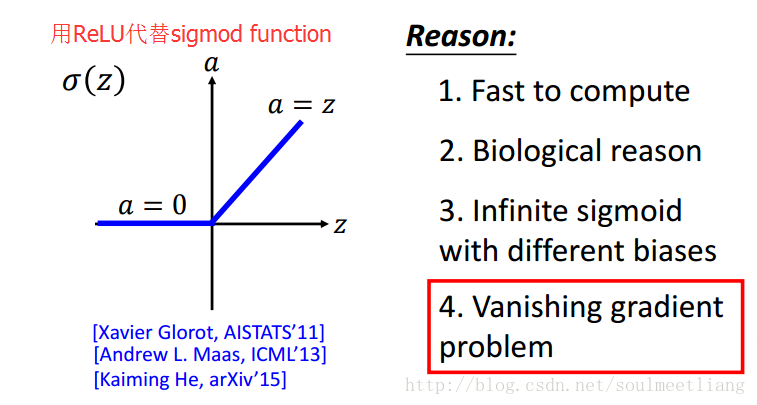
### Why?

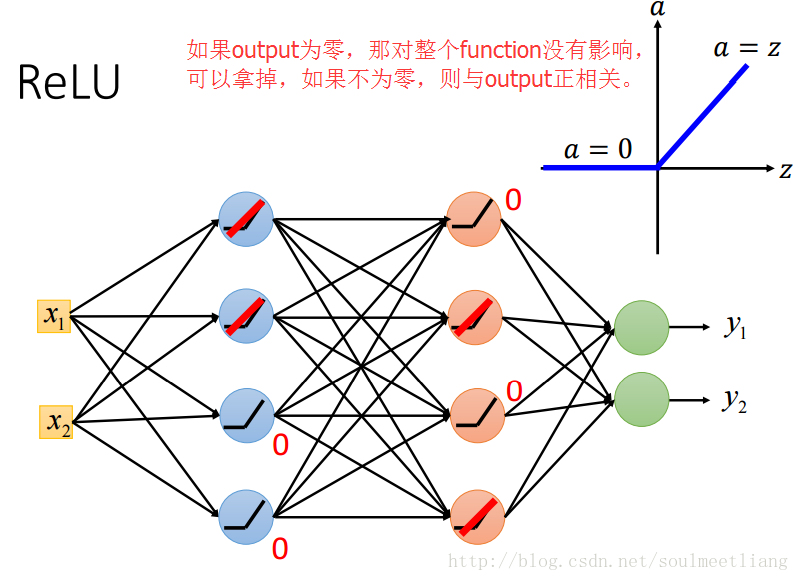
Sigmoid函数会使gradient衰减

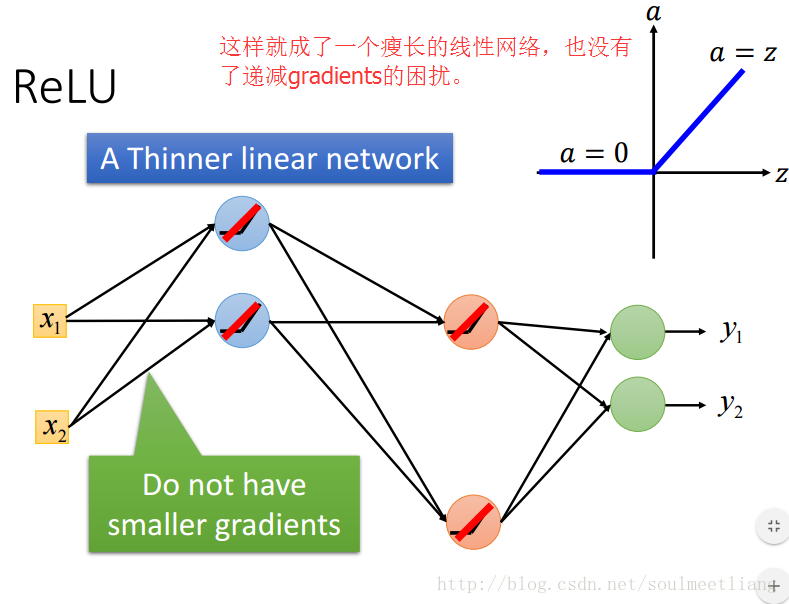


### Solution

ReLU (Rectified Linear Unit)



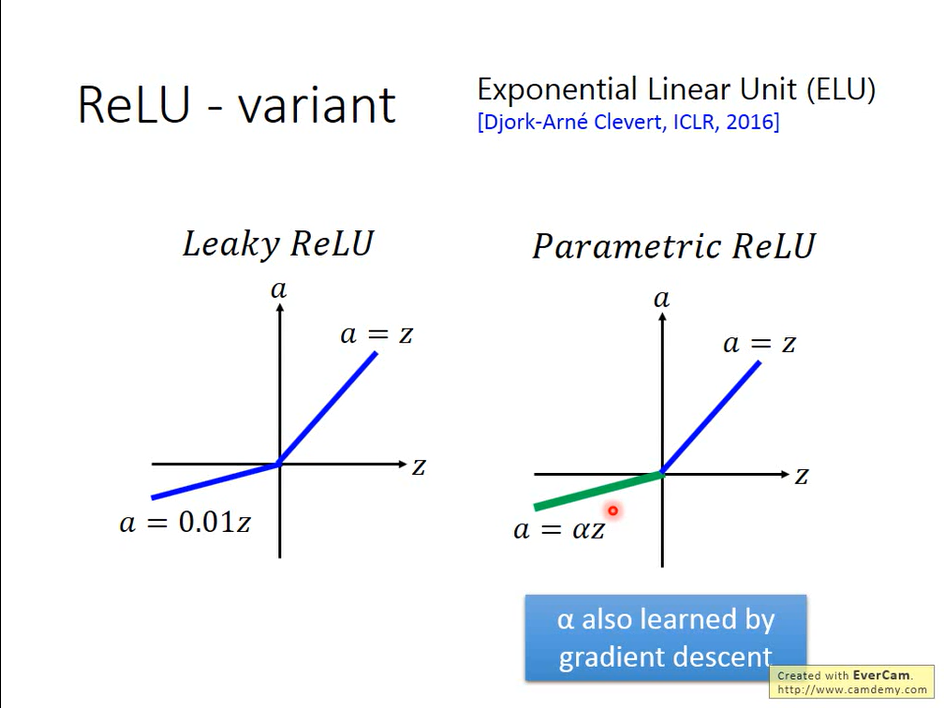


ruguoa

问题：如果neutral network变成了一个线性网络，那不是就不能很好的拟合复杂数据了吗？

不会，线性只是在一组输入training的时候是线性，但只要输入变化，就又会得到另外一组参数，线性的斜率和方向就改变了，所以整体而言是非线性的，小范围内是线性。

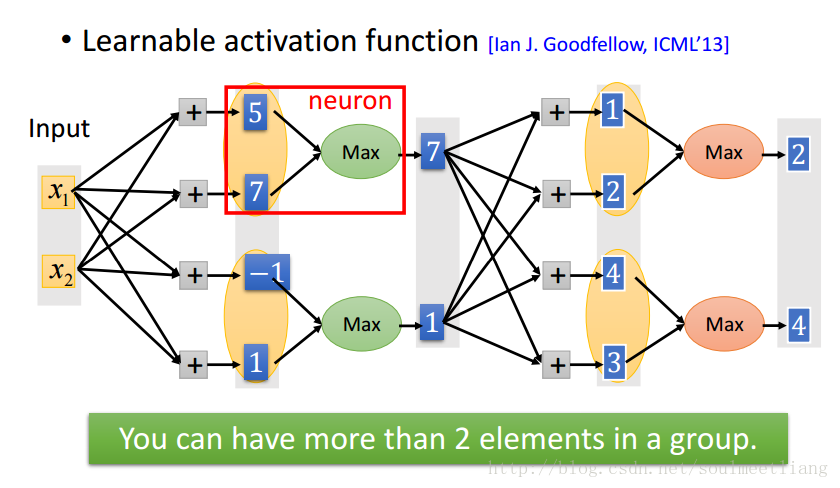
是否可以使用在Network intrusion中？



### Maxout Network(ReLU is a special case of maxout)

Learnable activation function, but need more data to training.

因为需要挑取每一层输出的最大的数。

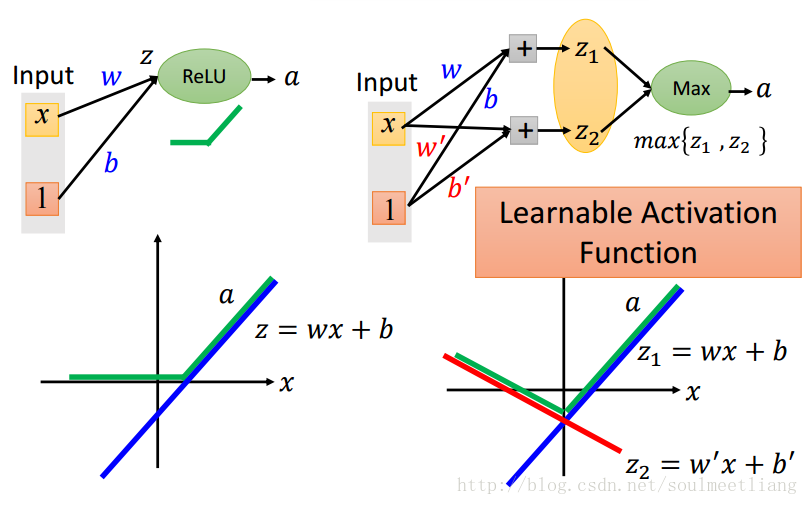


为什么要将x1、x2分别乘四个weight

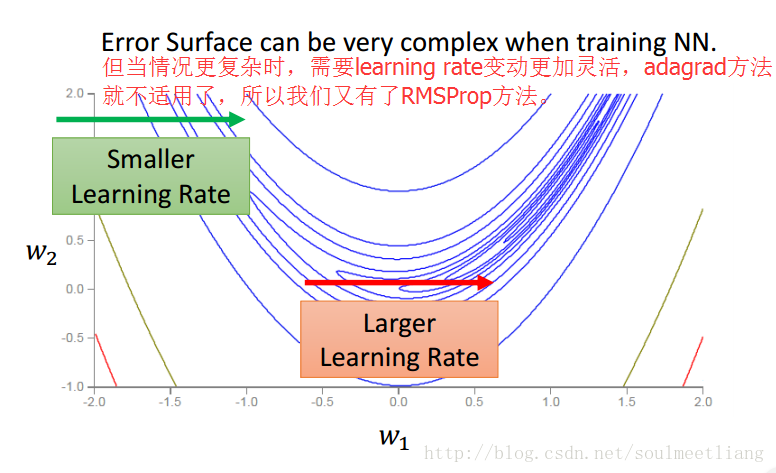
？

可以在多个元素中取最大值

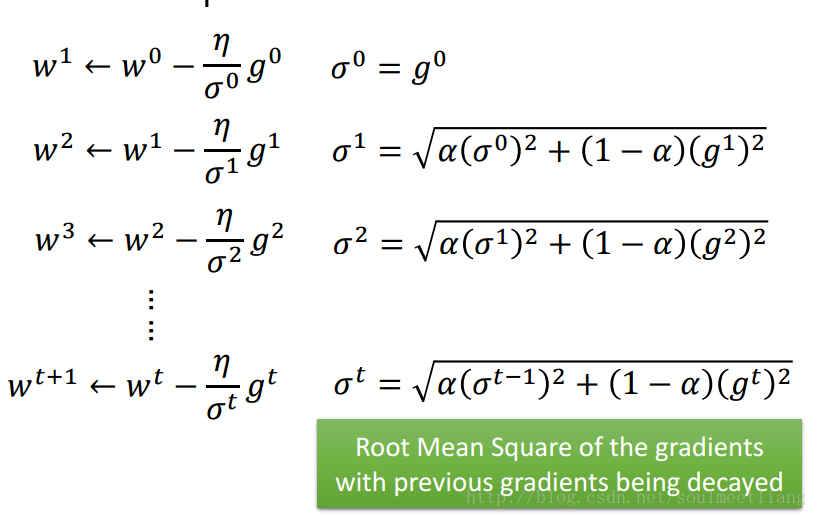
两个output但要乘所有的weight



Maxout is a universal approximator

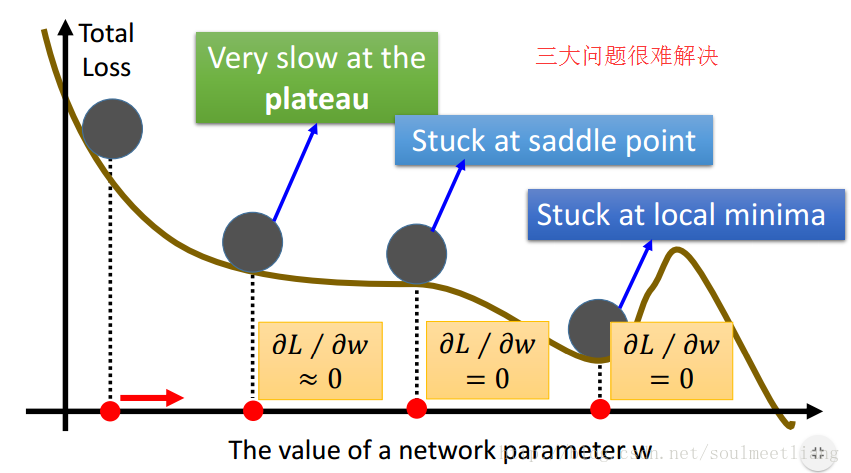


参数多维的情况下，loss function与参数的关系就画不出来了

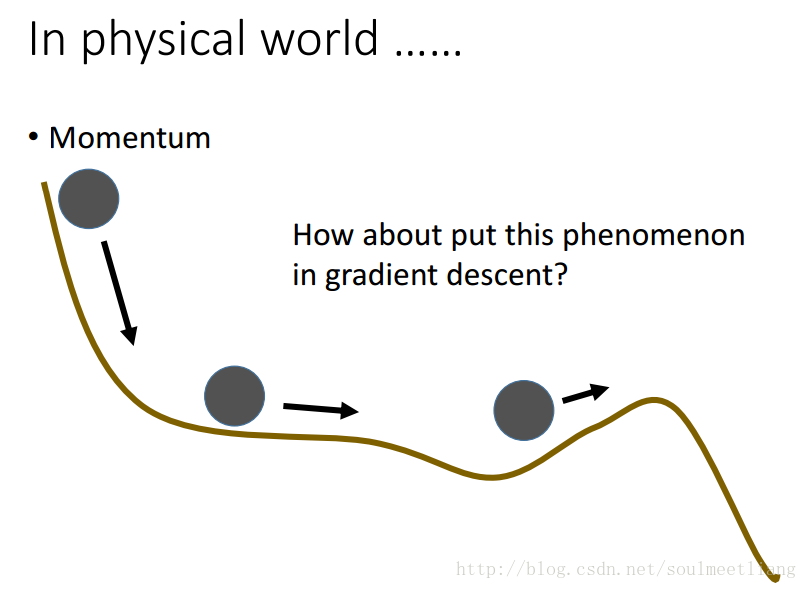
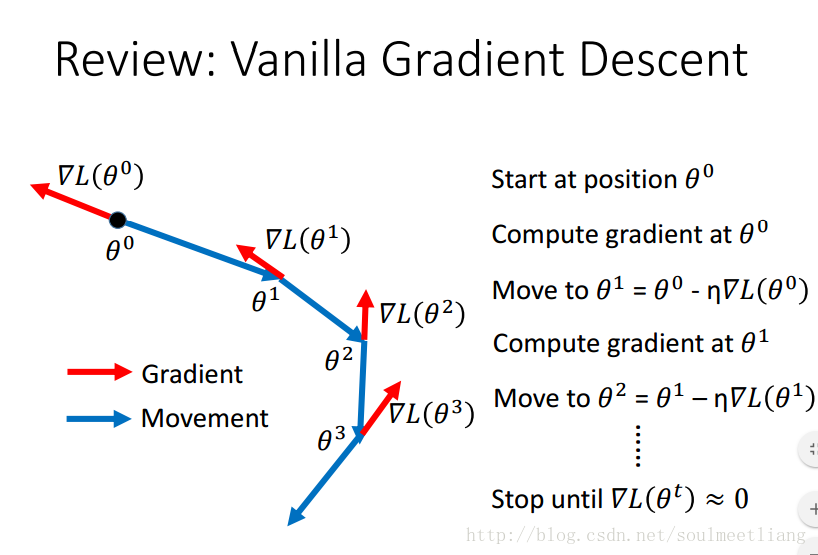


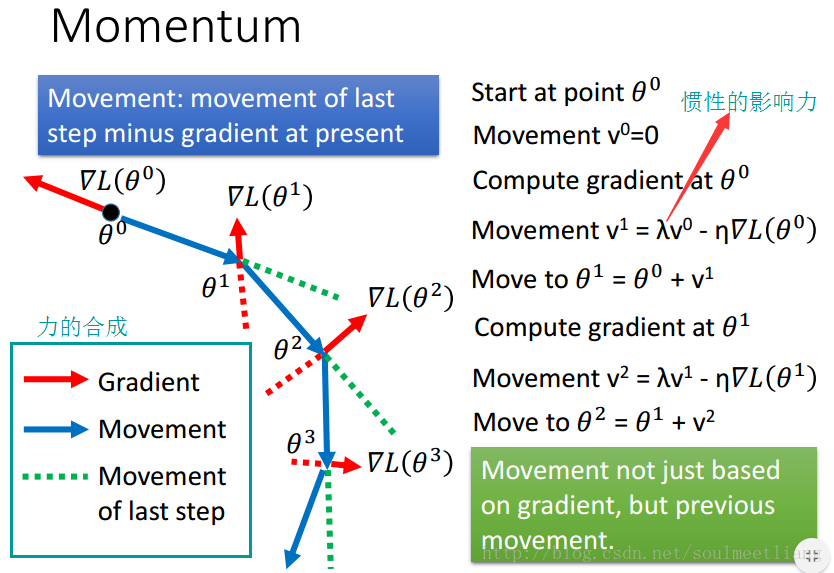
RMSprop可以算作Adadelta的一个特例：

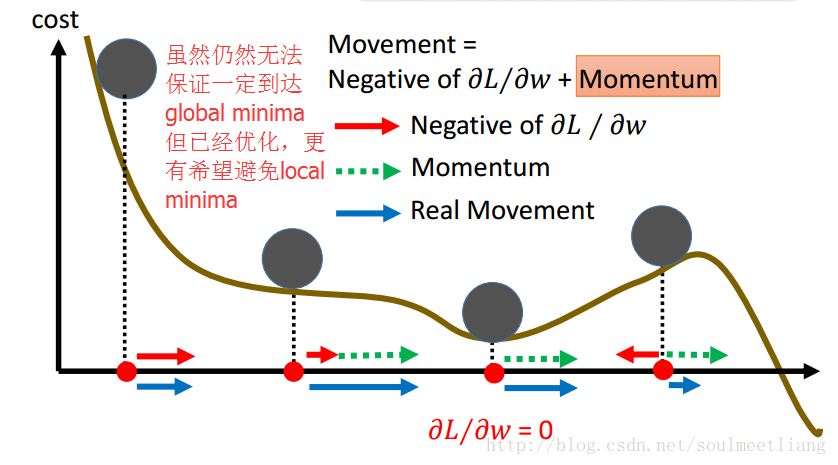
α一般设的较小，改变梯度累积为指数衰减的移动平均以丢弃遥远的过去历史。

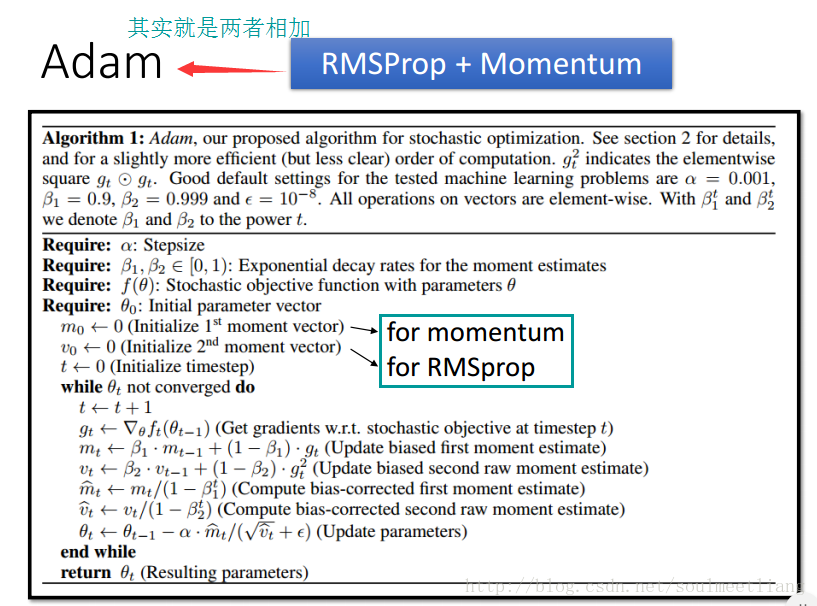


### Momentum（gradient descent 融入惯性作用）







### 空间范数

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D225/sign=1423af140ef3d7ca08f63874c71fbe3c/64380cd7912397dd2d45702e5b82b2b7d0a287d8.jpg

当p取

https://gss3.bdstatic.com/7Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D44/sign=ac59abd13cd12f2eca05af644ec2289d/d1160924ab18972b03b2cacceccd7b899e510a28.jpg

的时候分别是以下几种最简单的情形：

1-范数：║x║1=│x1│+│x2│+…+│xn│

2-范数：║x║2=（│x1│2+│x2│2+…+│xn│2）1/2

∞-范数：║x║∞=max（│x1│，│x2│，…，│xn│）

其中2-范数就是通常意义下的距离。

### 常用的三种p-范数(norm)推导出的[矩阵范数](https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5%E8%8C%83%E6%95%B0" \t "_blank)：

**1-范数：one norm**

║A║1 = max{ ∑|ai1|，∑|ai2|，……，∑|ain| } （列和范数，A每一列元素绝对值之和的最大值）（其中∑|ai1|第一列元素绝对值的和∑|ai1|=|a11|+|a21|+...+|an1|，其余类似）；

**2-范数：two norm**

║A║2 = A的最大奇异值 = (max{ λi(AH\*A) }) 1/2（谱范数，即A^H\*A[特征值](https://baike.baidu.com/item/%E7%89%B9%E5%BE%81%E5%80%BC)λi中最大者λ1的平方根，其中AH为A的转置[共轭矩阵](https://baike.baidu.com/item/%E5%85%B1%E8%BD%AD%E7%9F%A9%E9%98%B5" \t "_blank)）；

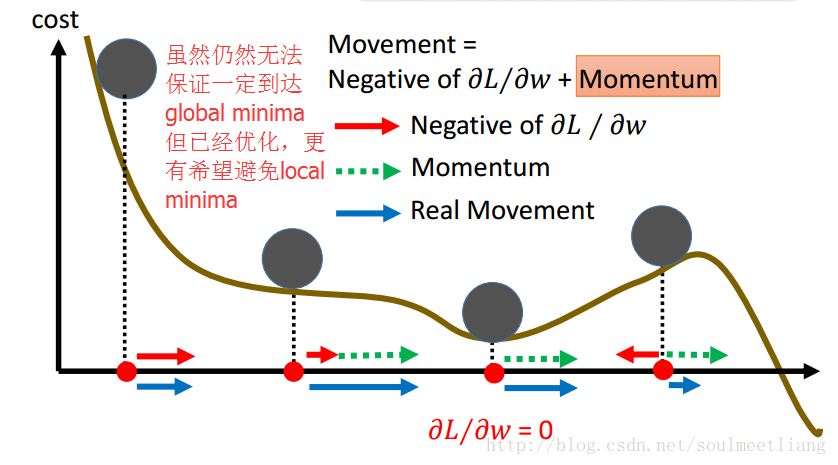
**∞-范数：**

║A║∞ = max{ ∑|a1j|，∑|a2j|,...，∑|amj| } （行和范数，A每一行元素绝对值之和的最大值）（其中∑|a1j| 为第一行元素绝对值的和，其余类似）；

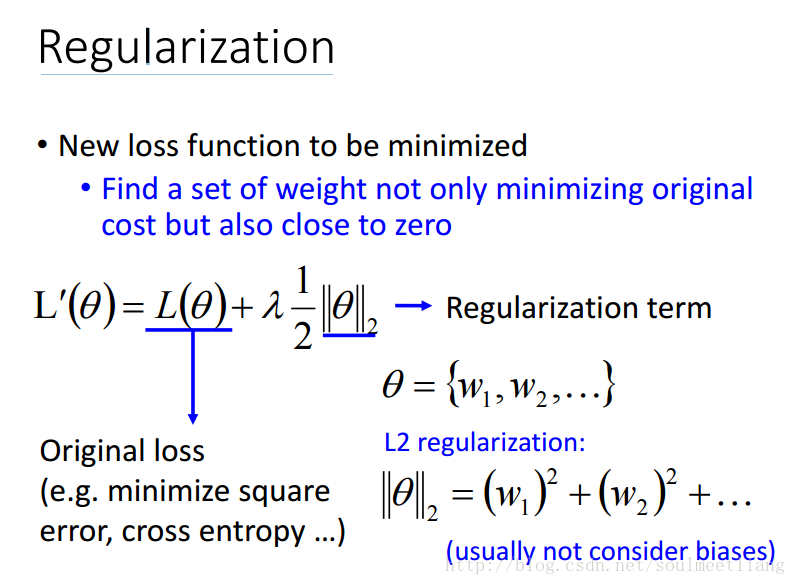
其它的p-范数则没有很简单的表达式。

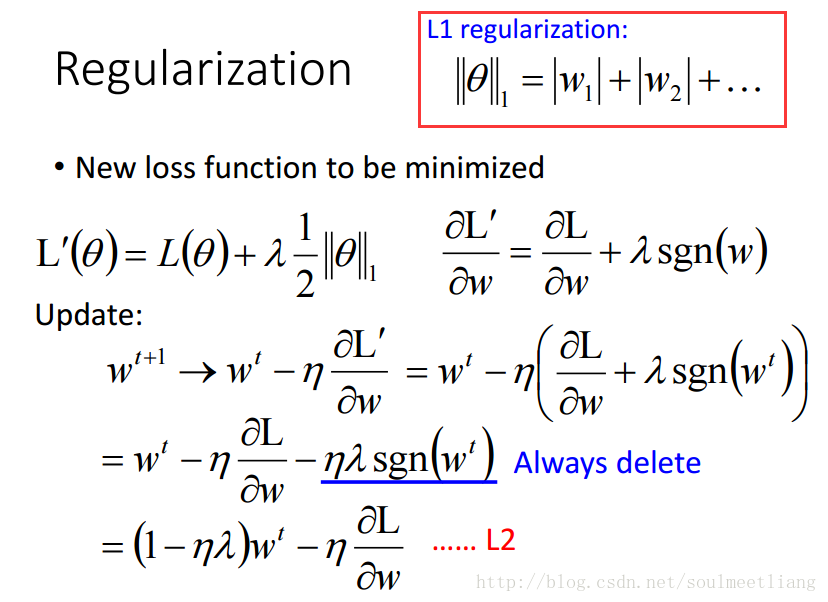
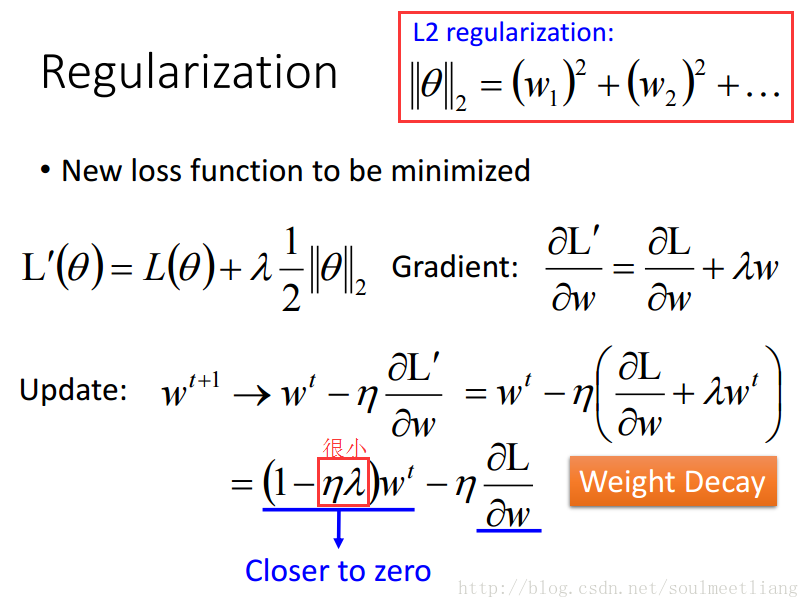
对于p-范数而言，可以证明║A║p=║AH║q，其中p和q是共轭指标。

简单的情形可以直接验证：║A║1=║AH║∞，║A║2=║AH║2，一般情形则需要利用║A║p=max{yH\*A\*x：║x║p=║y║q=1}。



## Regularization

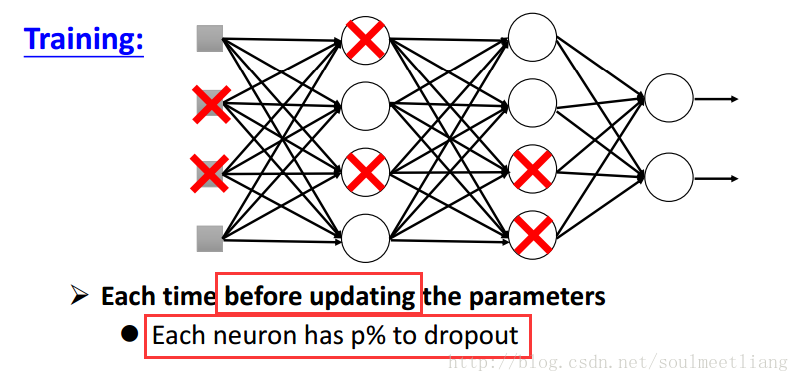


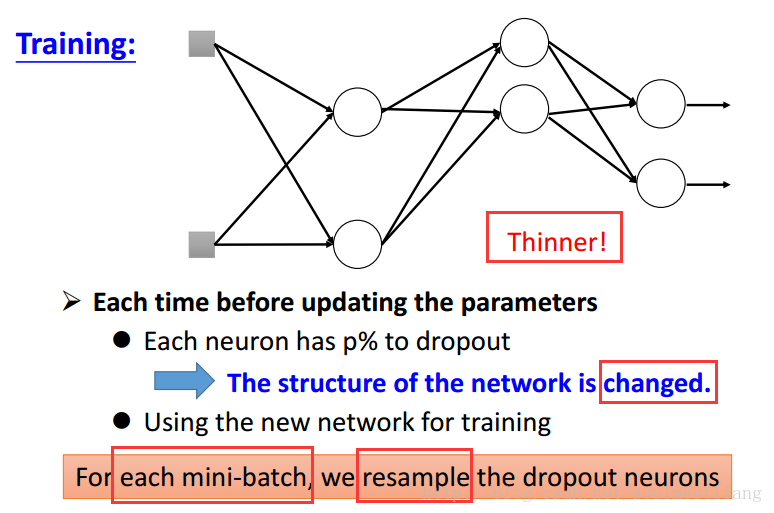


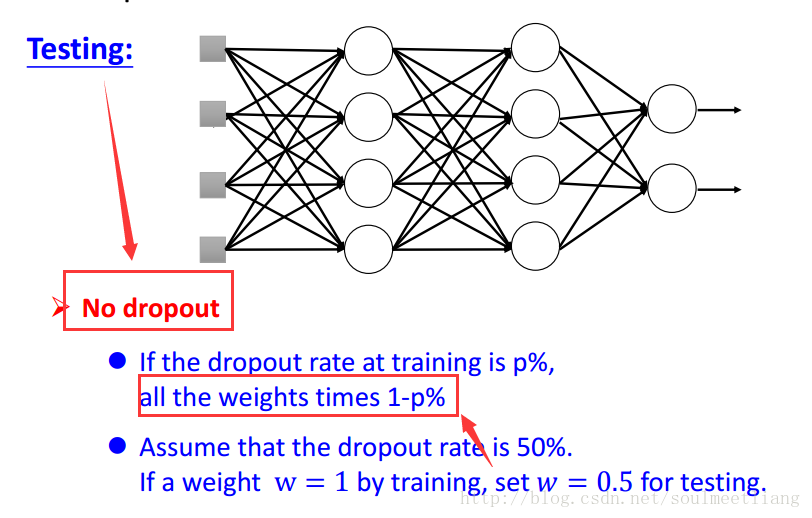
L2是二阶范数的平方,L1是一阶范数。

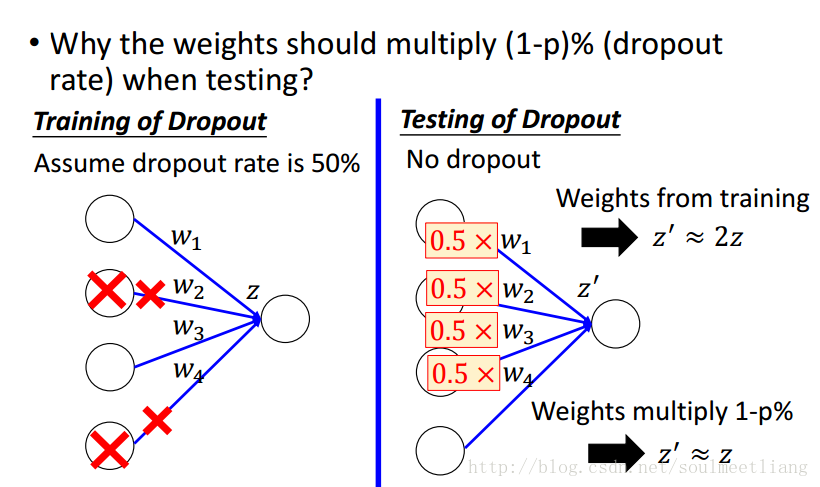
L2中w的值最后会趋近于0

L1中w的值会差的比较大









除了直觉上的解释之外。

比较靠谱的解释：

**(1)** 由于随机的让一些节点不工作了，因此可以避免某些特征只在固定组合下才生效，有意识地让网络去学习一些普遍的共性（而不是某些训练样本的一些特性）

**(2)** Bagging方法通过对训练数据有放回的采样来训练多个模型。而Dropout的随机意味着每次训练时只训练了一部分，而且其中大部分参数还是共享的，因此和Bagging有点相似。因此，Dropout可以看做训练了多个模型，实际使用时采用了模型平均作为输出

输出为什么要乘(1-p)

假设比例v=0.5，即在训练阶段，以0.5的比例忽略隐层节点；那么假设隐层有80个节点，每个节点输出值为1，那么此时只有40个节点正常工作；也就是说总的输出为40个1和40个0；输出总和为40；而在测试阶段，由于我们的权值已经训练完成，此时就不再按照0.5的比例忽略隐层输出，假设此时每个隐层的输出还是1，那么此时总的输出为80个1，明显比dropout训练时输出大一倍（由于dropout比例为0.5）；所以为了得到和训练时一样的输出结果，就缩减隐层输出为a（1-v）；即此时输出80个0.5，总和也为40.这样就使得测试阶段和训练阶段的输出“一致”了。v=0.4，留下的概率就是0.6，那在testing的时候，没有dropout，就是全留下，此时需要乘上（1-v），即0.6.

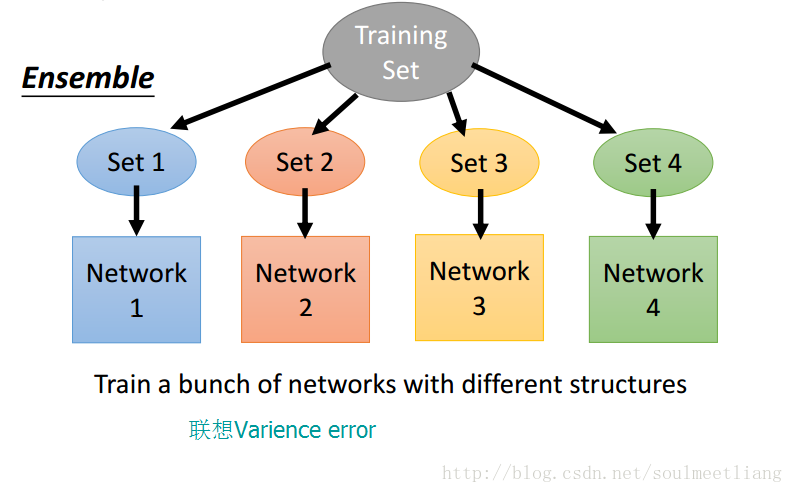
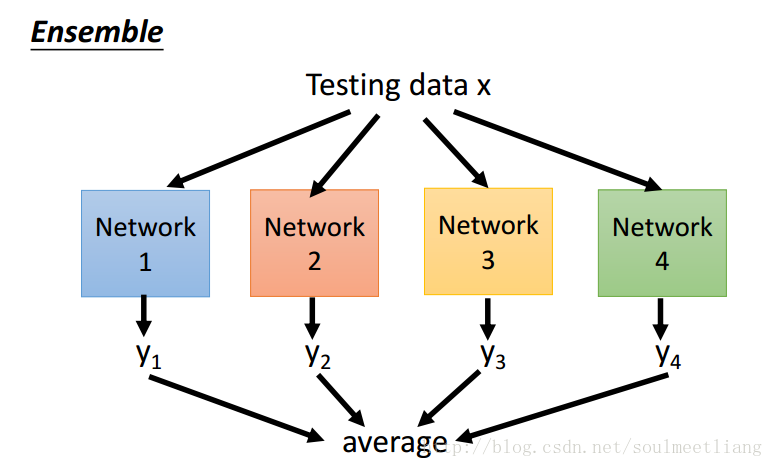
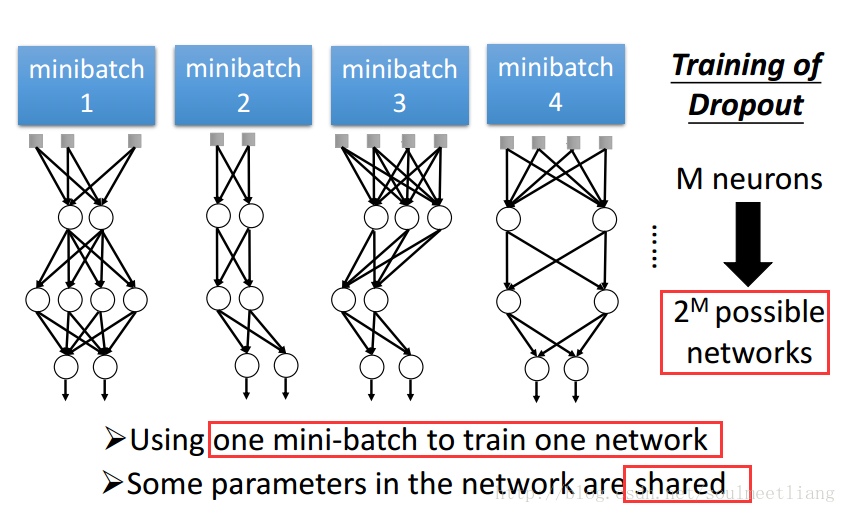
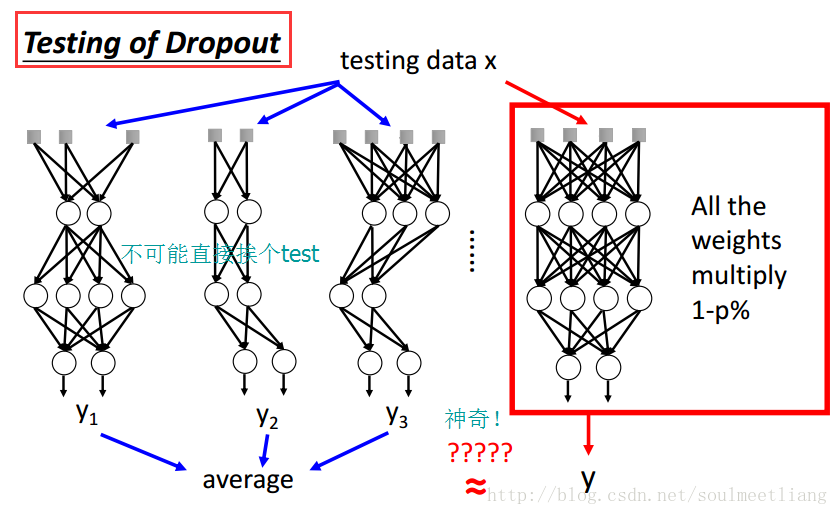
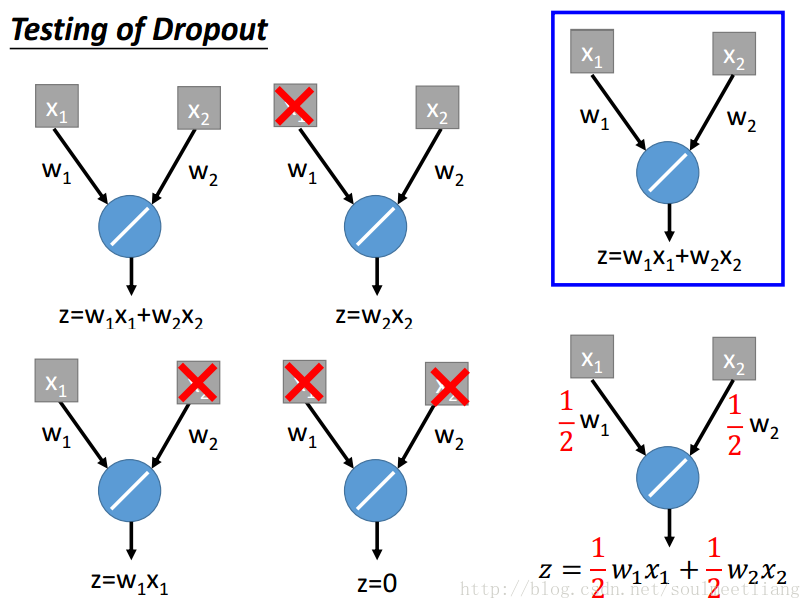
Inverted Dropout

在训练时输入的数据乘以放大系数1/(1-p)

该方法在训练阶段期间对激活值进行缩放，而测试阶段保持不变。

详细解析见[另外一篇文档](Machine%20Learning%209.docx)

### Dropout is a kind of ensemble

为了解决过拟合问题，一般会采用ensemble方法，即训练多个模型做组合，然后取模型的平均值，但耗费的时间却是训练2^n个神经网络的时间，dropout的效果和ensemble是近似的，但时间却只有2~3倍。    

似乎dropout只能用于linear network，但实际上非线性函数drop out也能work。