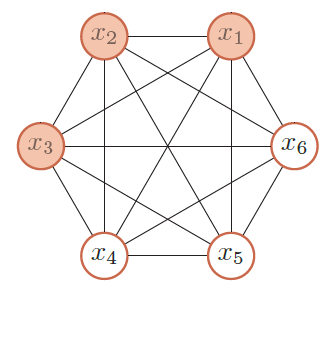
# 玻尔兹曼分布

能量收敛到最小后，热平衡趋于稳定，也就是说，在能量最少的时候，网络最稳定，此时网络最优。此时模型能量收敛到的分布，即为玻尔兹曼分布（吉布斯分布）。

# ****玻尔兹曼机（Boltzmann Machines，简称BM）****

玻尔兹曼机（BM）是由随机神经元全连接组成的反馈神经网络，且对称连接，由可见层、隐层组成，BM可以看做是一个无向图，如下图所示：

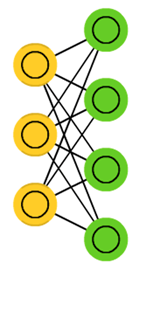


其中，x1、x2、x3为可见层，x4、x5、x6为隐层。

## 缺点

训练时间长

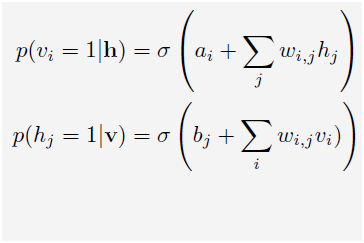
# ****受限玻尔兹曼机（Restricted Boltzmann Machines，简称RBM）****

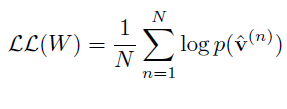


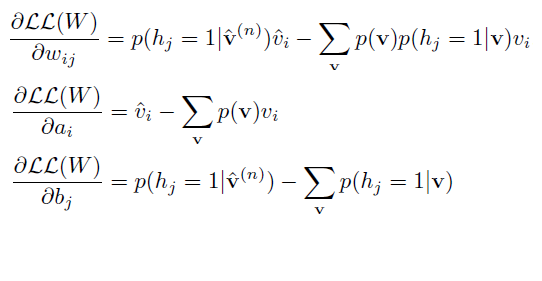
“受限玻尔兹曼机”（RBM）同样具有一个可见层，一个隐层，但层内无连接，层与层之间全连接，节点变量仍然取值为0或1，是一个二分图。也就是将“玻尔兹曼机”（BM）的层内连接去掉，对连接进行限制，就变成了“受限玻尔兹曼机”（RBM），这样就使得计算量大大减小，使用起来也就方便了很多。如上图。

“受限玻尔兹曼机”（RBM）的特点是：在给定可见层单元状态（输入数据）时，各隐层单元的激活条件是独立的（层内无连接），同样，在给定隐层单元状态时，可见层单元的激活条件也是独立的。

跟“玻尔兹曼机”（BM）类似，根据玻尔兹曼分布，可见层（变量为v，偏置量为a）、隐层（变量为h，偏置量为b）的概率为：

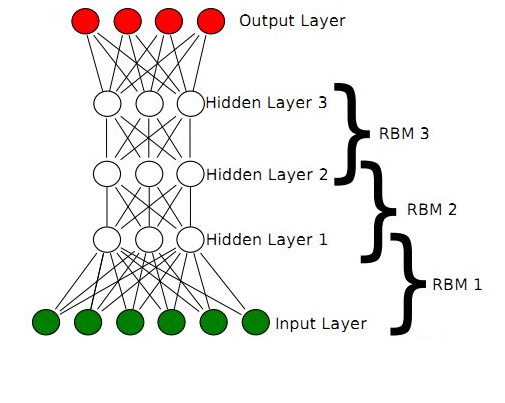
   
训练样本的对数似然函数为：

   
求导数：



RBM的本质是非监督学习的利器，可以用于降维（隐层设置少一点）、学习提取特征（隐层输出就是特征）、自编码器（AutoEncoder）以及深度信念网络（多个RBM堆叠而成）等等。

# ****深度信念网络（Deep Belief Network，简称DBN）****



在训练过程中，首先将可视向量值映射给隐单元；然后可视单元由隐层单元重建；这些新可视单元再次映射给隐单元，这样就获取新的隐单元。执行这种反复步骤叫做吉布斯采样。

<https://blog.csdn.net/u013631121/article/details/76652647>

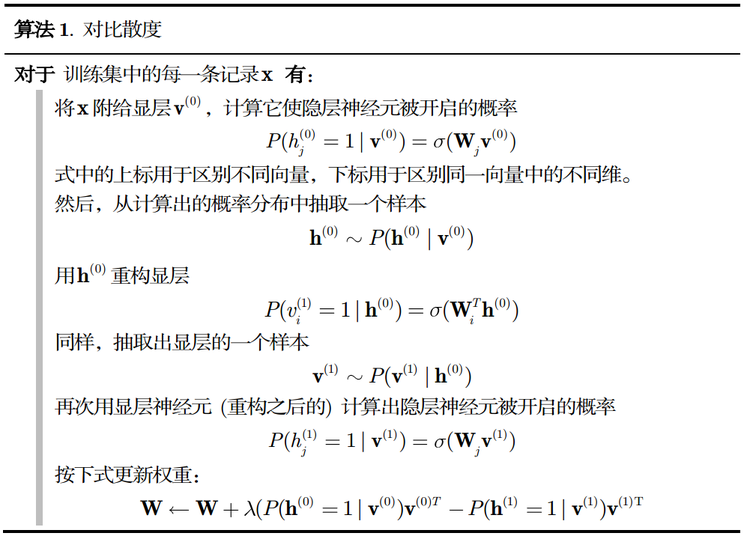
<https://blog.csdn.net/OPPOA113/article/details/42486833>

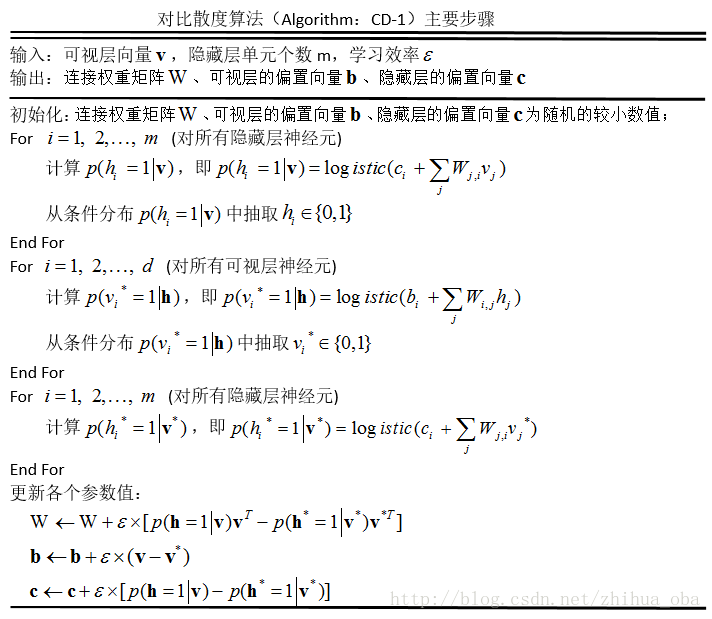
<https://blog.csdn.net/Rainbow0210/article/details/53010694>

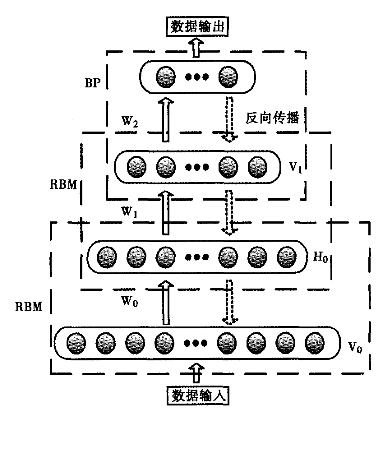
<https://blog.csdn.net/zhihua_oba/article/details/69487730>

<https://www.cnblogs.com/pythonlearing/p/9979161.html>

通俗理解：假设输入为X，输出为Y（已激活），则根据X求出Y，再根据Y求出X’（因为显层和隐层是双向连接的），再用X’求出Y’，Z为X与Y乘积，Z’为X’与Y’乘积，权重w更新为α（Z-Z’），隐层偏置b更新为α（Y-Y’），显层偏置a更新为α（X-X’）。







# 关键问题——为什么ISigmoid效果比Sigmoid好

选择Sigmoid函数作为激活函数的优势：1)可以引入非线性；2)容易求导；3)可以将实数压缩至(0,1)。

1. Sigmoid函数在压缩数据“幅度”方面有优势，对于深度网络，使用Sigmoid函数可以保证数据幅度不会有问题，幅度稳住后就不会有太大失误。
2. Sigmoid存在梯度消失的问题，在反向传播上有劣势。
3. ReLU不会对数据做幅度压缩，所以随着深度网络层数加深，数据的幅度会越来越大，最终影响模型的表现。
4. ReLU在反向传导时，能够将梯度信息“完完全全”地传递到浅层网络。

ReLU和LReLU不适用于DBN，因为它们无法充分发挥RBM预训练的作用。

因此在文章中结合了LReL和Sigmoid，在|x|>α时，fI’>fS’，权重变化值又与f’有关，则权重变化更快，收敛更快。同时减少了梯度消失的情况。

### 梯度消失

神经网络主要的训练方法是BP算法，BP算法的基础是导数的链式法则，也就是多个导数的乘积。而Sigmoid的导数最大为0.25，且大部分数值都被推向两侧饱和区域，这就导致大部分数值经过Sigmoid激活函数之后，其导数都非常小，多个小于等于0.25的数值相乘，其运算结果很小。随着神经网络层数的加深，梯度后向传播到浅层网络时，基本无法引起参数的扰动，也就是没有将loss的信息传递到浅层网络，这样网络就无法训练学习了。这就是所谓的梯度消失。

如果优化的话，可以思考如何优化激活函数。