正则化输入

一、过拟合问题

1.欠拟合与过拟合

泛化能力：在先前为观测到的输入数据上表现良好的能力
过拟合：将训练样本的一些特有的特点也当作潜在样本的一般性质，进而导致泛化能力下降。
欠拟合：训练样本性质没有学习完全，进而导致泛化能力较低。
决定机器学习算法效果的两个因素：
降低训练误差（对应欠拟合）
缩小训练误差和测试误差的差距（对应过拟合）
补充：欠拟合是指模型不能在训练集上获得足够低的误差，而过拟合是指训练误差和测试误差之间的差距太大

2.独立同分布假设

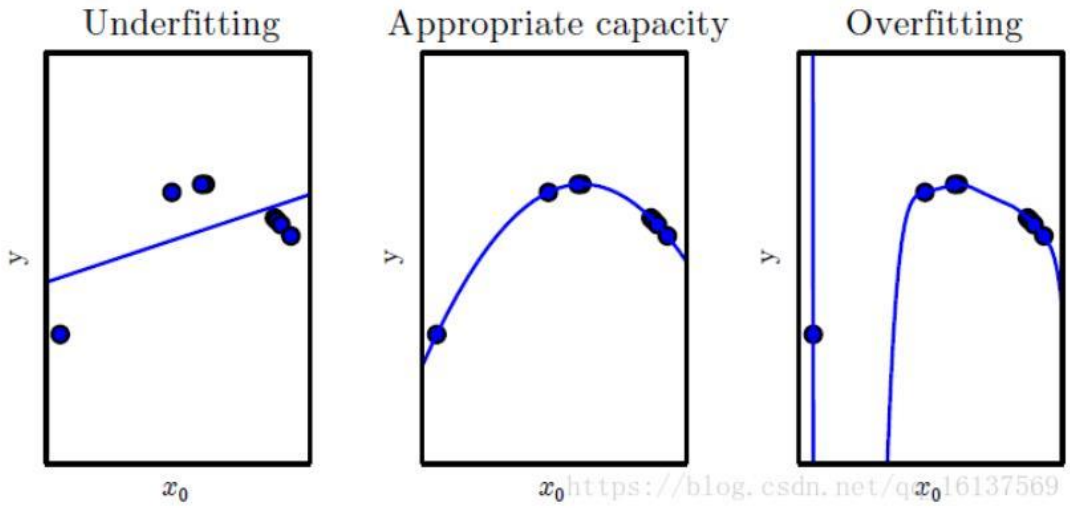
训练集和测试集数据通过数据集上被称为**数据生成过程**的概率分布生成，假设每个数据集中的样本都彼此相互独立，并且训练集和测试集都是同分布，采样自相同的分布，

我们将这个共享的潜在分布成为数据生成分布，即为 pdata，这就是**独立同分布假设**，

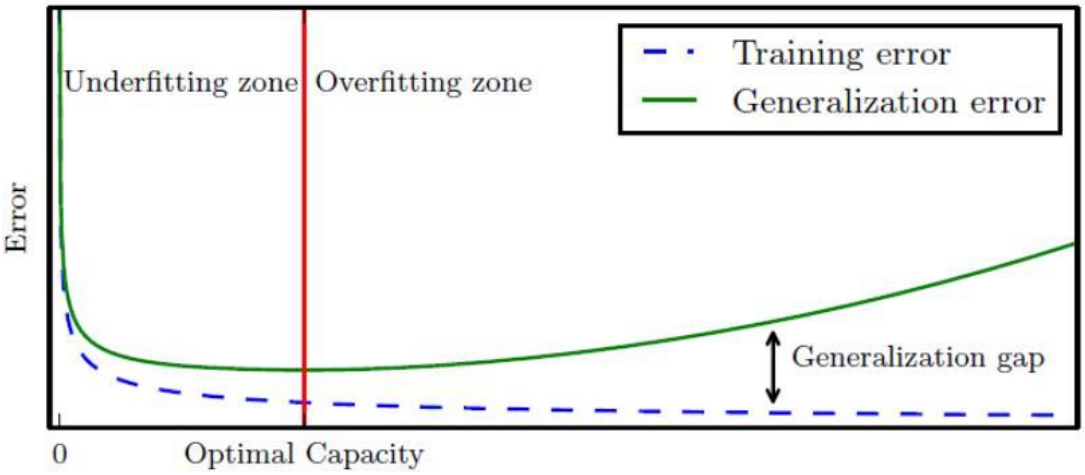
这使得我们能够用单个样本的概率分布表述数据生成过程。
在独立同分布的夹设下，训练样本的误差等于潜在样本的无查，只需要尽可能降低训练无查即可。（实际上假设基本不能成立，而并不影响使用该假设）

3.模型容量与过拟合

模型容量：指其拟合各种函数的能力。
我们可以通过调整模型的容量，控制模型是否偏向于过拟合与欠拟合。
过拟合、适当拟合、欠拟合：

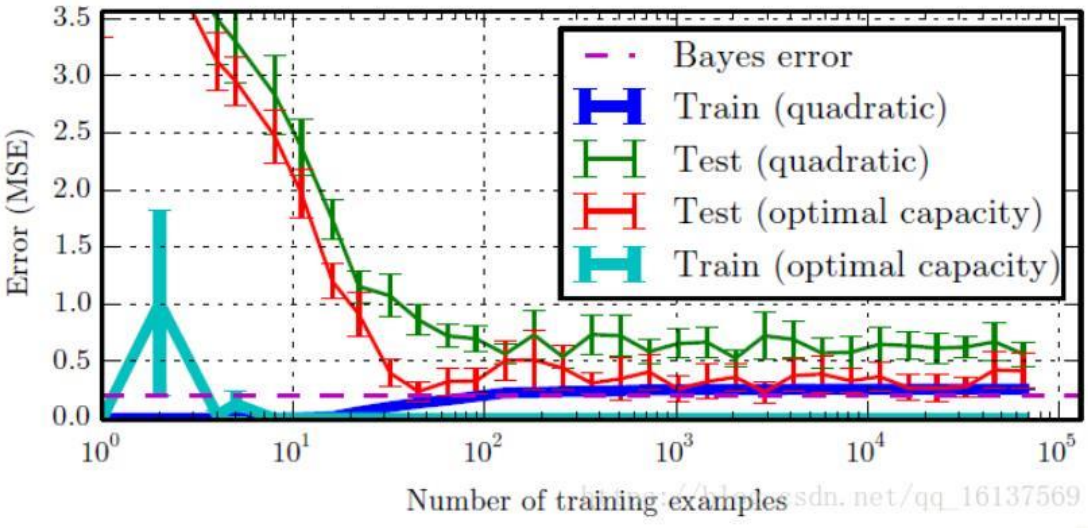


容量与误差之间典型关系：



综上：不能为了降低训练误差一味提高模型容量，需要根据具体问题选择合适的模型容量

训练样本数量对模型影响：

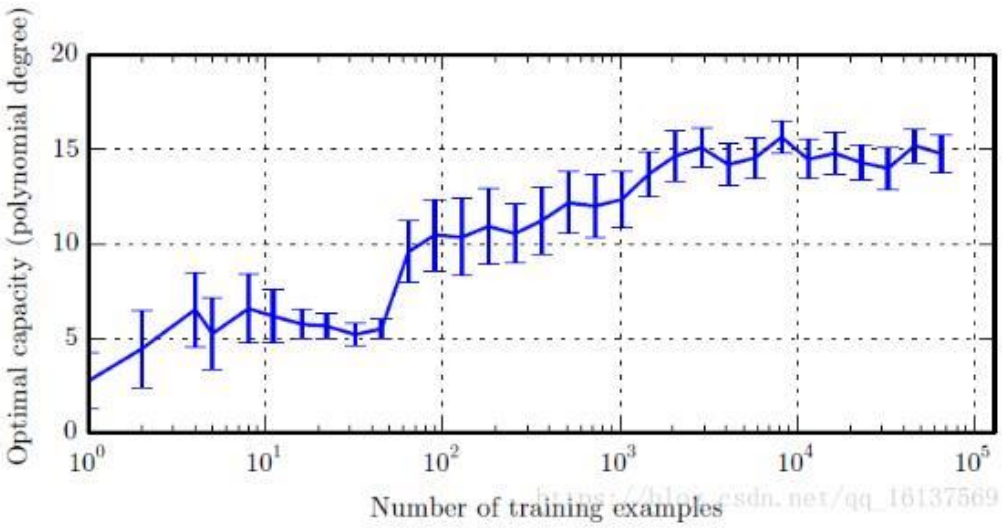


注：图中紫色的虚线是**贝叶斯误差**，即从预先知道的真实分布预测而出现的误差，也是理论上能达到的最佳误差。

对于二次模型，当训练样本数量还不足以匹配其模型容量时，训练误差会随着样本的增加而降低，但如果继续增加训练样本，超过了模型的拟合能力，训练误差（图中蓝色的线）会开始上升。而测试误差会随之减小，这是因为训练数据越多，关于训练数据的不正确的假设就越少。由于二次模型的容量不足以解决该问题，所以测试误差会稳定在一个较高的水平。

对于最优容量的模型，测试误差最终会趋近于贝叶斯误差。训练集误差可以低于贝叶斯误差，因为该模型有能力记住训练集中的样本。**但当训练集趋于无穷大时，任何固定容量的模型的训练误差都至少增至贝叶斯误差。**

训练集数量与最优容量之间关系：



即：训练集增大，最优容量随之增大，但当最优容量组足够捕获模型复杂度后就不再增长。

假设解决问题复杂度固定，当样本数量不够多时，训练集还不能很好的反应真是的数据分布 Pdata，每次增加训练集都引入了新的有用特征，模型发现现有模型容量不足以学习，故在初期模型的最优容量会随着训练集的增大而增大。若训练集增加到一定地步，这时训练集已经足够反应真实数据分布了，再增加训练集只是相当于多了一些重复的样本，所以模型的最优容量不会再增加了。

二、正则化

1.正则化简介:

正则化是选择模型的一种方法，模型可以堪称一个高维函数，模型参数确定，函数确定，及那个所有的可能得到的模型称为**假设空间。**

正则化可以帮助我们从假设空间中找到这样一个模型：训练误差较低，而且模型复杂度也较小。

2.常见正则化策略:

（1）参数范数惩罚

通过惩罚对目标函数 J 添加一个参数范数惩罚 Ω(θ)，限制模型的学习能力，将正则化的目标函数记为$\tilde{J}$

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)$$

其中α≥0 是权衡范数惩罚项 Ω 和标准目标函数 J(X;θ)相对贡献的超参数，通过调整 α 的大小，可以获得不同的参数选择偏好。

需要注意的一点是，参数包括模型中每一层仿射变换的权重和偏置，我们通常只对权重做惩罚，而不对偏置做正则惩罚。因为精确拟合偏置所需的数据通常比权重少的多，正则化偏置参数可能会导致明显的欠拟合。

L2 正则化：

L2 正则化也被称为权重衰减或岭回归，通过向目标函数添加一个L_2 范数平方项，使权重更加接近原点。

L2 正则化形式：

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha \frac{1}{2} \|w\|_2^2$$

分析L_2正则化对优化过程带来的效果：

假设没有偏置参数，模型的总目标函数：

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y)$$

与之对应的梯度：

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y)$$

使用单步梯度下降更新权重，学习率 $\epsilon \geq 0$：

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y))$$

即：$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y)$

与没有进行 L2 正则化的权重更新方式相比：

$$w \leftarrow w - \epsilon \nabla_w J(w; X, y)$$

加入 L2 正则项后，每次更新梯度前，都会先对权重向量进行收缩（权重收缩）

关于这种单步权重衰减方式对于整个训练过程带来的影响：

https://blog.csdn.net/qq_16137569/article/details/81584165

在能够显著减小目标函数方向上的参数会保留的相对完好，在无助于目标函数减小的方向上的参数会在训练中逐渐衰减掉。

L2 正则化能让学习算法“感知”到具有较高方差的输入 x，因此与输出目标的协方差较小的特征的权重将会收缩。这里高方差的输入就是与其他样本差异性大的意思。

L1 正则化：

L1 正则化也被称为 Lasso 回归，形式：

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha \|w\|_1$$

对应梯度：

$$\nabla_w \tilde{J}(\theta; X, y) = \nabla_w J(\theta; X, y) + \alpha \text{sign}(w)$$

L1 正则化对梯度的影响不再是线性地缩放每个 wi，而是添加了一项与 sign(wi)同号的常数。这种形式的梯度不一定能得到直接算术解。

由于L1惩罚项在完全一般化的Hessian情况下，无法得到清晰的代数表达式。如果我们假设问题中的数据已被预处理（比如PCA），去除了输入特征之间的相关性，那么Hessian矩阵是对角的，即 $H = \text{diag}([H_{1,1}, \ldots, H_{n,n}])$，其中每个$H_{i,i} \geq 0$。同式(5)一样对$J(\theta)$进行二次泰勒展开近似，并将L1正则化目标函数的二次近似分解成关于参数的求和：

$$\begin{aligned}\tilde{J}(w; X, y) &= J(w; X, y) + \alpha\|w\|_1 \\ &= J(w^*; X, y) + \frac{1}{2}(w - w^*)^T H(w - w^*) + \alpha\|w\|_1 \\ &= J(w^*; X, y) + \sum_i [\frac{1}{2} H_{i,i}(w_i - w_i^*)^2 + \alpha|w_i|]\end{aligned}$$

下面形式的解析解可以最小化这近似代价函数：

$$w_i = \text{sign}(w_i^*) \max\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\} \tag{10}$$

对每个i，考虑$w_i^* > 0$的情况，会有两种可能的结果：

当$w_i^* \leq \frac{\alpha}{H_{i,i}}$时，正则化中的$w_i$最优值是0；

当$w_i^* > \frac{\alpha}{H_{i,i}}$时，正则化不会将$w_i$的最优值推至0，而是在那个方向上移动$\frac{\alpha}{H_{i,i}}$的距离。

相比L2正则化，L1正则化会产生更稀疏的解。此处的稀疏性是指最优值中的一些参数为0。回顾式(7)，在L2正则化中，如果w_i^*不为零，那么$\tilde{w}_i$也不为0，这表明L2正则化不会使参数变得稀疏，而L1正则化有可能通过足够大的α实现稀疏。L2是对那些与多数特征不同的分量进行衰减，而L1则是舍弃那些低于某个标准的特征。这种稀疏性广泛用于特征选择机制，可以从可用的特征子集中选择出有意义的特征，化简机器学习问题。

L1正则化可通过假设权重w的先验分布为拉普拉斯分布，由最大后验概率估计导出.

除了L1，L2正则化，还有很多其他的正则化策略，因为相比L1和L2正则更容易理解，所以就不详细展开了。

（2）噪声注入：

模型容易过拟合的原因之一就是没有太好的抗噪能力，如果输入数据稍微改变一点点，就可能得到完全不一样的结果。正所谓“你害怕什么，就应该勇敢去面对什么”。最简单提高网络抗噪能力的方法，就是在训练中加入随机噪声一起训练。我们可以在网络的不同位置加入噪声：输入层，隐藏层，输出层。

数据集增强在某种意义上也能看做是在输入层加入噪声，通过随机旋转、翻转，色彩变换，裁剪等操作人工扩充训练集大小。这样可以使得网络对于输入更加鲁棒。

Dropout 方法属于在隐藏层中加入噪声的一种方式。这个因为比较重要，待会儿单独提出来讲。

对于输出层有人可能会有疑问，要知道我们手中数据集并不可能 100%保证标记正确，多多少少总会有一点错误率的。当然如果你能保证数据集没错误，不向输出层加入噪声也没关系。解决这个问题常见的办法是标签平滑，通过把确切的分类

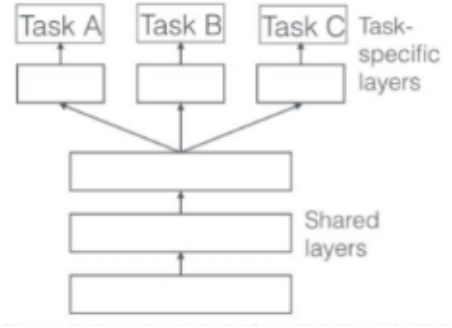
目标从 0 和 1 替换成 ϵk−1 和 1−ϵ，正则化具有 k 个输出的 softmax 函数的模型。

标签平滑的优势是能够防止模型追求确切的概率而不能学习正确分类。

从优化过程的角度来看，对权重叠加方差极小噪声等价于对权重是假范数惩罚，可以被解释为关于权重的贝叶斯推断的随即实现（这一点我不是很理解）。换个角度就很结论就很明朗了，因为我们在权重中添加了一些随机扰动，这鼓励优化过程找到一个参数空间，该空间对微小参数变化引起的输出变化影像很小。将模型送入了一个对于微小变化不敏感的区域，不仅找到了最小值，还找到了一个宽扁的最小值区域。

（3）多任务学习

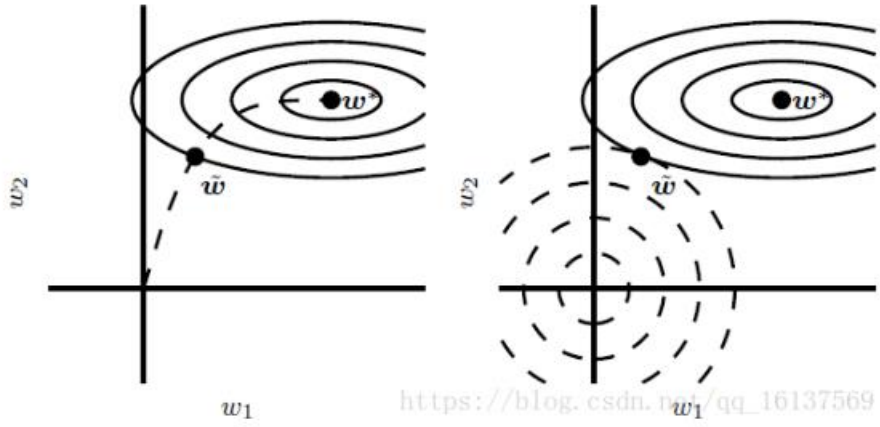
通过合并几个任务中的样例（可视为对参数使假的软约束）来提高泛化的一种方式。当模型的一部分被多个额外的任务共享时，这部分被约束为良好的值（若共享合理），通常会带来更好的泛化能力。



（4）提前终止

提前终止需要验证集损失作为观测指标，当验证集损失开始持续上升时，这时就该停止训练过程了。

具有正则化效果:



可以认为提前终止可以将优化过程的参数空间限制在初始参数θ_0的小邻域内。

（5）稀疏表示

惩罚网络中的激活单元，稀疏化激活单元，从而间接地对模型参数施加了复杂惩罚。

$$\underset{y \in \mathbb{R}^m}{\begin{bmatrix}18\\5\\15\\-9\\-3\end{bmatrix}} = \underset{A \in \mathbb{R}^{m\times n}}{\begin{bmatrix}4&0&0&-2&0&0\\0&0&-1&0&3&0\\0&5&0&0&0&0\\1&0&0&-1&0&-4\\1&0&0&0&-5&0\end{bmatrix}} \underset{x \in \mathbb{R}^n}{\begin{bmatrix}2\\3\\-2\\-5\\1\\4\end{bmatrix}}$$

$$\underset{y \in \mathbb{R}^m}{\begin{bmatrix}-14\\1\\19\\2\\23\end{bmatrix}} = \underset{B \in \mathbb{R}^{m\times n}}{\begin{bmatrix}3&-1&2&-5&4&1\\4&2&-3&-1&1&3\\-1&5&4&2&-3&-2\\3&1&2&-3&0&-3\\-5&4&-2&2&-5&-1\end{bmatrix}} \underset{h \in \mathbb{R}^n}{\begin{bmatrix}0\\2\\0\\0\\-3\\0\end{bmatrix}}$$

上式中第一个表达式是对参数进行惩罚的例子，第二个是对激活函数进行惩罚的例子，含有隐藏单元的模型在本质上都能变得稀疏。

（6）dropout

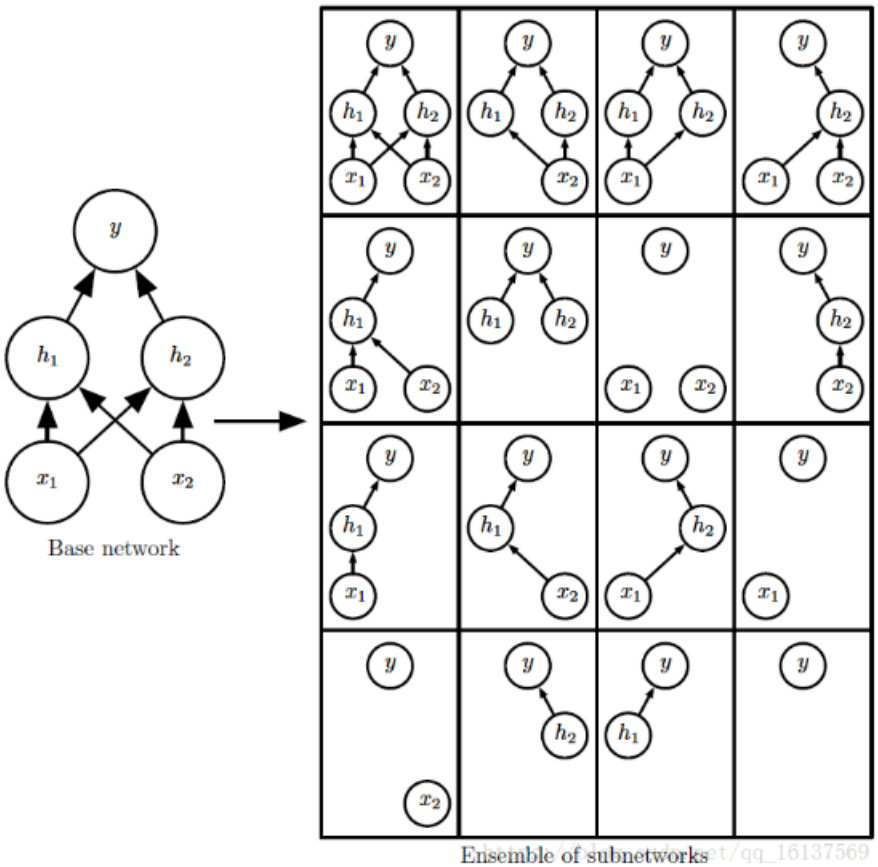
集成学习 bagging：通过结合几个模型降低泛化误差的技术，分别训练不同的模型，然后让所有模型表决测试样例的输出，也称**模型平均**。（有效的原因是不同的模型通常不会在测试集上产生完全相同的误差）

假设有 k 个回归模型，每个模型在每个例子上的误差为 ϵi，这个误差服从均值为 0，方差$\mathbb{E}[\epsilon_i^2] = v$且协方差$\mathbb{E}[\epsilon_i\epsilon_j] = c$的多维正态分布。通过所有集成模型的平均预测所得误差是

$$\mathbb{E}[(\frac{1}{k}\sum_i \epsilon_i)^2] = \frac{1}{k^2}\mathbb{E}[\sum_i (\epsilon_i^2 + \sum_{j\neq i} \epsilon_i\epsilon_j)]$$
$$= \frac{1}{k}v + \frac{k-1}{k}c$$

在误差完全相关，即 c = v 的情况下，$\mathbb{E}[(\frac{1}{k}\sum_i \epsilon_i)^2] = v$，模型平均对提升结果没有任何帮助；在误差完全不相关，即 c=0 的情况下，集成模型的误差期望减少到$\mathbb{E}[(\frac{1}{k}\sum_i \epsilon_i)^2] = \frac{1}{k}v$。**平均上，集成至少与它任何成员表现的一样好，并且如果成员误差是独立的，集成将显著地比其成员表现的更好。**神经网络中随机初始化差异、小批量的随机选择、超参数的差异或不同输出的非确定性实现往往足以使得集成中的不同成员具有部分独立的误差。

再考虑 dropout：



Dropout 提供了一种廉价的 Bagging 集成近似，能够训练和评估指数级数量的神

经网络。Dropout 是所有模型共享参数，每个模型集成父神经网络参数的不同子集。
隐藏单元经过 Dropout 训练后，它必须学习与不同采样神经元的合作，使得神经元具有更强的健壮性，并驱使神经元通过自身获取到有用的特征，而不是依赖其他神经元去纠正自身的错误。
Dropout 的另一个重要方面是噪声是乘性的。