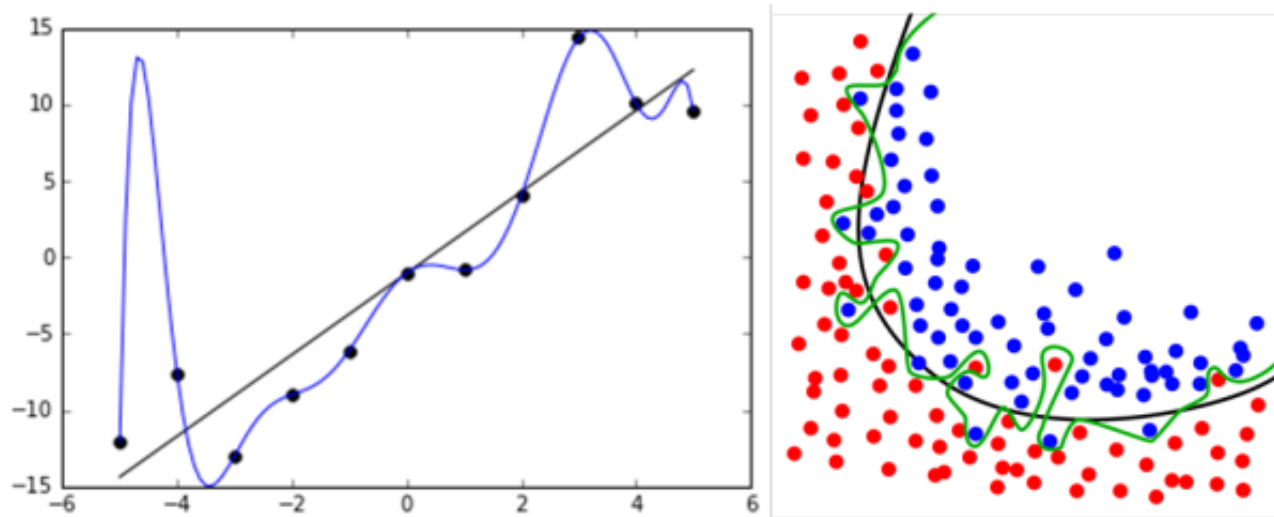


# 深度学习系列（11）：神经网络防止过拟合的方法

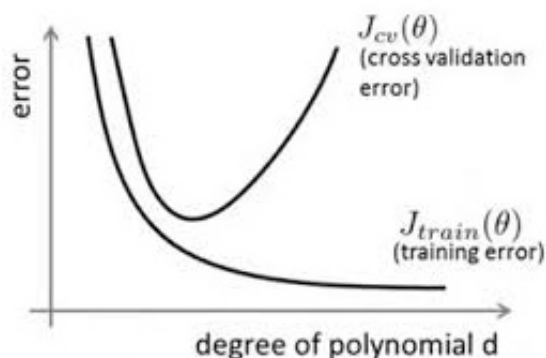
过拟合（overfitting）是指在模型参数拟合过程中的问题，由于训练数据包含抽样误差，训练时，复杂的模型将抽样误差也考虑在内，将抽样误差也进行了很好的拟合。具体表现就是最终模型在训练集上效果好，而在测试集上的效果很差，模型的泛化能力比较弱。



那为什么要解决过拟合现象呢？这是因为我们拟合的模型一般是用来预测未知的结果（不在训练集内），过拟合虽然在训练集上效果很好，但在实际使用时（测试集）效果很差。同时，在很多问题上，我们无法穷尽所有状态，不可能将所有情况都包含在训练集上。所以，必须要解决过拟合问题。

之所以过拟合在机器学习中比较常见，就是因为机器学习算法为了满足尽可能复杂的任务，其模型的拟合能力一般远远高于问题复杂度，也就是说，机器学习算法有“拟合出正确规则的前提下，进一步拟合噪声”的能力。

过拟合主要是有两个原因造成的：数据太少+模型太复杂。所以，我们可以通过使用合适复杂度的模型来防止过拟合问题，让其足够拟合真正的规则，同时又不至于拟合太多抽样误差。



通过上图可以看出，随着模型训练的进行，模型的复杂度会增加，此时模型在训练数据集上的训练误差会逐渐减小，但是在模型的复杂度达到一定程度时，模型在验证集上的误差反而随着模型的复杂度增加而增大。此时便发生了过拟合，即模型的复杂度升高，但是该模型在除训练集之外的数据集上却不work。

为了防止过拟合，我们需要用到一些方法，如下所示：

## 1. 获取更多数据

- 1.1 从数据源头获取
- 1.2 数据增强
- 1.3 根据当前数据集生成

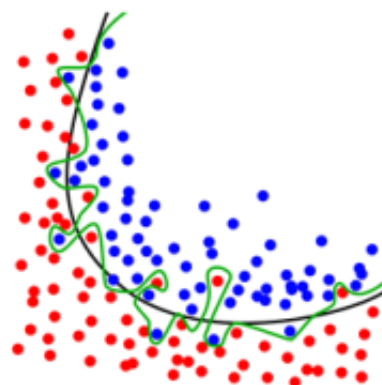
## 3. 结合多种模型

- Bagging
- Boosting
- Dropout

## 2. 使用合适的模型

- 2.1 网络结构 Architecture
- 2.2 训练时间 Early stopping
- 2.3 限制权值 Weight-decay
- 2.4 增加噪声 Noise

## 4. 贝叶斯方法



# 一、获取更多的数据

所有的过拟合无非就是训练样本的缺乏和训练参数的增加。一般要想获得更好的模型，需要大量的训练参数，这也是为什么CNN网络越来越深的原因之一，而如果训练样本缺乏多样性，那再多的训练参数也毫无意义，因为这造成了过拟合，训练的模型泛化能力相应也会很差。大量数据带来的特征多样性有助于充分利用所有的训练参数。

在数据挖掘领域流行着这样的一句话，“有时候往往拥有更多的数据胜过一个好的模型”。因为我们在训练数据训练模型，通过这个模型对将来的数据进行拟合，而在这之间又一个假设便是，训练数据与将来的数据是独立同分布的。即使用当前的训练数据来对将来的数据进行估计与模拟，而更多的数据往往估计与模拟地更准确。因此，更多的数据有时候更优秀。但是往往条件有限，如人力物力财力的不足，而不能收集到更多的数据，如在进行分类的任务中，需要对数据进行打标，并且很多情况下都是人工得进行打标，因此一旦需要打标的数据量过多，就会导致效率低下以及可能出错的情况。所以，往往在这时候，需要采取一些计算的方式与策略在已有的数据集上进行手脚，以得到更多的数据。通俗得讲，数据扩增即需要得到更多的符合要求的数据，

即和已有的数据是独立同分布的，或者近似独立同分布的。

如何获取更多的数据，一般有以下几个方法：

- 1) 从数据源头获取更多数据：这个是容易想到的，例如物体分类，我就再多拍几张照片好了；但是，在很多情况下，大幅增加数据本身就不容易；另外，我们不清楚获取多少数据才算够；
- 2) 根据当前数据集估计数据分布参数，使用该分布产生更多数据：这个一般不用，因为估计分布参数的过程也会代入抽样误差。
- 3) 通过一定规则扩充数据，即数据增强（Data Augmentation）。如在物体分类问题里，物体在图像中的位置、姿态、尺度，整体图片明暗度等都不会影响分类结果。我们就可以通过图像平移、翻转、缩放、切割等手段将数据库成倍扩充，以下为具体的方案：

## ⑤ 防止过拟合—Data Augmentation

- 这是解决过拟合最有效的方法，只要给足够多的数据，让模型「看见」尽可能多的「例外情况」，它就会不断修正自己，从而得到更好的结果
- 通过一定规则扩充数据。
  - Color Jittering：对颜色的数据增强：图像亮度、饱和度、对比度变化；
  - PCA Jittering：首先按照RGB三个颜色通道计算均值和标准差，再在整个训练集上计算协方差矩阵，进行特征分解，得到特征向量和特征值，用来做PCA Jittering；
  - Random Scale：尺度变换；
  - Random Crop：采用随机图像差值方式，对图像进行裁剪、缩放；包括Scale Jittering方法（VGG及ResNet模型使用）或者尺度和长宽比增强变换；
  - Horizontal/Vertical Flip：水平/垂直翻转；Shift：平移变换；Rotation/Reflection：旋转/仿射变换；
  - Noise：高斯噪声、模糊处理；
  - Label shuffle：类别不平衡数据的增广，参见海康威视ILSVRC2016的report；另外，文中提出了一种Supervised Data Augmentation方法

<https://zhuanlan.zhihu.com/p/23249000>

## 二、使用合适的模型

### 2.1 限制权值 Weight Decay

常用的weight decay有L1和L2正则化，L1较L2能够获得更稀疏的参数，但L1零点不可导。在损失函数中，weight decay是放在正则项（regularization）前面的一个系数，正则项一般指示模型的复杂度，所以weight decay的作用是调节模型复杂度对损失函数的影响，若weight decay很大，则复杂的模型损失函数的值也就大。

## ⑤ 防止过拟合—L1/L2正则

### ■ L1 regularization

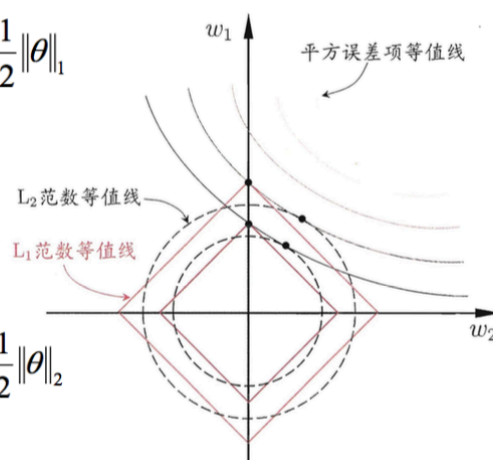
$$\|\theta\|_1 = |w_1| + |w_2| + \dots \quad L'(\theta) = L(\theta) + \lambda \frac{1}{2} \|\theta\|_1$$

$$\begin{aligned} w^{t+1} &\rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta \left( \frac{\partial L}{\partial w} + \lambda \operatorname{sgn}(w^t) \right) \\ &= w^t - \eta \frac{\partial L}{\partial w} - \eta \lambda \operatorname{sgn}(w^t) \quad \text{Always delete} \end{aligned}$$

### ■ L2 regularization

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \dots \quad L'(\theta) = L(\theta) + \lambda \frac{1}{2} \|\theta\|_2$$

$$\begin{aligned} w^{t+1} &\rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta \left( \frac{\partial L}{\partial w} + \lambda w^t \right) \\ &= \underbrace{(1 - \eta \lambda) w^t}_{\text{Closer to zero}} - \eta \frac{\partial L}{\partial w} \quad \text{Weight Decay} \end{aligned}$$



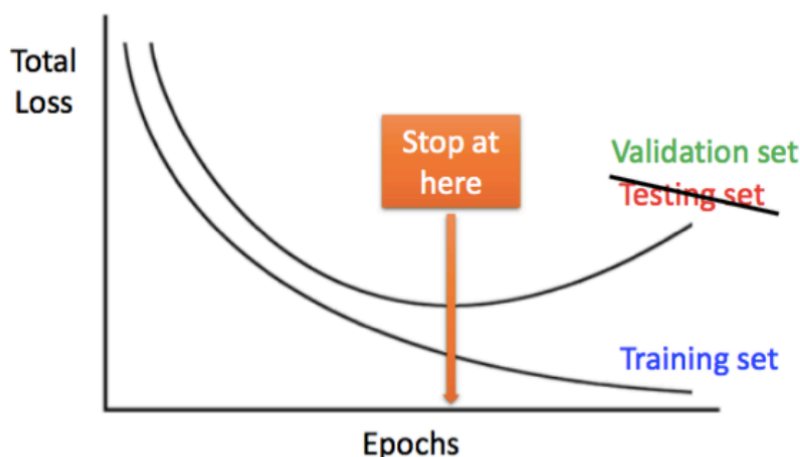
L1和L2正则化是很重要的过拟合方法，后边专门用一篇文章来讲。

## 2.2 训练时间 Early stopping

提前停止其实是另一种正则化方法，就是在训练集和验证集上，一次迭代之后计算各自的错误率，当在验证集上的错误率最小，在没开始增大之前停止训练，因为如果接着训练，训练集上的错误率一般是会继续减小的，但验证集上的错误率会上升，这就说明模型的泛化能力开始变差了，出现过拟合问题，及时停止能获得泛化更好的模型。如下图（左边是训练集错误率，右图是验证集错误率，在虚线处提前结束训练）：

## ⑤ 防止过拟合—Early Stopping

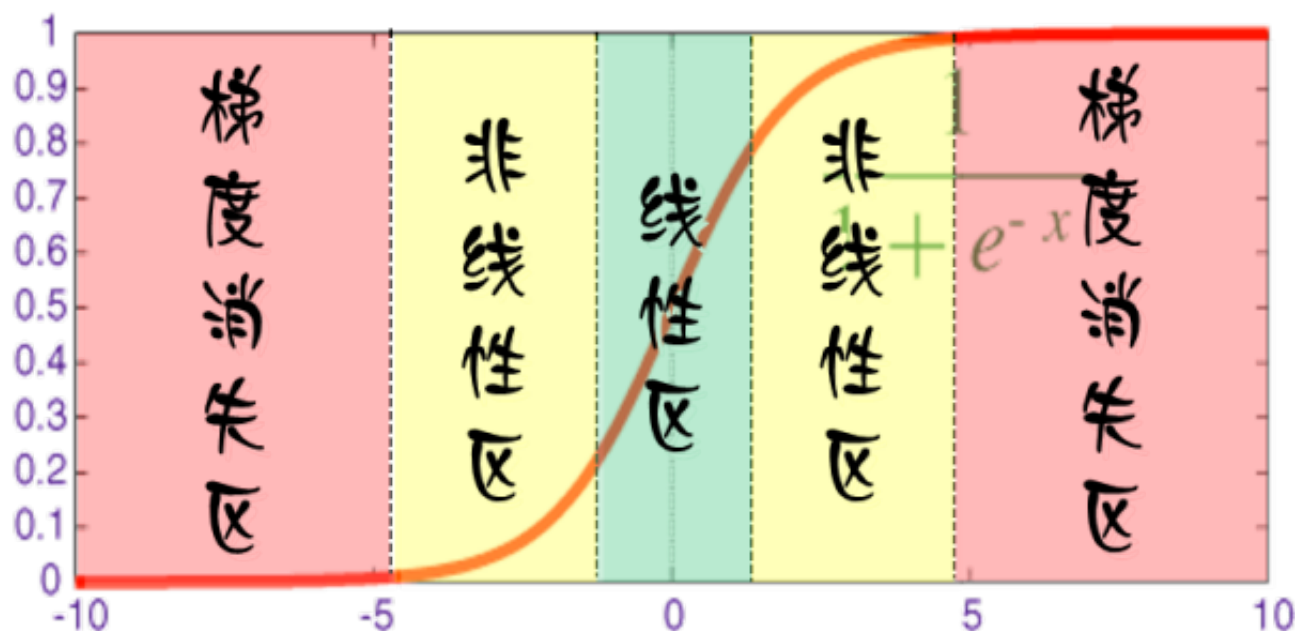
- 每一个epoch结束时（一个epoch即对所有训练数据的一轮遍历）计算 validation data的total loss，当loss不再降低时，就停止训练，这样可以防止overfitting。



Early stopping方法的具体做法是，在每一个Epoch结束时（一个Epoch集为对所有的训练数据的一轮遍历）计算validation data的accuracy，当accuracy不再提高时，就停止训练。这种做法很符合直观感受，因为accuracy都不再提高了，在继续训练也是无益的，只会提高训练的时间。那么该做法的一个重点便是怎样才认为validation accuracy不再提高了呢？并不是说validation accuracy一降下来便认为不再提高了，因为可能经过这个Epoch后，accuracy降低了，但是随后的Epoch又让accuracy又上去了，所以不能根据一两次的连续降低就判断不再提高。一般的做法是，在训练的过程中，记录到目前为止最好的validation accuracy，当连续10次Epoch（或者更多次）没达到最佳accuracy时，则可以认为accuracy不再提高了。此时便可以停止迭代了（Early Stopping）。这种策略也称为“No-improvement-in-n”，n即Epoch的次数，可以根据实际情况取，如10、20、30。

在神经网络中，对于每个神经元而言，其激活函数在不同的区间的性能是不同的：

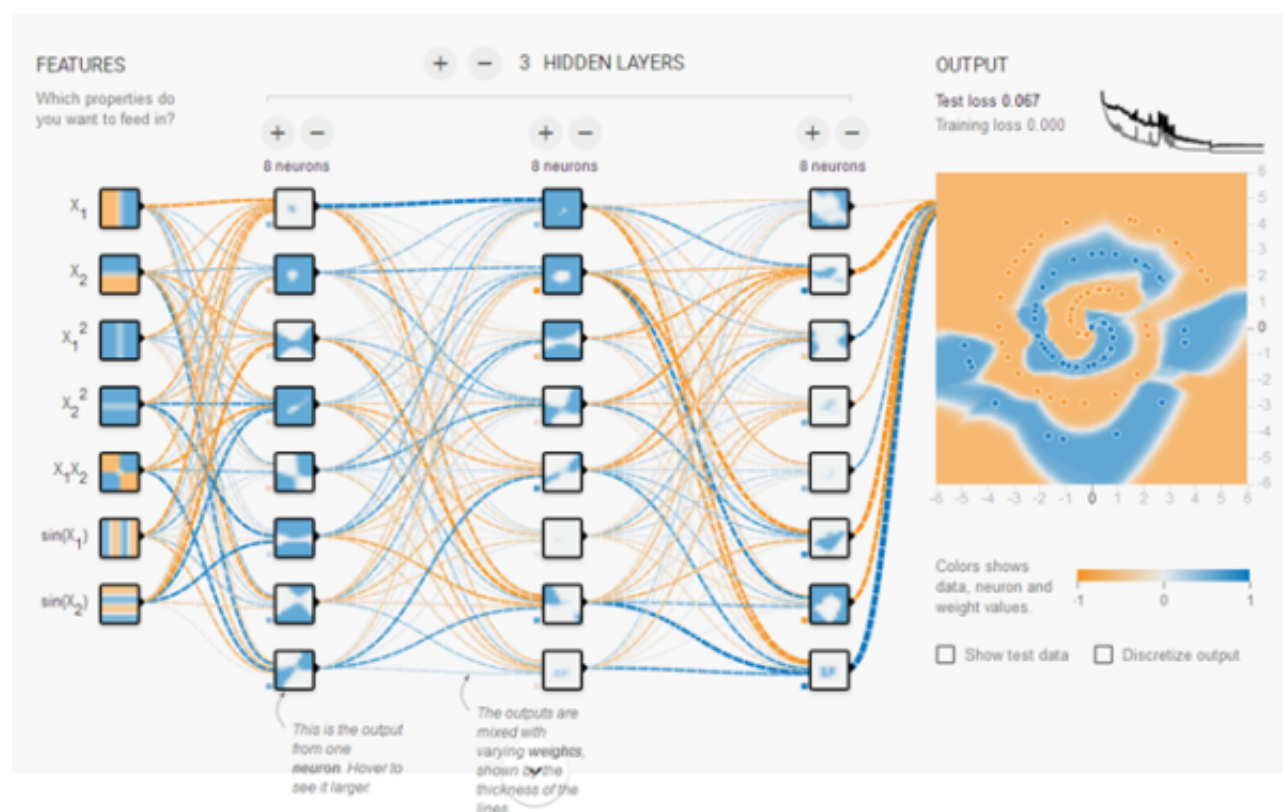




当网络权值较小时，神经元的激活函数工作在线性区，此时神经元的拟合能力较弱（类似线性神经元）。有了以上共识之后，就可以解释为什么训练时间（early stopping）有用：因为我们在初始化网络的时候一般都是初始为较小的权值。训练时间越长，部分网络权值可能越大。如果我们在合适时间停止训练，就可以将网络的能力限制在一定范围内。

## 2.3 网络结构

这个很好理解，减少网络的层数、神经元个数等均可以限制网络的拟合能力。



## 2.4 增加噪声

给网络加噪声也有很多方法：

### 2.4.1 在输入中加噪声

噪声会随着网络传播，按照权值的平方放大，并传播到输出层，对误差 Cost 产生影响。推导直接看 Hinton 的 PPT 吧：

$$\begin{aligned} \text{output on one case} &\longrightarrow y^{\text{noisy}} = \sum_i w_i x_i + \sum_i w_i \varepsilon_i \quad \text{where } \varepsilon_i \text{ is sampled from } N(0, \sigma_i^2) \\ E[(y^{\text{noisy}} - t)^2] &= E\left[\left(y + \sum_i w_i \varepsilon_i - t\right)^2\right] = E\left[\left((y - t) + \sum_i w_i \varepsilon_i\right)^2\right] \\ &= (y - t)^2 + E\left[2(y - t) \sum_i w_i \varepsilon_i\right] + E\left[\left(\sum_i w_i \varepsilon_i\right)^2\right] \\ &= (y - t)^2 + E\left[\sum_i w_i^2 \varepsilon_i^2\right] \quad \begin{array}{l} \text{because } \varepsilon_i \text{ is independent of } \varepsilon_j \\ \text{and } \varepsilon_i \text{ is independent of } (y - t) \end{array} \\ &= (y - t)^2 + \sum_i w_i^2 \sigma_i^2 \quad \text{So } \sigma_i^2 \text{ is equivalent to an L2 penalty} \end{aligned}$$

在输入中加高斯噪声，会在输出中生成  $\sum_i \sigma_i^2 w_i^2$  的干扰项。训练时，减小误差，同时也会对噪声产生的干扰项进行惩罚，达到减小权值的平方的目的，达到与 L2 regularization 类似的效果（对比公式）。

### 2.4.2 在权值上加噪声

在初始化网络的时候，用 0 均值的高斯分布作为初始化。Alex Graves 的手写识别 RNN 就是用了这个方法：

Graves, Alex, et al. "A novel connectionist system for unconstrained handwriting recognition." IEEE transactions on pattern analysis and machine intelligence 31.5 (2009): 855-868.

- It may work better, especially in recurrent networks (Hinton)

### 2.4.3 对网络的响应加噪声

如在前向传播过程中，让某些神经元的输出变为 binary 或 random。显然，这种有点乱来的做法会打乱网络的训练过程，让训练更慢，但据 Hinton 说，在测试集上效果会有显著提升（But it

does significantly better on the test set!) 。


## 三、结合多种模型

简而言之，训练多个模型，以每个模型的平均输出作为结果。

从  $N$  个模型里随机选择一个作为输出的期望误差  $\langle [(t - y_i)]^2 \rangle$ ，会比所有模型的平均输出的误差  $\langle [(t - \bar{y})]^2 \rangle$  大：

$$\begin{aligned}\bar{y} &= \langle y_i \rangle_i = \frac{1}{N} \sum_{i=1}^N y_i && i \text{ is an index over the } N \text{ models} \\ \langle (t - y_i)^2 \rangle_i &= \langle ((t - \bar{y}) - (y_i - \bar{y}))^2 \rangle_i \\ &= \langle (t - \bar{y})^2 + (y_i - \bar{y})^2 - 2(t - \bar{y})(y_i - \bar{y}) \rangle_i \\ &= (t - \bar{y})^2 + \langle (y_i - \bar{y})^2 \rangle_i - 2(t - \bar{y}) \langle (y_i - \bar{y}) \rangle_i\end{aligned}$$

this term  
vanishes



大概基于这个原理，就可以有很多方法了。

### 3.1 Bagging和Boost

简单理解，就是分段函数的概念：用不同的模型拟合不同部分的训练集。以随机森林（Rand Forests）为例，就是训练了一堆互不关联的决策树。但由于训练神经网络本身就需要耗费较多自由，所以一般不单独使用神经网络做Bagging。

bagging和boosting详细可见[机器学习算法系列（6）：AdaBoost](#)

### 3.2 Dropout

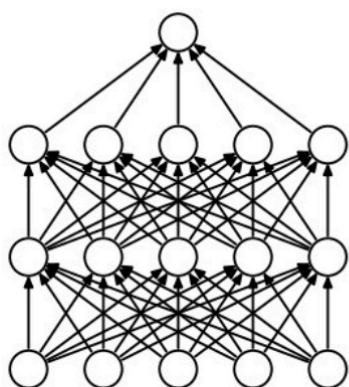
正则是在代价函数后面加上正则项来防止模型过拟合的。而在神经网络中，有一种方法是通过修改神经网络本身结构来实现的，其名为Dropout。该方法是在对网络进行训练时用一种技巧（trick），

Dropout是hinton最近2年提出的，源于其文章Improving neural networks by preventing co-adaptation of feature detectors.中文大意为：通过阻止特征检测器的共同作用来提高神经网络的性能。

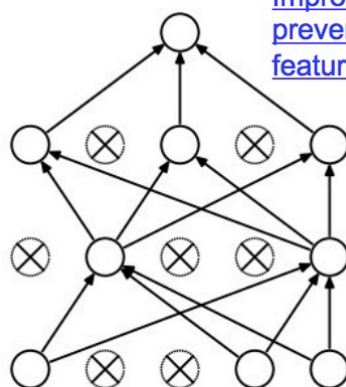


## ⑤ 防止过拟合—Dropout

Improving neural networks by preventing co-adaptation of feature Detectors Hinton



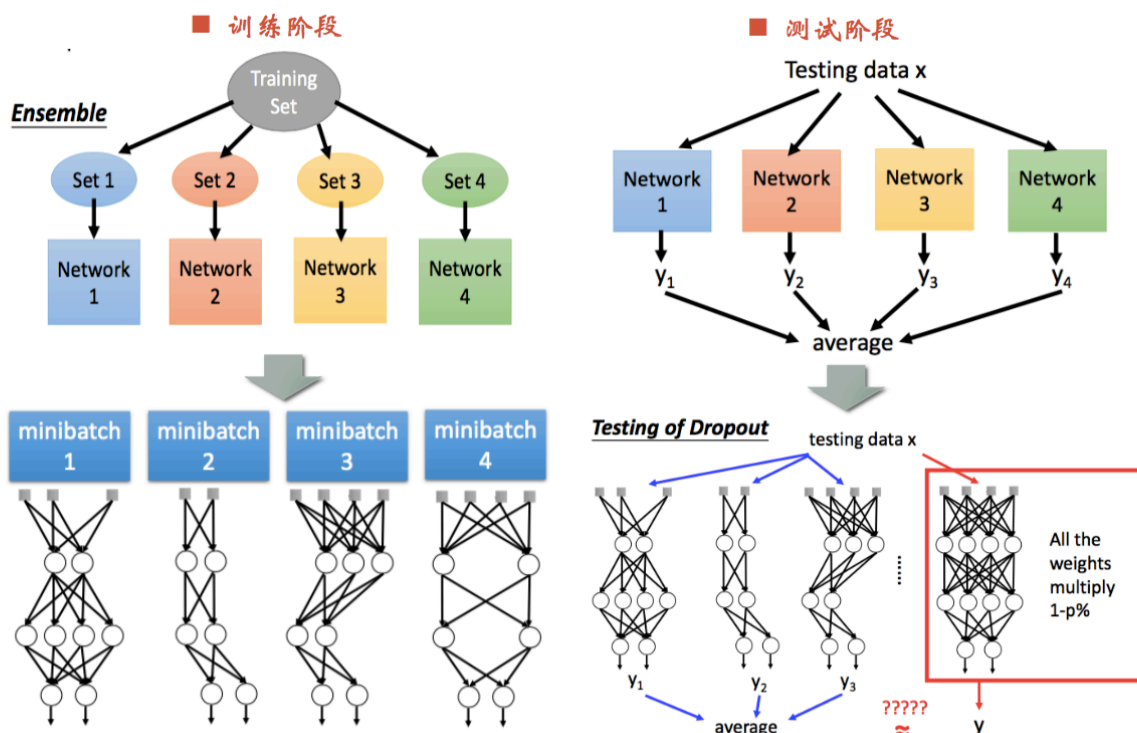
(a) Standard Neural Net



(b) After applying dropout.

- DNNs是以概率 $p$ 舍弃部分神经元，其它神经元以概率 $q=1-p$ 被保留，舍弃的神经元的输出都被设置为零。
- Dropout在实践中能很好工作是因为其在训练阶段阻止神经元的共适应

## Dropout的ensemble思想



在训练时，每次随机（如50%概率）忽略隐层的某些节点；这样，我们相当于随机从 $2^H$ 个模型中采样选择模型；同时，由于每个网络只见过一个训练数据（每次都是随机的新网络），所以类似 bagging 的做法，这就是我为什么将它分类到「结合多种模型」中；

此外，而不同模型之间权值共享（共同使用这  $H$  个神经元的连接权值），相当于一种权值正则方法，实际效果比 L2 regularization 更好。

正则化方法：L1和L2 regularization、数据集扩增、dropout