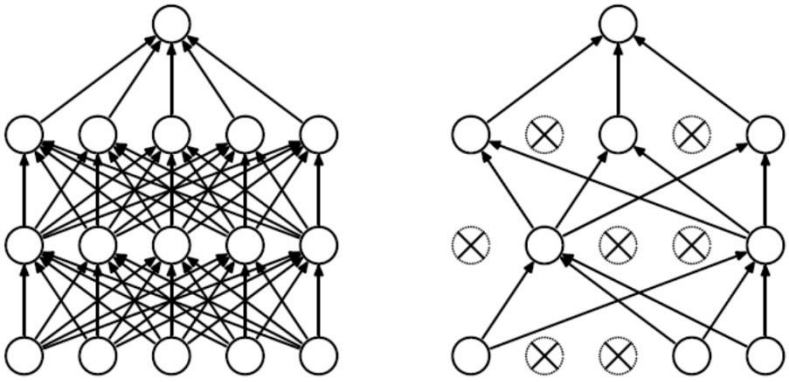
基本做法

Vanilla Dropout

训练阶段：



(a) Standard Neural Net (b) After applying dropout.

在训练每个 batch 时，让神经元以一定的概率 p“失活”得到新的网络模型，进行前向传播，然后把得到的损失进行反传，在没有被“失活”的神经元上按照随机梯度下降法更新相应的参数。

预测阶段：

因为我们训练的时候会随机的丢弃一些神经元，但是预测的时候就没办法随机丢弃了。如果丢弃一些神经元，这会带来结果不稳定的问题，也就是给定一个测试数据，有时候输出 a 有时候输出 b，结果不稳定，这是实际系统不能接受的，用户可能认为模型预测不准。那么一种“补偿”的方案就是每个神经元的权重都乘以一个 p，这样在“总体上”使得测试数据和训练数据是大致一样的。比如一个神经元的输出是 x，那么在训练的时候它有 p 的概率参与训练，(1-p)的概率丢弃，那么它输出的期望是 px+(1-p)0=px。因此测试的时候把这个神经元的权重乘以 p 可以得到同样的期望。

Inverted Dropout

是将所有的修改过程放在训练阶段，在前向传播的时候先“失活”再对输入进行 $\frac{1}{1-p}$ 倍的放大，保证总期望不变预测阶段不做额外的处理。

基本原理

通过随机失活网络隐藏层中的神经元，一来由于每次选择失活的对象不同，所以即使对于相同的输入得到的结果也不同，这样就相当于在网络中引入了噪声，可以防止网络发生过拟合。二来使得权重的更新不再依赖于固定关系的隐含节点的共同作用，减少了特征之间的相互依赖，从而使得模型去学习更加通用的特征。

应用

做分类时一般将 Dropout 层加在全连接层后面防止网络过拟合。