# 一 超参数调节经验

## 1.1 数据集重要性

应用深度学习是一个典型的迭代过程，需要多次循环往复，才能为应用程序找到一个称心的神经网络，因此循环该过程的效率是决定项目进展速度的一个关键因素，而创建高质量的训练数据集，验证集和测试集也有助于提高循环效率。

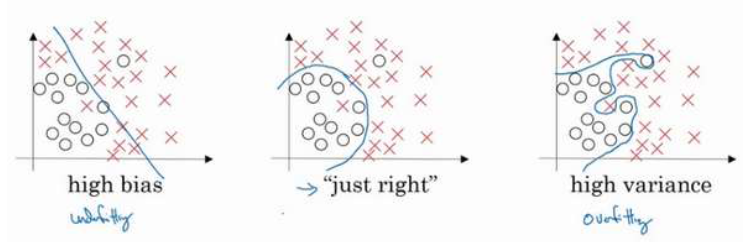
## 1.2 数据集尽可能相同分布

因为你们要用验证集来评估不同的模型，尽可能地优化性能。如果验证集和测试集来自同一个分布就会很好。

比如用清晰的图做训练集, 而用比较模糊,含糊的图做eval集,就会大打折扣.

二 偏差(bias)和方差(variance)

目的是弄清偏差和方差的权衡.



### 2.1 偏差和方差的度量标准.



以上图为例, 猫是正类. 前提是人分辨错误的概率是0%.

1. 如果train set误差是1%, 而eval set误差是16%. 则认为该分类器的方差很大.
2. 如果train set误差15%,eval set误差16%,则方差大同时偏差很大,更认为偏差大, 没有在train上收敛.
   1. 特别地, 如果人分辨误差是15%, 则可以认为trainset还不错, 相对才查1%.
3. 如果train set 误差0.5%, evalset误差是1%.则认为效果很好.

### 2.2 方差偏差的基本解决思路—正则化

对于高偏差的情况,

1. 选择新网络, 更多layers.更多chns.
2. 更多时间来train
3. 先进的优化算法.

如何对偏差和方差的权衡

首先, 需要满足偏差要小, 至少要满足在trainset上的误差要小.

然后, 如果偏差满足一定阈值后, 检查方差情况. 如果方差过大, 即过拟合.

1. 可以增加数据,更多数据.
2. 添加正则化项.

1.3 正则化

它解决如下问题: 方差大(过拟合). 增大数据可以降低方差, 如果不能增大数据, 可以加正则化, 可能会损失一些偏差,但是损失的并不大.

#### 逻辑回归的正则化



拉姆达是正则化参数. 对w做正则化.wTw是向量w 欧几里德范数的平方(2范数), 此处是L2正则化, 又称为w的L2范数.



|w|成为向量w的L1范数.

L1的缺点, 使w更加稀疏(w含有更多的0).

目前趋近于使用L2范数.

#### 神经网络的正则化



称||𝑊[𝑙]||2为范数平方，这个矩阵范数||𝑊[𝑙]||2（即平方范数），被定义

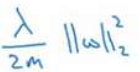
为矩阵中所有元素的平方求和。

第一个求和符号其值𝑖从1 到𝑛[𝑙−1]，第二个其𝐽值从1 到

𝑛[𝑙]，因为𝑊是一个𝑛[𝑙] × 𝑛[𝑙−1]的多维矩阵，𝑛[𝑙]表示𝑙 层单元的数量，𝑛[𝑙−1]表示第𝑙 − 1层隐藏单元的数量。

它表示的是一个矩阵中所有元素的平方和.

#### 正则化如何实现梯度下降?

左上紫色式子是J对w求导后的.  .

左下是参数更新. 带入后得:

相当于我们给矩阵W 乘以(1 – 𝑎𝜆/𝑚)倍的权重，矩阵𝑊减去𝛼𝜆/𝑚倍的它本身. 该系数小于1，因此𝐿2范数正则化也被称为“权重衰减”

#### 正则化为何能预防过拟合?

直观来讲.引入正则化后,w每次等于上次的(1- 𝑎𝜆/𝑚)倍加上一个更新量.

当lamda足够大的时候,这个系数会为0.也就是说神经元的权重会是0, 这样就把网络简化了,如左上图, 更类似一个逻辑回归的网络. 更不容易过拟合.



另一种思路:

从激活函数上讲, 当w每次缩小(1- 𝑎𝜆/𝑚)倍时, 激活函数的z轴会更加靠近中心部分.这是激活函数可以近似看成线性函数.而线性程度增强会抑制住”高方差”的过拟合效果.



#### Dropout正则化(随机失活)

随机去掉一些节点.

1. 每个batch时候,会随机一个概率值, 满足dropout的值的那个节点就会被消除掉.
2. 每次batch时候,消除的并不一样.(随机分布的).
3. Test时候,dropout会被bypass.



常见的dropout是inverted dropout(反向随机失活).

具体如何做inverted dropout?

引入一个三层网络, d3是dropout的概率,它是a3一个维度的概率矩阵, a3需要和d3相乘.约有20%的元素被清零.



然后对a3如下操作:

𝑎3/= 𝑘𝑒𝑒𝑝 − 𝑝𝑟𝑜𝑏

其目的是保持a3输出到z4的值相同(z4期望不变).具体讲, 因

𝑧[4] = 𝑤[4]𝑎[3] + 𝑏[4]，

期望a3减小20%, 但是z4的期望不变,就只能:

𝑤[4]𝑎[3]/0.8

扩大值和z4期望一致.

