21.9.25小红书面试×

一、 自我介绍

二、 研究方向介绍

三、 实习与技能

1．项目与实习中挑一个详细说一下

2．逻辑回归与线性回归的联系与区别

联系：

- 逻辑回归可以理解为在线性回归后加了一个sigmoid函数。将线性回归变成一个0~1输出的分类问题。

区别：

- 线性回归用来预测连续的变量（房价预测），逻辑回归用来预测离散的变量（分类，癌症预测）
- 线性回归是拟合函数，逻辑回归是预测函数
- 线性回归的参数计算方法是最小二乘法，逻辑回归的参数计算方法是似然估计的方法

如果用最小二乘法，目标函数就是 $E_{w,b}=\sum_{i=1}^{m}\left(y_i-\frac{1}{1+e^{-(w^Tx_i+b)}}\right)^2$,是非凸的，不容易求解，会得到局部最优。

如果用最大似然估计，目标函数就是对数似然函数：

$l_{w,b}=\sum_{i=1}^{m}\left(-y_i\left(w^Tx_i+b\right)+ln\left(1+e^{w^Tx_i+b}\right)\right)$,是关于 (w,b) 的高阶连续可导凸函数，可以方便通过一些凸优化算法求解，比如梯度下降法、牛顿法等。

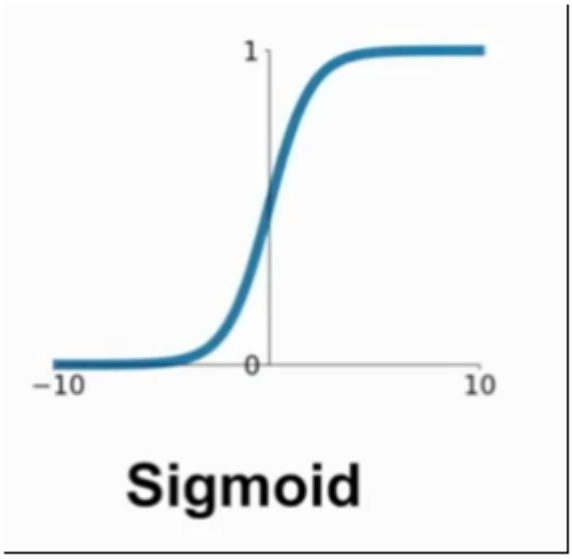
3．sigmoid作为激活函数的优缺点是什么

a．sigmoid函数

- 特点：所有元素都被压缩在[0,1]范围内，当输入数字很大或很小时，图

像都趋于平滑，在0附近趋于线性；其函数表达式和图像如下所示

$$\sigma(x) = 1/(1+e^{-x})$$



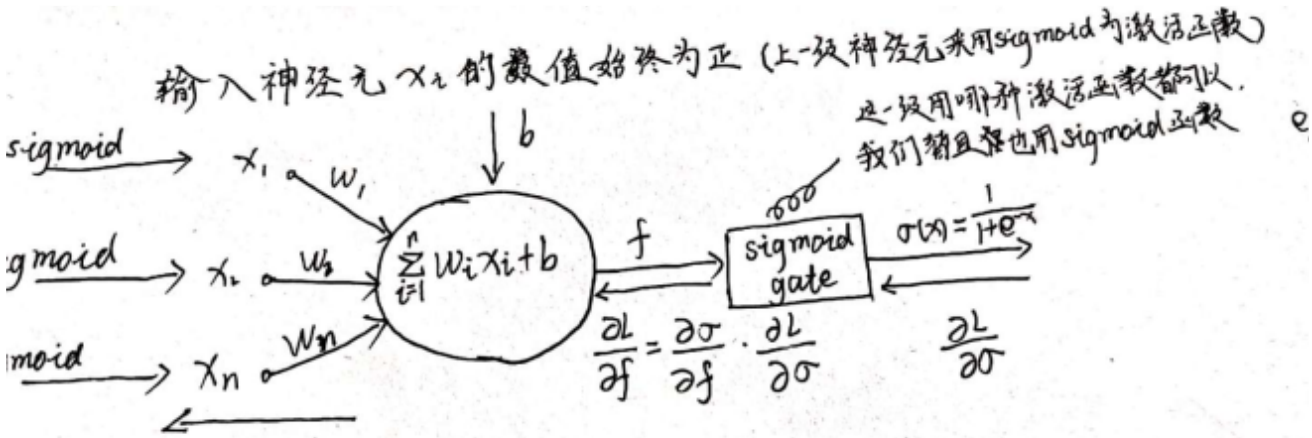
- 缺点：

  ⅰ．sigmoid函数饱和会导致梯度消失

  当x是很小的负数或很大的正数时，它们都处于sigmoid函数的平滑区域，这些区域的梯度就会消失，从而无法得到梯度流的反馈。（关于其梯度的计算，可以参照下面的推导，也可以之间看图的斜率）

  ⅱ．sigmoid是一个非零中心的函数

  下面我通过一个例子让大家看得更清楚一点。由sigmoid函数特点可知，假设上一级神经元采用了sigmoid激活函数，那么它输出的就全是[0,1]之间的数，即输入到这一级神经元的数x均大于0，下图的推导给大家展示了整个过程，可以看出它在走Z 字形逼近最优解，收敛速度很慢；

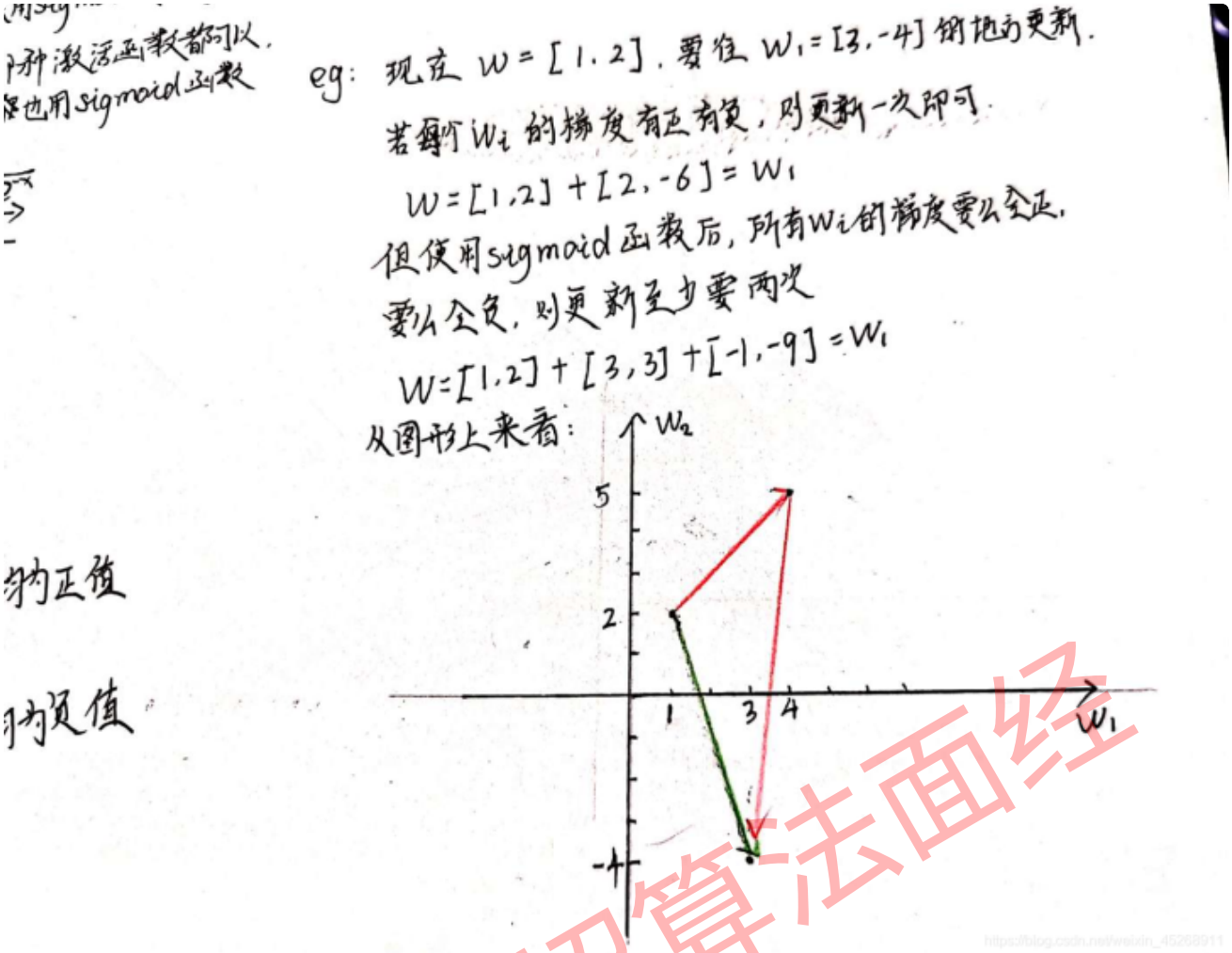


$$\frac{\partial L}{\partial w_i} = \frac{\partial f}{\partial w_i} \cdot \frac{\partial \sigma}{\partial f} \cdot \frac{\partial L}{\partial \sigma}$$

$$= x_i \cdot \frac{\partial \sigma}{\partial f} \cdot \frac{\partial L}{\partial \sigma}$$

$x_i > 0$ 时：

- $\frac{\partial \sigma}{\partial f} \cdot \frac{\partial L}{\partial \sigma} > 0 \Rightarrow \frac{\partial L}{\partial w_i} > 0 \Rightarrow$ 所有关于 w 的梯度均为正值，要增大，都增大
- $\frac{\partial \sigma}{\partial f} \cdot \frac{\partial L}{\partial \sigma} < 0 \Rightarrow \frac{\partial L}{\partial w_i} < 0 \Rightarrow$ 所有关于 w 的梯度均为负值，要减小，都减小



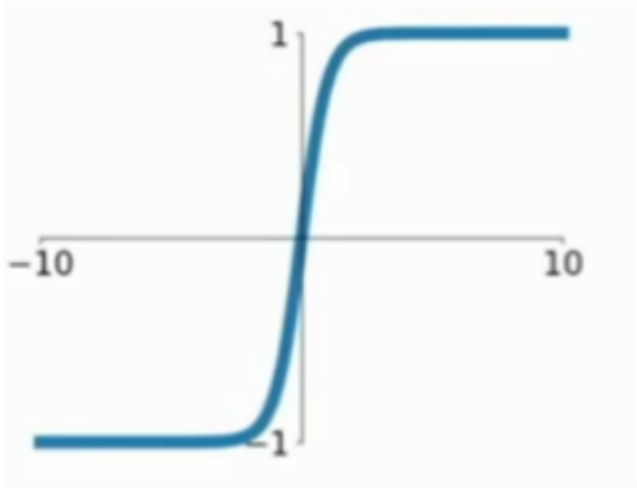
由此我们知道，如果使用sigmoid函数作为激活函数，所有W的梯度要么全正，要么全负，但如果换一个以0为中心的激活函数，W的梯度就可以同时有正有负，直接逼近最优解；所以我们一般不用sigmoid函数作为激活函数。

iii. 指数函数的计算代价有点高

尽管这在这个复杂神经网络框架中不值一提，但仍然是我们需要注意的一小点

b. tanh函数

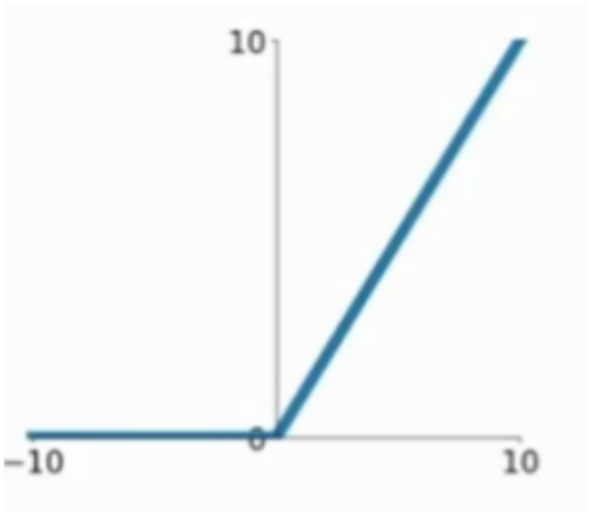
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



- 特点：它看起来和sigmoid函数非常相似，但不同之处在于，它将输出值变换到了[-1,1]的范围内，这就解决了sigmoid函数以非0为中心的问题，但它仍然有梯度消失的问题；

c. ReLU函数

$$f(x) = \max(0,x)$$



- 特点：这就是我们上一章讲到的卷积神经网络中常用的激活函数，与前

两个激活函数相比，ReLU函数不会在正的区域产生饱和现象，也就是在正的区域不会有梯度消失，这是一个很大的优势；它的计算成本也不高，且收敛速度大概是前两种的6倍；

- 缺点：

  ⅰ．但是我们注意到，它也是以非0为中心的，所以tanh刚刚解决的问题现在又出现了

  ⅱ．此外，ReLU函数会产生Dead ReLU Problem（这就是我困惑的第二个点了，为什么会说梯度过大，或者学习率过大就会出现神经元死亡？）

  首先，我们来解释一下什么叫Dead ReLU Problem，它指的是某些神经元可能永远不会被激活，导致相应的参数永远不能被更新训练神经网络的时候；举个例子，一旦学习率没有设置好，第一次更新权重的时候，输入是负值，那么这个含有ReLU的神经节点就会死亡，再也不会被激活。因为：ReLU的导数在x>0的时候是1，在x<=0的时候是0。如果x<=0，那么ReLU的输出是0，那么反向传播中梯度也是0，权重就不会被更新，导致神经元不再学习。

  https://www.zhihu.com/question/67151971?sort=created
  <https://www.zhihu.com/question/67151971?sort=created>

- 优点

  ⅰ．采用Relu激活函数，整个过程的计算量节省很多。

  ⅱ．对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失的情况（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失，从而无法完成深层网络的训练。

  ⅲ．ReLU会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生

- 附加：relu是非线性激活函数还是线性激活函数？

  relu是非线性激活函数

  题主的疑问在于，为什么relu这种“看似线性”（分段线性）的激活函数所形成的网络，居然能够增加非线性的表达能力。

  1、首先什么是线性的网络，如果把线性网络看成一个大的矩阵M。那么输入样本A和B，则会经过同样的线性变换MA，MB（这里A和B经历

的线性变换矩阵M是一样的）。

2、的确对于单一的样本A，经过由relu激活函数所构成神经网络，其过程确实可以等价是经过了一个线性变换M1，但是对于样本B，在经过同样的网络时，由于每个神经元是否激活（0或者Wx+b）与样本A经过时情形不同了（不同样本），因此B所经历的线性变换M2并不等于M1。因此，relu构成的神经网络虽然对每个样本都是线性变换，但是不同样本之间经历的线性变换M并不一样，所以整个样本空间在经过relu构成的网络时其实是经历了非线性变换的。

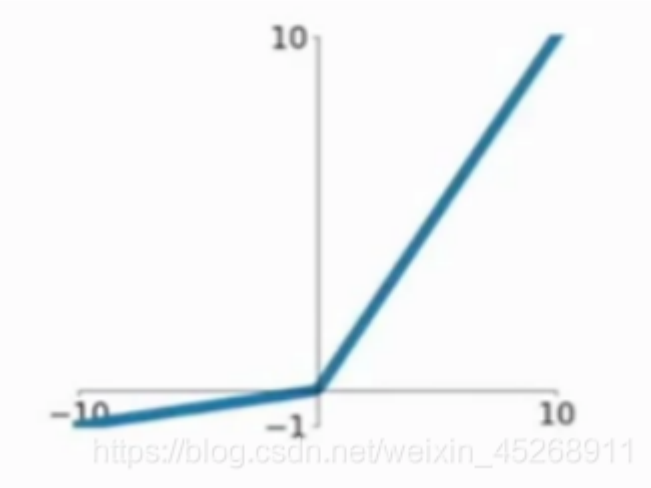
3、还有一种解释就是，不同样本的同一个feature，在通过relu构成的神经网络时，流经的路径不一样（relu激活值为0，则堵塞；激活值为本身，则通过），因此最终的输出空间其实是输入空间的非线性变换得来的。

4、更极端的，不管是tanh还是sigmoid，你都可以把它们近似看成是分段线性的函数（很多段），但依然能够有非线性表达能力；relu虽然只有两段，但同样也是非线性激活函数，道理与之是一样的。

5、relu的优势在于运算简单，网络学习速度快

d. Leaky ReLU函数

$$f(x) = \max(0.01x, x)$$



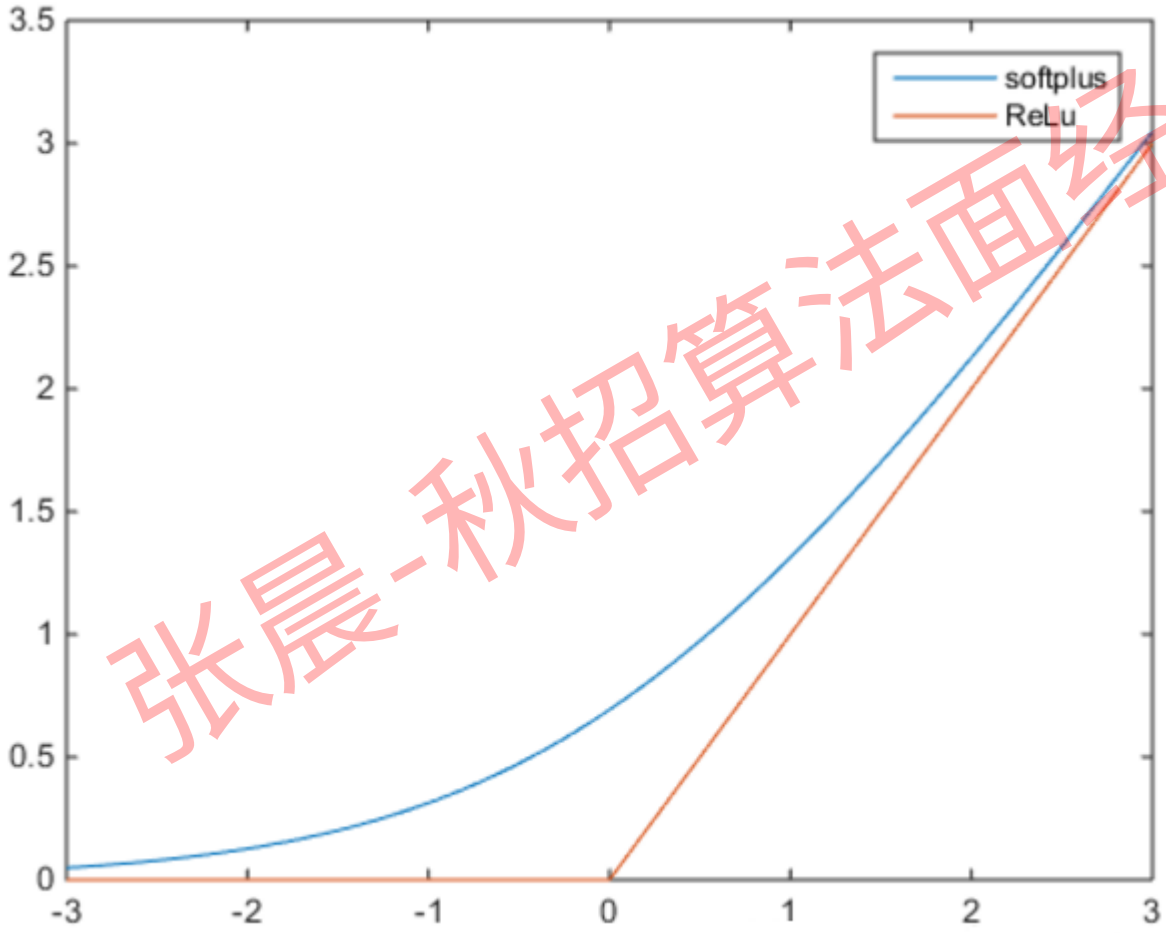
- 特点：在上文中我们提到，ReLU将所有的负值设置为0，造成神经元节点死亡情况。因此Leaky ReLU对其做了一个改进，给所有负值赋予一个

非零的斜率，这样，不管输入是正是负，都不会有饱和的现象。

4．还有哪些激活函数

5．针对rulu的断点缺陷有哪种改进

relu在0点的确不可导，正如@huluobo说的，可以用log(1+ex)来近似，这个函数是连续的，它在0点的导数是0.5。也就是相当于relu在0点的导数取为0.5，也正好是0和1的均值。tensorflow里面就默认relu在0点的导数是0。softplus可以看作是ReLu的平滑。根据神经科学家的相关研究，softplus和ReLu与脑神经元激活频率函数有神似的地方。也就是说，相比于早期的激活函数，softplus和ReLu更加接近脑神经元的激活模型。



6．梯度消失与梯度爆炸的原因与解决方法

- **梯度消失的原因：**

ⅰ．网络层数过于深：反向传播不断更新参数的过程是一种连乘机制，随着网络层数的加深，连乘的值越来越多，当很多小于1的值相乘时，会导致乘积的最终结果（梯度）非常小，最终导致输入侧的梯度更新慢或者无法更新。

ⅱ．使用了不恰当的激活函数：如sigmoid函数，该函数可将任意值映

射到（0, 1）范围之内，求导的结果是f’(x)=f(x)(1−f(x))，导数在（0, 0.25）之间。在反向传播过程中，一定会对外层sigmoid函数求导，很多接近0的数值相乘会导致梯度消失。

- **梯度爆炸原因：**
  - ⅰ．网络层数过于深：反向传播不断更新参数的过程是一种连乘机制，随着网络层数的加深，连乘的值越来越多，当很多大于1的值相乘时，会导致乘积的最终结果（梯度）非常大，最终导致输入侧的梯度过于大，以至于溢出，导致 NaN 值出现。
  - ⅱ．权重初始化值过于大：将w初始化为一个较大的值时，例如>10的值，那么从输出层到输入层每一层都会有一个s‘(zn)*wn的增倍，当s‘(zn)为0.25时s‘(zn)*wn>2.5，同梯度消失类似，当神经网络很深时，梯度呈指数级增长，最后到输入侧时，梯度将会非常大，最终会得到一个非常大的权重更新值。
- 解**决方法：**
  - ⅰ．l1、l2正则化：如果发生梯度爆炸，那么权值就会变的非常大，通过正则化项来约束权重的大小，可以在一定程度上降低梯度爆炸的发生。
  - ⅱ．梯度剪切：梯度剪切这个方案主要是针对梯度爆炸提出的，其思想是设置一个梯度剪切阈值，更新梯度的时候，如果梯度超过这个阈值，那么就将其强制限制在这个范围之内。
  - ⅲ．选择合适的激活函数：如：relu函数的导数在正数部分是恒等于1的，因此在深层网络中使用relu激活函数就不会导致梯度消失和爆炸的问题。
  - ⅳ．batch normalization：通过对每一层的输出规范为均值和方差一致的，消除了权重参数放大缩小带来的影响，进而解决梯度消失和爆炸的问题，或者可以理解为BN将输出从饱和区拉倒了非饱和区。
  - ⅴ．残差网络：相比较于以前直来直去的网络结构，残差中有很多跨层连接结构，这样的结构在反向传播中具有很大的好处，可以避免梯度消失。
  - ⅵ．LSTM的“门（gate）”结构：LSTM的结构设计可以改善RNN中的梯度消失的问题。主要原因在于LSTM内部的门结构，通过改善一条

路径上（Cell State）的梯度问题拯救了总体的远距离梯度。

vii. pre-training+fine-tunning：基本思想是每次训练一层隐节点，训练时将上一层隐节点的输出作为输入，而本层隐节点的输出作为下一层隐节点的输入，此过程就是逐层“预训练”（pre-training）；在预训练完成后，再对整个网络进行“微调”（fine-tunning）。此思想相当于是先寻找局部最优，然后整合起来寻找全局最优。

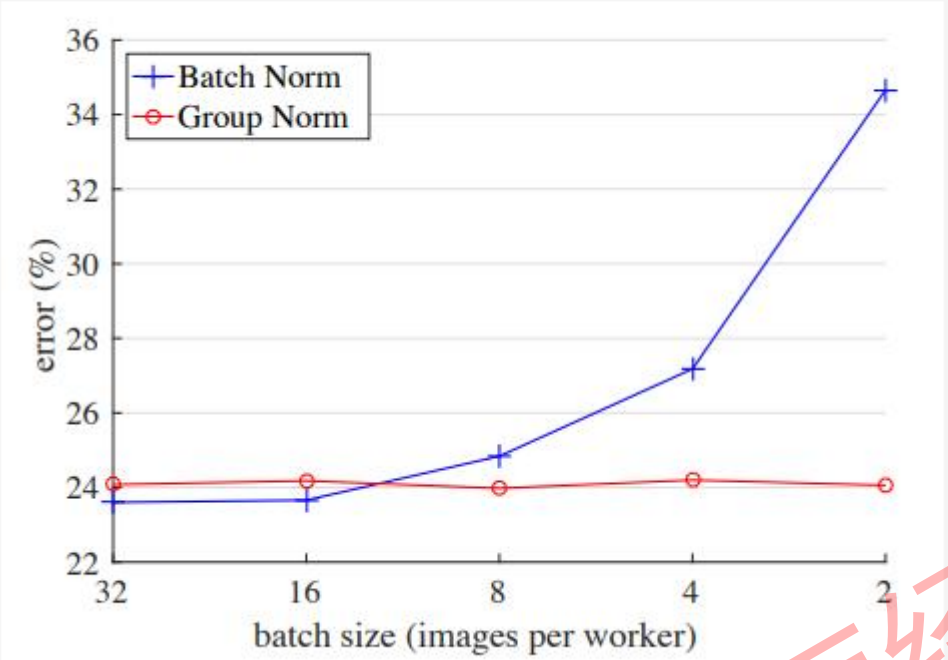
7．normolization有哪几种类型，区别是什么?

- Batch Normalization，其论文：https://arxiv.org/pdf/1502.03167.pdf <https://arxiv.org/pdf/1502.03167.pdf>
- Layer Normalizaiton，其论文：https://arxiv.org/pdf/1607.06450v1.pdf <https://arxiv.org/pdf/1607.06450v1.pdf>
- Instance Normalization，其论文：https://arxiv.org/pdf/1607.08022.pdf <https://arxiv.org/pdf/1607.08022.pdf>
- Group Normalization，其论文：https://arxiv.org/pdf/1803.08494.pdf <https://arxiv.org/pdf/1803.08494.pdf>
- Switchable Normalization，其论文：https://arxiv.org/pdf/1806.10779.pdf <https://arxiv.org/pdf/1806.10779.pdf>

  BN的不足根源在于测试时使用的两个在训练阶段时维护的参数,均值μ和修正的方差σ:

  - 当训练集和测试集的数据分布不一致的时候,训练集和测试集的均值和方差存在较大差异,最终影响模型预测精度;
  - 即使训练集和测试集的数据分布相对一致,但当batch-size较小时,计算出来的均值和方差所具有的统计意义不强,同样会在预测的时候影响模型准确度.
  - 一般BN合适的batch-size是32,但对于图像分割,目标检测这些任务对于显存要求大,机器无法满足大batch-size的要求,往往只能设置为1-2.而随

着batch-size不断变小,误差越来越大.如下图.



独立于batch进行归一化的方法有 Layer Normalization, Instance Normalization 和 Group Normalization.对比BN,结合一下几张图说明每一种Normalization的归一化方法.

考虑一个Batch=3, channels number=6, H, W 的tensor.绿色表示归一化的范围.

- BN:在batch,H,W的维度进行归一化,即同一个Channel的feature maps进行归一化.共操作了6(channel number)次归一化;
- LN:在Channel, H, W的维度进行归一化,即对mini-batch内的每一个样本进行归一化.共操作了3(Batch-size)次归一化;
- IN:在H,W维度进行归一化,即对每一个features map进行归一化,共操作了6*3(channels number * batch-size)次归一化
- GN:在一个样本的多个channels内分组,分成多组channels,在各组内进行归一化.图中将channels分成了两组(G=2), 因此归一化操作次数为:2*3(G*Batch-size).
- SN:将 BN、LN、IN 结合，赋予权重，让网络自己去学习归一化层应该使用什么方法

8．正则化方法有哪几种，L1与L2的区别是什么，为啥会有这些区别

在训练数据不够多时，或者overtraining时，常常会导致过拟合（overfitting）。正则化方法即为在此时向原始模型引入额外信息，以便防止过拟合和提高模型泛化性能的一类方法的统称。

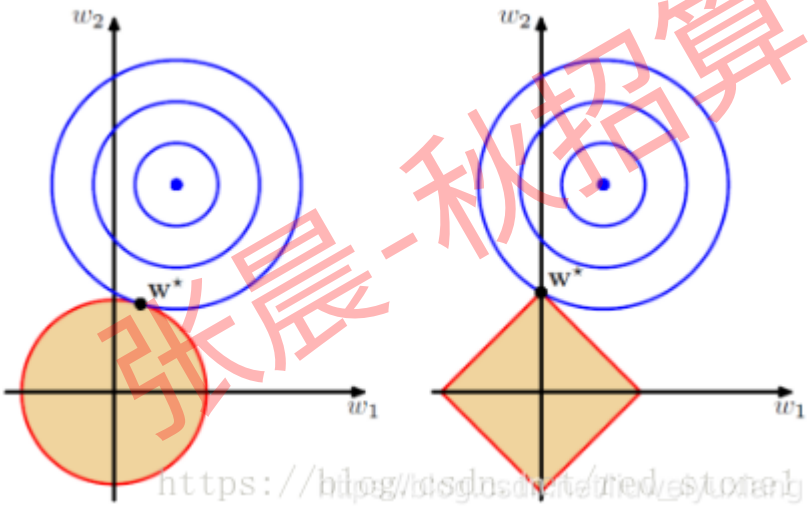
· 方法一：L1和L2正则化：

$$cost function = Loss + \lambda \sum ||w||$$

在这个公式中，我们直接用权值的绝对值来对模型进行惩罚，使得模型不要太拟合于训练集。和L2不同的是，此时的权值很可能最终会是0。当我们想要压缩模型的时候，采用L1正则化非常的合适。否则，其他情况我们通常更多的使用L2.

$$cost function = Loss + \lambda \sum ||w||^2$$

这里的 λ \lambdaλ 是一个超参数，需要人为的指定，通过控制 λ \lambdaλ 的大小，可以控制前面的损失项和正则化项所占的比例。通过加入这个L2正则化项，会使得模型中的参数的数值趋近于0.（但不是0）

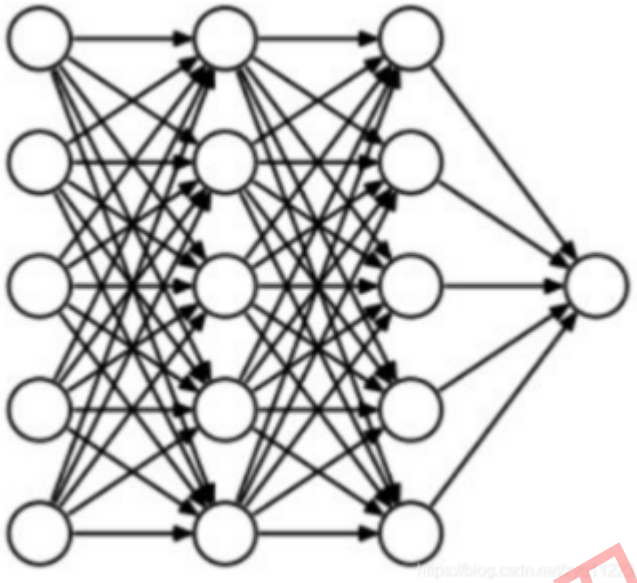


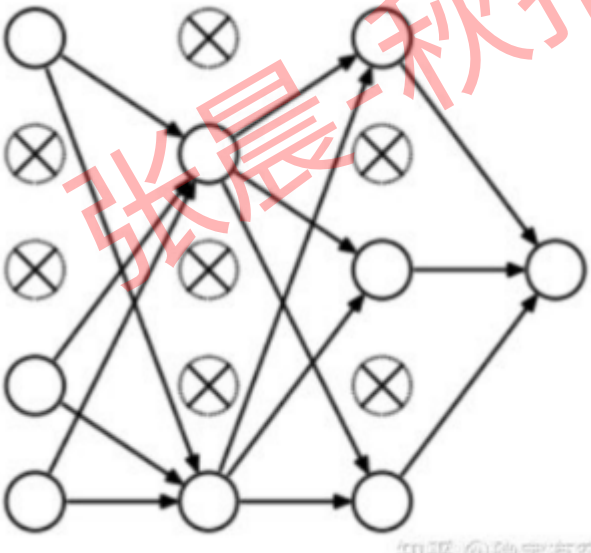
上面的图，左面是L2约束下解空间的图像，右面是L1约束下解空间的图像。蓝色的圆圈表示损失函数的等值线。同一个圆上的损失函数值相等的，圆的半径越大表示损失值越大，由外到内，损失函数值越来越小，中间最小。如果没有L1和L2正则化约束的话，w1和w2是可以任意取值的，损失函数可以优化到中心的最小值的，此时中心对应的w1和w2的取值就是模型最终求得的参数。但是填了L1和L2正则化约束就把解空间约束在了黄色的平面内。黄色图像的边缘与损失函数等值线的交点，便是满足约束条件的损失函数最小化的模型的参数的解。 由于L1正则化约束的解空间是一个菱形，所以等值线与菱形端

点相交的概率比与线的中间相交的概率要大很多，端点在坐标轴上，一些参数的取值便为0。L2正则化约束的解空间是圆形，所以等值线与圆的任何部分相交的概率都是一样的，所以也就不会产生稀疏的参数。

- 方法二：Dropout

dropout，说白了就是让一些参数失效。具体我们举例说明。假设一个神经网络模型结构如下图所示：





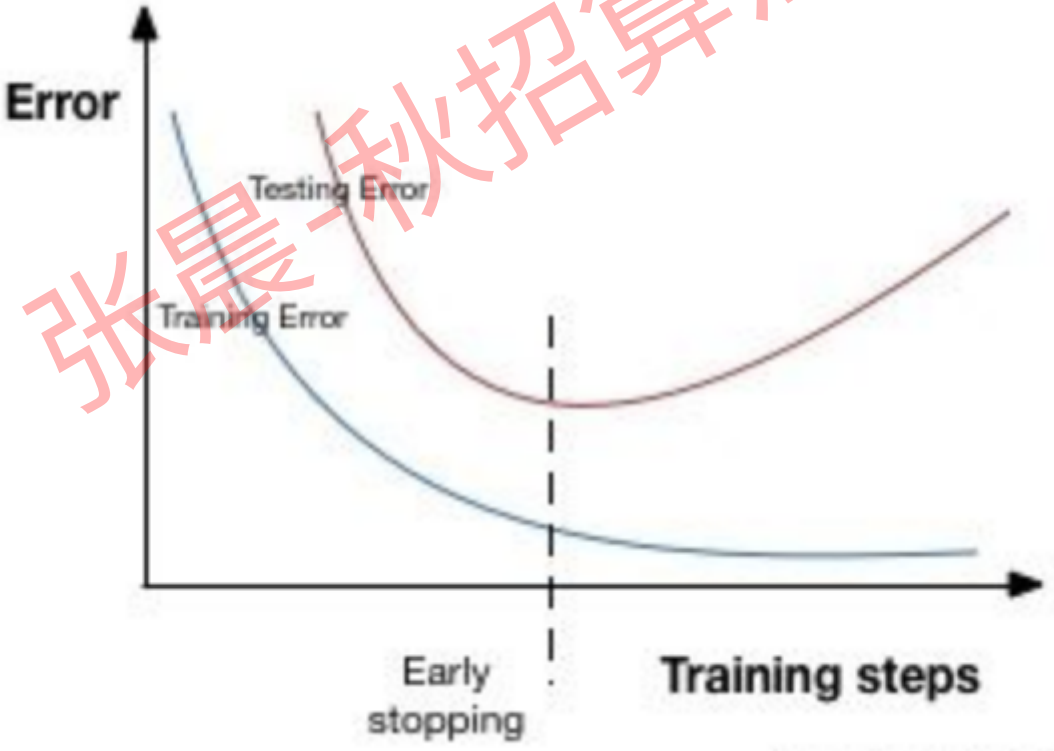
在训练的每次循环中，我们随机的将一些参数进行隐藏，说白了就是将这些参数的值置为0。此时的网络，可能就如下图所示。此时的网络，相比于最初的网络，就显得没有那么复杂了，从而可以减少过拟合的发生。dropout的细节是，在训练网络的过程中（检测阶段是没有dropout这个过程中，检测阶段所有神经元都参与计算），每一次的训练，都随机的选取一部分点，将这些

点的值置为0。由于在每次的训练过程中，被隐藏的点都不一样，所以，我们也可以看成dropout使得该网络从一个网络变成了多个网络的累加。因为每一次训练，都隐藏了不同的权值，相当于都是一个新的网络。通过多次训练，相当于将不同的网络进行了叠加。一般来讲，叠加而成的组合网络，一般优于单一网络，因为组合网络能够捕捉到更多的随机因素。同样的，采用了dropout以后，网络的性能一般也比没有使用dropout的网络要好。

- 方法三：data augmentation（数据增强）

之所以会发生过拟合，是因为模型的参数太多，可以用于训练的数据太少，所以除了更改模型的一些性质之外，也可以尝试增加训练数据的数量。假设我的训练数据中，只有一张猫的图片，如上图左侧所示。但是，我们可以通过翻转，裁切，旋转，平移甚至高斯模糊等各种手段，用这一张图片生成很多张猫的图片，如上图右边所示。这些生成的图片，和原图片同属于一个类别，通过这种方法，可以大大的提高数据量，从而也可以防止过拟合。

- 方法四：early stopping



在训练的过程中，可以将训练集的一部分分出来，作为验证集，然后在训练过程中，同时观察error在训集上和验证集上的大小，如上图所示。当发现训练集上error在持续减小，但是验证集上的error不再减小反而增大的时候，

就表示该模型可能对训练集发生了过拟合，这个时候，就可以停止训练了。在验证集error最小的时候停止训练，从而可以得到一个最好的效果。

9．SVM用过吗，有哪几种类型，推导过程用到了什么方法

a．拉格朗日

b．对偶问题

c．求导，令导数为0,计算参数值

d．带入，求另外的参数，使用SMO（序列最小优化）

10．有哪几种核函数

- 线性
- 多项式
- 高斯
- 傅里叶
- sigmoid

四、算法

1．逆波兰表达式求值：https://leetcode-cn.com/problems/evaluate-reverse-polish-notation/ <https://leetcode-cn.com/problems/evaluate-reverse-polish-notation/>