**问题0   
写出常见的正则化和防止过拟合的技术，平时调参的经验**

**问题1****分析梯度弥散和梯度爆炸的原因，及其解决手段**

梯度消失：许多激活函数将输出值挤压在很小的区间内，在激活函数两端较大范围的定义域内梯度为0，导致权重更新的缓慢训练难度增加，造成学习停止。（前面层上的梯度是来自后面的层上项的乘积，当层数过多时，随着乘积的累积，将越来越小。）

梯度爆炸：前面层比后面层变化更快时，引起梯度爆炸。

解决手段：减少层数；增大学习率；用Relu代替sigmoid。

梯度爆炸的预防方案：L1或者L2正则化；

梯度爆炸的解决方案：梯度截断；

梯度消失的预防方案：LSTM结构；参数初始化；选用ReLU激活函数

**问题2   
权重初始化（Weight Initialization）**

1）小的随机数

w= 0.01 \* np.random.randn(fan\_in,fan\_out)

2）神经元将饱和，梯度为0

w = 1.0 \* np.random.randn(fan\_in,fan\_out)

3）合理的初始化（Xavier init）

w = np.random.randn((fan\_in,fan\_out)/np.sqrt(fan\_in)

**dropout为什么解决overfitting**

**L1和L2 regularization原理**

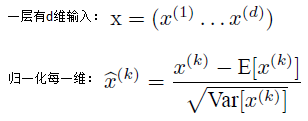
**为什么L1 regularization可以使参数优化到0**

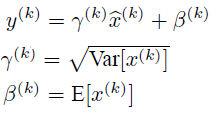
**batch normalizatin为什么可以防止梯度消失爆炸**

为了便于训练，我们经常归一化参数的初始值，通过mean=0, variance=1的高斯分布来初始化参数。在训练过程中，我们不同程度地更新参数，使用参数失去了归一化，这将降低训练速度且放大变化，网络越深问题越严重。

BN的本质原理：在网络的每一层输入的时候，又插入了一个归一化层，也就是先做一个归一化处理（归一化至：均值0、方差为1），然后再进入网络的下一层。BN是一个可学习、有参数（γ、β）的网络层。

归一化公式：



如果是仅仅使用上面的归一化公式，对网络某一层A的输出数据做归一化，然后送入网络下一层B，这样是会影响到本层网络A所学习到的特征的。比如我网络中间某一层学习到特征数据本身就分布在S型激活函数的两侧，你强制把它给我归一化处理、标准差也限制在了1，把数据变换成分布于s函数的中间部分，这样就相当于我这一层网络所学习到的特征分布被你搞坏了，这可怎么办？于是文献使出了一招惊天地泣鬼神的招式：变换重构，引入了可学习参数γ、β，这就是算法关键之处： 

上面的公式表明，通过学习到的重构参数γ、β，是可以恢复出原始的某一层所学到的特征的。

在使用BN前，减小学习率、小心的权重初始化的目的是：使其输出的数据分布不要发生太大的变化。

**batch normalizatin在CNN中卷积层怎么做**

BN层是对于每个神经元做归一化处理，甚至只需要对某一个神经元进行归一化，而不是对一整层网络的神经元进行归一化。既然BN是对单个神经元的运算，那么在CNN中卷积层上要怎么搞？假如某一层卷积层有6个特征图，每个特征图的大小是100\*100，这样就相当于这一层网络有6\*100\*100个神经元，如果采用BN，就会有6\*100\*100个参数γ、β，这样岂不是太恐怖了。因此卷积层上的BN使用，其实也是使用了类似权值共享的策略，**把一整张特征图当做一个神经元进行处理**。

卷积神经网络经过卷积后得到的是一系列的特征图，如果min-batch sizes为m，那么网络某一层输入数据可以表示为四维矩阵(m,f,w,h)，m为min-batch sizes，f为特征图个数，w、h分别为特征图的宽高。在CNN中我们可以把每个特征图看成是一个特征处理（一个神经元），因此在使用Batch Normalization，mini-batch size 的大小就是：m\*w\*h，于是对于每个特征图都只有一对可学习参数：γ、β。说白了吧，这就是相当于求取所有样本所对应的一个特征图的所有神经元的平均值、方差，然后对这个特征图神经元做归一化。

**batch normalizatin的作用**

1. 改善流经网络的梯度

2）允许更大的学习率，大幅提高训练速度：

你可以选择比较大的初始学习率，让你的训练速度飙涨。以前还需要慢慢调整学习率，甚至在网络训练到一半的时候，还需要想着学习率进一步调小的比例选择多少比较合适，现在我们可以采用初始很大的学习率，然后学习率的衰减速度也很大，因为这个算法收敛很快。当然这个算法即使你选择了较小的学习率，也比以前的收敛速度快，因为它具有快速训练收敛的特性；

3）减少对初始化的强烈依赖

4）改善正则化策略：作为正则化的一种形式，轻微减少了对dropout的需求

你再也不用去理会过拟合中dropout、L2正则项参数的选择问题，采用BN算法后，你可以移除这两项了参数，或者可以选择更小的L2正则约束参数了，因为BN具有提高网络泛化能力的特性；

5）再也不需要使用使用局部响应归一化层了（局部响应归一化是Alexnet网络用到的方法，搞视觉的估计比较熟悉），因为BN本身就是一个归一化网络层；

6）可以把训练数据彻底打乱（防止每批训练的时候，某一个样本都经常被挑选到，文献说这个可以提高1%的精度）。

注：以上为学习过程，在测试时，均值和方差(mean/std)不基于小批量进行计算， 可取训练过程中的激活值的均值。

**问题3   
模型欠拟合的解决方法**

**问题4   
写出三个损失函数，并分析特点和使用场景**

**问题5   
VGGNet,GoogleNet，Resnet的特点和对比**  
**问题7   
[深度学习](http://lib.csdn.net/base/deeplearning" \o "深度学习知识库" \t "http://blog.csdn.net/victoriaw/article/details/_blank)是否是万能的，深度学习未来发展的方向，你的观点**

**问题8   
CNN做卷积的运算时间复杂度**

**问题9   
LSTM结构推导，为什么比RNN好？**

**问题10   
类别不平衡时怎么办？**

**CNN最成功的应用是在CV，那为什么NLP和Speech的很多问题也可以用CNN解出来？为什么AlphaGo里也用了CNN？这几个不相关的问题的相似性在哪里？CNN通过什么手段抓住了这个共性？**

**什么时候用local-conv？什么时候用全卷积（每一个点用同一个filter）？**

**为什么引入非线性激励函数？**

**常用的激励函数？**

问题11   
Random forest和GBT描述；

问题12   
random forest的**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/victoriaw/article/details/_blank)**描述+bias和variance的分解公式；

问题13   
用EM理论推导GMM

问题14   
SVM的dual problem推导

问题15   
HMM原理

问题16   
HMM和CRF的本质区别；

问题17   
频率学派和贝叶斯派的本质区别；

问题18   
常用的优化方法；

问题19   
矩阵行列式的物理意义（行列式就是矩阵对应的线性变换对空间的拉伸程度的度量，或者说物体经过变换前后的体积比）

问题20   
**[Python](http://lib.csdn.net/base/python" \o "Python知识库" \t "http://blog.csdn.net/victoriaw/article/details/_blank)**写k-means

问题21   
动态预测每个区域的用车需求量；

问题22   
怎么样识别文本垃圾信息