**Deep Learning**

**1. 深度学习的深度体现在哪里？**

深度学习的“深度”便指的是图模型的层数以及每一层的节点数量，相对于之前的神经网络而言，有了很大程度的提升。

**2. 什么是batch normalization？为什么它能加速收敛和防止过拟合？**

1. batch normalization本质上解决的是反向传播过程中的梯度问题。在batch normalization中，是通过将activation规范为均值和方差一致的手段使得原本会减小的activation的scale变大。可以说是一种更有效的local response normalization方法。

首先，batch normalization算法在每一次迭代中的每一层输入都进行了归一化，将输入数据的分布归一化为均值为0，方差为1的分布：

其中，表示输入数据的第k维，表示该为数据的平均值，表示标准差。

但是这种做法有一个致命的缺点，尽管这样把每层的数据分布都固定了，但是这种分布不一定是前面一层的要学习到的数据分布，这样强行归一化就会破坏掉刚刚学习到的特征，batch normalization算法的第二步就解决了这个缺点。

batch normalization算法在第二部中设置了两个可学习的变量和，然后用这两个可学习的变量去还原上一层应该学到的数据分布。

添加这种操作的目的就是还原出上一层需要学习的数据分布，这样batch normalization就把原来不固定的数据分布全部转换为固定的数据分布，而这种数据分布恰恰就是要学习到的分布，从而加速了网络的训练。

batch normalization算法在测试阶段：batch normalization首先提取每次迭代时的每个mini-batch的平均值和方差进行归一化，再通过两个可学习的变量恢复要学习的特征。实际上在测试的过程中，batch normalization算法的参数就已经固定好了，首先进行归一化时的平均值和方差分别为：

即平均值为所有mini-batch的平均值的平均值，而方差为每个batch的方差的无偏估计。

1. batch normalization本质上解决的是反向传播过程中的梯度问题。

详细点说，反向传播时经过该层的梯度是要乘以该层的参数的，即前向有：

那么反向传播时便有：

那么考虑从层传到层的情况，有：

上面这个便是问题所在。因为网络层很深，如果大多小于1，那么传到这里的时候梯度会变得很小比如；而如果又大多大于1，那么传到这里的时候又会有梯度爆炸问题，比如。BN所做的就是解决这个梯度传播的问题，因为BN作用抹去了的scale影响。具体有：

那么反向求导时便有了：

可以看到此时反向传播乘以的数不再和的尺度相关，也就是说尽管我们在更新过程中改变了的值，但是反向传播的梯度却不受影响。更进一步：

即尺度较大的将获得一个较小的梯度，在同等的学习速率下其获得的更新更少，这样使得整体的更新更加稳健起来。总结起来就是BN解决了反向传播过程中的梯度问题（梯度消失和爆炸），同时使得不同scale的整体更新步调更一致。

**3. 什么是梯度消失和梯度爆炸？**

1. 梯度消失：根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都小于1的话，那么即使这个结果是0.99，在经过足够多层传播之后，误差对输入层的偏导会趋于0。
2. 梯度爆炸：根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都大于1的话，在经过足够多层传播之后，误差对输入层的偏导会趋于无穷大。

**4. 什么造成梯度消失问题? 推导一下。**

根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都小于1的话，那么即使这个结果是0.99，在经过足够多层传播之后，误差对输入层的偏导会趋于0。

推导过程：

假设损失函数为

运用符合求导公式：

从Sigmoid函数的图像可知，当偏大或者偏小时，，而，，因此会出现梯度消失。

**5. 如何解决梯度消失和梯度爆炸?**

1. 梯度消失：根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都小于1的话，那么即使这个结果是0.99，在经过足够多层传播之后，误差对输入层的偏导会趋于0。解决：可以采用ReLU激活函数有效的解决梯度消失的情况。
2. 梯度爆炸：根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都大于1的话，在经过足够多层传播之后，误差对输入层的偏导会趋于无穷大。解决：可以通过激活函数来解决。

**6. 如何确定是否出现梯度爆炸？**

训练过程中出现梯度爆炸会伴随一些细微的信号，如：

1. 模型无法从训练数据中获得更新（如低损失）
2. 模型不稳定，导致更新过程中的损失出现显著变化
3. 训练过程中，模型损失变成NaN

以下是一些稍微明显一点的信号，有助于确认是否出现梯度爆炸问题。

1. 训练过程中模型梯度快速变大
2. 训练过程中模型权重变成NaN
3. 训练过程中，每个节点和层的误差梯度值持续超过1.0

**7. 如何修复梯度爆炸问题？**

1. 重新设计网络模型：

在深度神经网络中，梯度爆炸可以通过重新设计层数更少的网络来解决。

使用更小的批尺寸对网络训练也有好处。

在循环神经网络中，训练过程中在更少的先前时间步上进行更新（沿时间的截断反向传播）可以缓解梯度爆炸问题。

1. 使用 ReLU 激活函数：

在深度多层感知机神经网络中，梯度爆炸的发生可能是因为激活函数，如之前很流行的 Sigmoid 和 Tanh 函数。

使用 ReLU 激活函数可以减少梯度爆炸。采用 ReLU 激活函数是最适合隐藏层的新实践。

1. 使用长短期记忆网络：

在循环神经网络中，梯度爆炸的发生可能是因为某种网络的训练本身就存在不稳定性，如随时间的反向传播本质上将循环网络转换成深度多层感知机神经网络。

使用长短期记忆（LSTM）单元和相关的门类型神经元结构可以减少梯度爆炸问题，

采用 LSTM 单元是适合循环神经网络的序列预测的最新最好实践。

1. 使用梯度截断（Gradient Clipping）：

处理梯度爆炸有一个简单有效的解决方案：如果梯度超过阈值，就截断它们。

1. 使用权重正则化（Weight Regularization）：

如果梯度爆炸仍然存在，可以尝试另一种方法，即检查网络权重的大小，并惩罚产生较大权重值的损失函数。该过程被称为权重正则化，通常使用的是 L1 惩罚项（权重绝对值）或 L2 惩罚项（权重平方）。

**8. 深度学习的优化方法有哪些？**

常用的深度学习优化方法有SGD，Adam和Adagrad。

**9. SGD、Adam、Adagrad有什么区别？**

1. SGD随机选取一个样本点更新参数
2. Adam利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率
3. Adagrad设置全局学习率之后，每次通过全局学习率逐参数的除以历史梯度平方和的平方根，使得每个参数的学习率不同

**10. Adagrad的原理是什么？为什么Adagrad适合处理稀疏梯度？**

Adagrad公式推导：

表示第个参数的梯度，对于经典的SGD优化函数，梯度更新的公式如下：

Adagrad这样表示

代表每一次迭代；一般是一个极小值，作用是防止分母为0；表示前步参数梯度的累加：。

容易看出，随着算法不断迭代，会越来越大，整体的学习率会越来越小。所以一般来说Adagrad算法一开始时激励收敛，到了后面就慢慢变成惩罚收敛，速度越来越慢。

Adagrad算法能够在训练中自动的对学习率进行调整，对于出现频率较低参数采用较大的更新；相反，对于出现频率较高的参数采用较小的更新。因此，Adagrad非常适合处理稀疏数据。