**1、batch-GD， SGD， Mini-batch-GD， Stochastic GD， Online-GD的区别？**



深度学习的优化算法，说白了就是梯度下降。每次的参数更新有两种方式。

**第一种**，遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习，这称为Batch gradient descent，批梯度下降。

**另一种**，每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降，stochastic gradient descent。这个方法速度比较快，但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈。

为了克服两种方法的缺点，现在一般采用的是一种折中手段，**mini-batch gradient decent**，小批的梯度下降，这种方法把数据分为若干个批，按批来更新参数，这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大。

（这就是我们在tf或者其他机器学习模型中，经常要设置的batch参数）

**2.偏差、方差、噪声、泛化误差的联系与区别？**

**对学习算法除了通过实验估计其泛化性能（也就是算模型在测试数据上，模型输出与实际输出之间的误差，例如平方误差等等。。。）。我们还希望解释，为什么模型具有这样的性能，“偏差-方差分解”就是理解这个问题的工具。**

**该分解对期望泛化误差进行拆解：**

**泛化误差 = 偏差 + 方差 + 噪声**

其中，

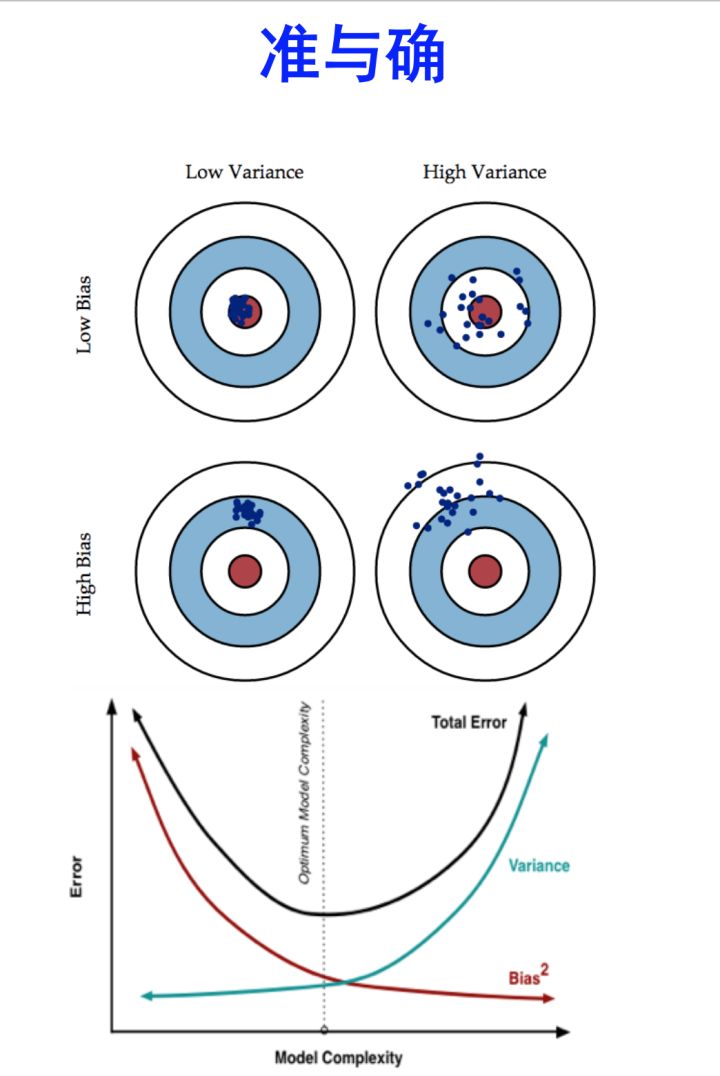
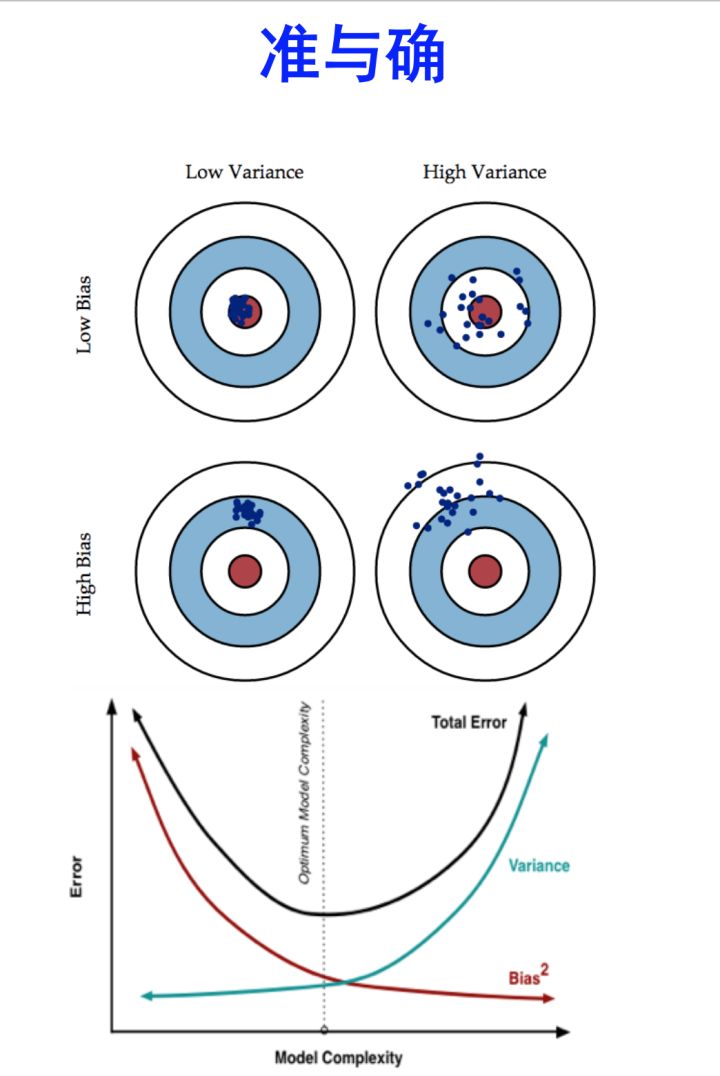
**偏差：**模型**期望输出与真实标记之间的差别**，即( - y)^2；反映的是**模型本身的拟合能力**；

**方差：**在某个数据集上，样本预测结果的集中程度，即模型输出与期望输出之间的差别，它反映的是**数据扰动对算法的影响、模型的稳定性**：

E{(f(x)–)^ 2}

**噪声**：噪声反映的是**标记与真实标记之间**的偏差，反映的是问题本身的复杂程度。

Error反映的是**整个模型的准确度**，**Bias**反映的是**模型在样本上的期望输出与真实值之间的误差**，即**模型本身的精准度**，**Variance反映的是模型每一次输出结果与模型输出期望之间的误差**，即**模型的稳定性**。



**3、训练中为什么要加入dropout？**

Dropout是指在模型训练时随机让网络某些隐含层节点的权重不工作，不工作的那些节点可以暂时认为不是网络结构的一部分，但是它的权重得保留下来，因为下次样本输入时它可能又得工作了。

**目的**：一种防止模型过拟合的手段。

**为什么**：

（1）**类似于样本加噪声**：加入dropout可以视作在样本中加入了噪声，能够提升模型的泛化能力

（2）**类似于bagging**：可以将dropout看作是模型平均的一种。对于每次输入到网络中的样本，其对应的网络结构都是不同的，但所有的这些不同的网络结构又同时share隐含节点的权值。这样不同的样本就对应不同的模型，是bagging的一种极端情况。

**4、为什么要用RELU？**

**（1）运算速度快**

因为RELU其分段线性性质，导致其前传，后传，求导都是分段线性。计算起来方便，而sigmoid等，涉及复杂的运算；

**（2）不容易梯度消失**

sigmoid和tanh的梯度在两端的饱和区域非常平缓，接近于0，很容易造成梯度消失的问题，减缓收敛速度。梯度消失在网络层数多的时候尤其明显，是加深网络结构的主要障碍之一

**（3）缓解过拟合**

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。

**5、什么是Batch Normalization？**

网络训练过程中**参数不断改变导致后续每一层输入的分布也发生变化**，**而学习的过程又要使每一层适应输入的分布，因此我们不得不降低学习率、小心地初始化**。作者将分布发生变化称之为 internal covariate shift。

Batch normalization的做法是，在每次进行随机梯度下降时**通过batch来对相应的激活结果做规范化操作，使得（输出信号各个维度）来对相应的激活结果做规范化操作，使得（输出信号各个维度）均值为 0，方差为1。**

**好处**： （1）解决sigmoid函数的梯度消失问题，但是效果不如RELU

（2）更容易设置学习率，因为每层的输入都变成了统一尺度，方便设置统一的学习率

（3）归一化后使得更多的权重分界面落在了数据中，降低了overfit的可能性，因此一些防止overfit但会降低速度的方法，例如dropout和权重衰减就可以不使用或者降低其权重。

**6、LeNet-5的结构？**



C1层是卷积层，单通道下用了**6个卷积核**，这样就得到了**6个feature map**，其中每个卷积核的大小为**5\*5**，用每个卷积核与原始的输入图像进行卷积，这样feature map的大小为(32-5+1)×(32-5+1)= **28×28**，所需要的参数的个数为(5×5+1)×6= 156(其中5×5为卷积模板参数，1为偏置参数)，连接数为(5×5+1)×28×28×6=122304(其中28×28为卷积后图像的大小)。

S2层为 pooling 层，也可以说是池化或者特征映射的过程，拥有**6个 feature map，每个feature map的大小为14\*14**，每个feature map的隐单元与上一层C1相对应的feature map的 2×2 单元相连接，这里没有重叠。计算过程是：2×2 单元里的值相加然后再乘以训练参数w，再加上一个偏置参数b(每一个feature map共享相同w和b)，然后取sigmoid （S函数：0-1区间）值，作为对应的该单元的值。（这个地方和之前自己想的不太一样）所以S2层中每 feature map 的长宽都是上一层C1的一半。S2层需要2×6=12个参数，连接数为(4+1)×14×14×6 = 5880。注：这里池化的过程与ufldl教程中略有不同。下面为卷积操作与池化的示意图：

C3层也是一个卷积层（多通道(14个通道)），注意此处**C3并不是与S2全连接而是部分连接**），**有16个卷积核，卷积模板的大小为5\*5，因此具有16个feature maps**，每个feature map的大小为(14-5+1)×(14-5+1）= 10×10。每个feature map只与上一层S2中部分feature maps相连接

S4层也是采样层，**有16个feature maps，每个feature map的大小为5×5**，计算过程和S2类似，需要参数个数为16×2 = 32个，连接数为(4+1)×5×5×16 = 2000.

C5为卷积层，**有120个卷积核，卷积核的大小仍然为5×5，因此有120个feature maps**， **Feature map的大小为1×1，这样刚好变成了全连接**，但是我们不把它写成F5，因为这只是巧合。C5层有120\*(5\*5\*16+1) = 48120(16为上一层所有的feature maps个数)参数，连接数也是这么多。

F6位全连接层，**有86个节点**。

**经典模型结构比较：**

