AI

Deep Learning

使用NVIDIA显卡的TensorFlow，需要准备一下工作：

1. 安装NVIDIA的Toolkit 8.0，链接见: <https://developer.nvidia.com/cuda-downloads>
2. 安装NVIDIA对应的补丁，见上述链接
3. 安装cuDNN v5.1，从对应的链接下载后(<https://developer.nvidia.com/cudnn>)，将文件进行解压，然后分别复制到C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0相应的文件夹下面即可，本次安装使用的是python3.6，在使用过程中发现cuDNN v6.0版本报DLL load failed的错误，使用cuDNN v5.1替换之后，TensorFlow可以正常运行。
4. 使用pip命令安装带有gpu版本的TensorFlow:

Pip install tensorflow -gpu

神经网络

神经网络是由一层一层构建的，那么每**层**究竟在做什么？

**数学式子**：\vec{y}= a(W\cdot\vec{x} + {b})，其中\vec{x}是输入向量，\vec{y}是输出向量，\vec{b}是偏移向量，W是权重矩阵，a()是激活函数。每一层仅仅是把输入\vec x经过如此简单的操作得到\vec y

**数学理解：通过如下5种对输入空间（输入向量的集合）的操作，完成 输入空间 ——> 输出空间 的变换 (矩阵的行空间到列空间)。   
注：用“空间”二字的原因是被分类的并不是单个事物，而是一类事物。空间是指这类事物所有个体的集合。**

**1. 升维/降维**

**2. 放大/缩小**

**3. 旋转**

**4. 平移**

**5. “弯曲”**

增加节点数：增加维度，即增加线性转换能力。

增加层数：增加激活函数的次数，即增加非线性转换次数。

梯度下降有两个主要问题：

1. **局部极小值**：解决该问题的主要方法为

* **调节步伐：调节学习速率，使每一次的更新“步伐”不同。常用方法有：**

1. 随机梯度下降（Stochastic Gradient Descent (SGD)：每次只更新一个样本所计算的梯度
2. 小批量梯度下降（Mini-batch gradient descent）：每次更新若干样本所计算的梯度的平均值
3. 动量（Momentum）：不仅仅考虑当前样本所计算的梯度；Nesterov动量（Nesterov Momentum）：Momentum的改进
4. Adagrad、RMSProp、Adadelta、Adam：这些方法都是训练过程中依照规则降低学习速率，部分也综合动量

* **优化**起点：合理初始化权重（weights initialization）、预训练网络（pre-train），使网络获得一个较好的“起始点”，如最右侧的起始点就比最左侧的起始点要好。常用方法有：高斯分布初始权重（Gaussian distribution）、均匀分布初始权重（Uniform distribution）、Glorot 初始权重、He初始权、稀疏矩阵初始权重（sparse matrix）

1. **提高梯度计算速度**

名词解释：

以分类为例，当要分类正数、负数、零，三类的时候，一维空间的直线可以找到两个超平面（超平面：**比当前空间低一维的子空间**。当前空间是直线的话，超平面就是点）

**处理结构化数据**：当需要将数据库中的结构化数据持久化或者进行网络传输时，就需要先将他们序列化，所谓序列化，是将结构化的数据变成数据流的格式，简单地说就是变成一个字符串，如何将结构化的数据序列化，并从序列化之后的数据流中还原出原来的结构化数据，统称为处理结构化数据，这就是Protocol Buffer解决的主要问题，除了Protocol Buffer之外，XML和JSPN是两种比较常用的结构化数据处理工具。

**经典损失函数：**

1. **对于分类问题：**交叉熵(信息论中的概念，原本是用来估算信息论中平均编码长度)是常用的评判方法之一，交叉熵刻画了两个概率分布之间的距离，他是分类问题中比较常用的一种损失函数。Softmax回归是一种常见的将神经网络的前向传播得到结果的方法，TensorFlow中已经完整封装了交叉熵的计算过程，可以使用softmax\_cross\_entropy\_with\_logits直接进行交叉熵的计算。
2. **对于回归问题：**回归问题区别于分类问题，分类问题会有多个输出，分类问题只会有一个输出，这个输出就是预测值，对于回归问题，最常用的损失函数 是均方误差(MSE mean squared error)
3. **自定义损失函数:**由于在现实使用场景中，预测多于预测少之间得到的投入产出比并不对等，因此需要自定义损失函数，使得最终的结果符合预期(利润最大化等等)

**神经网络优化算法**

神经网络的优化过程可以分为两个阶段，第一个阶段先通过前向传播算法计算得到预测值，并将预测值和真实值之间对比得到两者之间的差距，然后在第二个阶段通过反向传播算法计算损失函数对每一个参数的梯度，再根据梯度和学习率使用梯度下降算法更新每一个参数。

由于梯度下降法存在一个局部最优的问题，所以在训练神经网络时，参数的数值化值很大程度影响到最后得到的结果，只有损失函数为凸函数时，梯度下降算法才能保证达到全局最优。

梯度下降的另外一个问题就是计算时间太长，因为要在全部训练数据上最小化损失，所以损失函数在所有的训练数据上需要计算损失和，这在海量数据情况下，计算损失函数非常消耗时间，为了加速可以使用**随机梯度下降的算法**，这个算法不是计算全部的损失函数，而是在每一轮的迭代中，随机优化某一条数据上的损失函数，但是随机梯度下降法在某一条数据上损失函数更小并不代表全部数据在损失函数上更小，所以随机梯度下降算法甚至不能达到局部最优。综合这两种方法，实际应用中一般采用这两种算法的折中，每次计算一小部分训练数据的损失函数，这一小部分数据被称之为一个batch，

**学习速率**

TensorFlow中提供了一种灵活的学习速率设置方法，指数衰减法，通过折法函数，可以先使用较大的学习率来快速得到一个比较优的解，然后随着迭代的继续逐步减少学习率，是的模型在训练后期更加稳定。

**过拟合**

为了避免过拟合，一个非常常用的方法是正则化，正则化的思想就是在损失函数中加入 模型复杂度的指标，通常正则化有两种一种为L1正则化，一种为L2正则化，无论哪一种正则化，基本的思想都是希望通过限制权重的大小，使得模型不能任意你和训练数据中的随机噪声，L2相比于L1具有不会让参数变得稀疏以及可导的优势。

还可以使用滑动平均模型，使得模型在测试数据上更健壮

**卷积神经网络**

全连神经网络处理图像的最大问题在于全连接层参数太多，参数太多容易导致计算速度减慢，还很容易导致过拟合的问题，所以需要一个更合理的神经网络结构来有效地减少神经网络中的参数个数，卷积神经网络就可以达到这个目的。

**池化层**

池化层神经网络不会改变三维矩阵的深度，但是它可以缩写矩阵的大小，池化操作可以认为是将一张分辨率较高的图片转化为分辨率较低的图片，通过池化层，可以进一步缩写最后全连接层中节点个数，从而达到减少整个神经网络中参数的目的，池化层与卷积层类似，也是通过移动一个类似的过滤器结构完成，不过池化层过滤器计算不是节点的加权和，而是采用更简单的最大值或者平均值运营，使用最大值的池化层被称之为最大池化层，还是使用最多的池化层结构，使用平均值操作的池化层被称之为平均池化层，其它池化层使用较少。

**过滤器**

过滤器中的尺寸指的是一个过滤器输入节点矩阵的大小，而过滤器的深度指的是输出节点矩阵的深度**过滤器的前向传播过程就是通过左侧小矩阵中的节点计算出右侧单位矩阵中节点的过程。为了避免前向传播得到的矩阵尺寸小于当前层矩阵的尺寸，可以再当前层矩阵的边界上加上全0填充，这样可以使得卷积层前向传播结果矩阵的大小和当前层矩阵保持一致。**

关于卷积层参数，应该说对输入层每一个深度所对应的filter的参数是一致的，且对于每个输出深度，对应的参数也是不同的，所以每个卷积层的参数=filter长度\*filter宽度\*输入层深度\*输出层深度+输出层深度，注意偏移量只与输出深度有关系，即不同输入深度的输入共享同一个偏移量，但不共享同一个filter参数

**迁移学习**

所谓迁移学习，就是将一个问题上训练号的模型通过简单的调整使其适用于一个新的问题，例如可以保留训练好的Inception-v3模型中所有卷积层的参数，只是替换最后一个层全连接层，在最后这一层全连接层之前的网络层称之为瓶颈层，将新的图像通过训练好的卷积神经网络直到瓶颈层的过程可以看成是对图像进行特征提取的过程