课题一报告书

研究员：陈向军

联系方式：15104684631

chenxiangjun19931227@gmail.com

对问题的分析：

这是一个典型的回归分析问题，样本维数是 14 维（包含时间 15 维），标签维数是 1 维（包含时间是 2 维），样本中第 4 维（TIPS-30Y）和第 6 维（UST Bill 1-Y RETURN）中有大量零值，因为数据的缺省值已经用 0 填补，所以不容易判断两个维度的数据是原本为 0 还是缺省为 0，现均按缺省值处理。样本数量是 4729,因为样本的维度不高，数量不多，采用 3-4 层的 BP 神经网络即可。

数据清洗与归一化：

经过代码测试，样本中前 6 维中 0 出现的数量所占的百分比分别是：第 1 维 0.23%，第 2 维 0.42%，第 3 维 0.42%，第 4 维 40.60%，第 5 维 0.31%，第 6 维 27.25%，其余维数中没有 0,第 4 维和第 6 维数据中缺省值比重太大，故将这 2 维的数据舍弃，保留剩下的 12 维数据。为了消除某些维数值较大，从而造成训练模型时权重较大的影响，加快模型收敛速度，对数据进行归一化处理，样本和标签均进行了归一化处理。

归一化的手段采用 Z-score 标准化方法，这种方法给予原始数据的均值（mean）和标准差（standard deviation）进行数据的标准化。经过处理的数据符合标准正态分布，即均值为 0，标准差为 1，转化函数为：

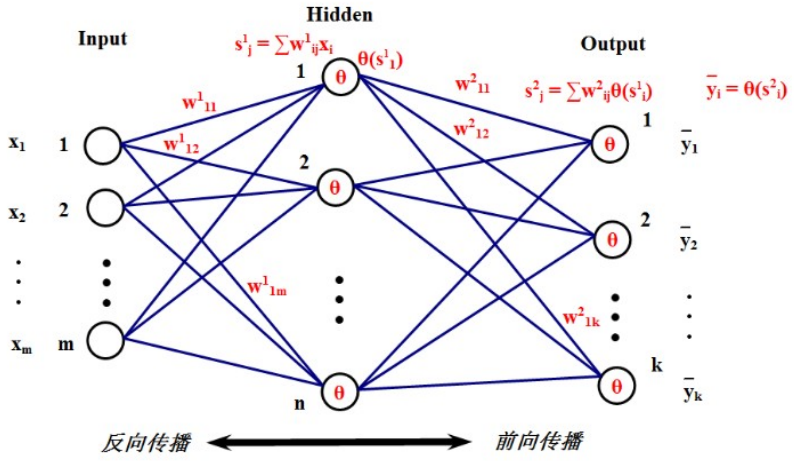
$$x^* = \frac{x - \mu}{\sigma}$$

,其中 μ 为所有样本数据的均值， σ 为所有样本数据的标准差。

算法：

采用 bp 神经网络算法并使用现在流行的 tensorflow 深度学习框架来实现算法。BP 神经网络模型拓扑结构包括输入层（input）、隐层(hide layer)和输出层(output layer)。在本次课题研究的代码中采用了 4 层网络的神经网路，两层隐含层，第一层隐含层包含 24 个神经元，第二层隐含层包含 12 个神经元。输入层 12 个神经元，输出层 1 个神经元。

BP 算法是一种有监督式的学习算法，其主要思想是：输入学向习样本，使用反传播算法对网络的权值和偏差进行反复的调整训练，使输出的向量与期望向量尽可能地接近，当网络输出层的误差平方和小于指定的误差时训练完成，保存网络的权值和偏差。BP 算法的数学推导在此就不详细论述了，只简单描述算法过程。



每层的每个神经元都有到后一层的每一个神经元都有一个权值，每个神经元接受上一层每个神经元的数值作为自己的输入，然后乘上两个神经元间的权值，最后对上一层所有神经元求和作为自己的数值并传递给下层，这是前向传播。

输出层的神经元最终会得出一个预测数值，这个数值与该样本的标签并不一定相等，一般都会有偏差，我们可以设定这个偏差函数，我们希望这个偏差最小，而且希望对所有的样本而言，这个偏差的总和最小。我们为了让偏差最小，则需要更新上一层每个神经元到当前这个神经元的所有权值，采用的方法是梯度下降，简单说就是对预测值与真实值的偏差关于权值求导，然后用-导数×步长来更新权值，但是只有最后一层有这个预测值与真实值的偏差，其他层虽然有预测，但是我们并不知道真实值，所以没有偏差，所以其他层的权值更新需要从最后一层开始使用链式相乘法则更新，这是一层一层往前回馈的过程，所以叫反向传播。

在代码中使用了 tensorflow 框架，大幅度减少了代码量。定义了损失函数（上述推导中的偏差函数）：均方差函数，1/n × (Σ(每个样本的预测值 – 每个样本的标签) ^ 2)。优化器并非选用普通的梯度下降，而是采用自适应步长的变种梯度下降法 AdamOptimizer，特点是加速收敛，又能减少震荡。使用每个变量的历史梯度值累加作为更新的分母，起到平衡不同变量梯度数值差异过大的问题，源代码如下：

```
cache += dx**2
x += -learning rate*dx/(np.sqrt(cache)+1e-7)
```

整体训练过程如下：

for（5000 次）{

用当前模型计算每层神经元的预测值（除了第一层）

用输出层的预测值计算偏差，再用链乘法计算每层的权值的修改值，从第一层权值开始修改，往后层层修改，直到倒数第二层}

模型选择问题：

隐层数：

一般认为，增加隐层数可以降低网络误差，提高精度，但也使网络复杂化，从而增加了网络的训练时间和出现“过拟合”的倾向。对于没有隐层的神经网络模型，实际上就是一个线性或非线性（取决于输出层采用线性或非线性转换函数型式）回归模型。

该课题我也尝试过进行线性拟合，但效果不是很好，预测值与真实值偏差在1%以下的比率只有3%左右，所以线性拟合效果不是很好，所以至少采用3层以上网络拟合效果更好。

隐层节点数：

在BP网络中，隐层节点数的选择非常重要，它不仅对建立的神经网络模型的性能影响很大，而且是训练时出现“过拟合”的直接原因，但是目前理论上还没有一种科学的和普遍的确定方法。 目前多数文献中提出的确定隐层节点数的计算公式都是针对训练样本任意多的情况，而且多数是针对最不利的情况，一般工程实践中很难满足，不宜采用。事实上，各种计算公式得到的隐层节点数有时相差几倍甚至上百倍。为尽可能避免训练时出现“过拟合”现象，保证足够高的网络性能和泛化能力，确定隐层节点数的最基本原则是：在满足精度要求的前提下取尽可能紧凑的结构，即取尽可能少的隐层节点数。研究表明，隐层节点数不仅与输入/输出层的节点数有关，更与需解决的问题的复杂程度和转换函数的型式以及样本数据的特性等因素有关。
在确定隐层节点数时必须满足下列条件：

(1)隐层节点数必须小于N-1（其中N为训练样本数），否则，网络模型的系统误差与训练样本的特性无关而趋于零，即建立的网络模型没有泛化能力，也没有任何实用价值。同理可推得：输入层的节点数（变量数）必须小于N-1。

(2) 训练样本数必须多于网络模型的连接权数，一般为2~10倍，否则，样本必须分成几部分并采用“轮流训练”的方法才可能得到可靠的神经网络模型。
总之，若隐层节点数太少，网络可能根本不能训练或网络性能很差；若隐层节点数太多，虽然可使网络的系统误差减小，但一方面使网络训练时间延长，另一方面，训练容易陷入局部极小点而得不到最优点，也是训练时出现“过拟合”的内在原因。因此，合理隐层节点数应在综合考虑网络结构复杂程度和误差大小的情况下用节点删除法和扩张法确定。

总结：

总而言之，层数和节点数就得靠试（我自己的理解）。最终我采用了3层网络，中间层节点数是24，因为拟合效果好于2层线性规划网络，但是4层网络（12,20,20,1）会出现过拟合，所以最终决定3层，后面验证部分有数据。

模型验证：

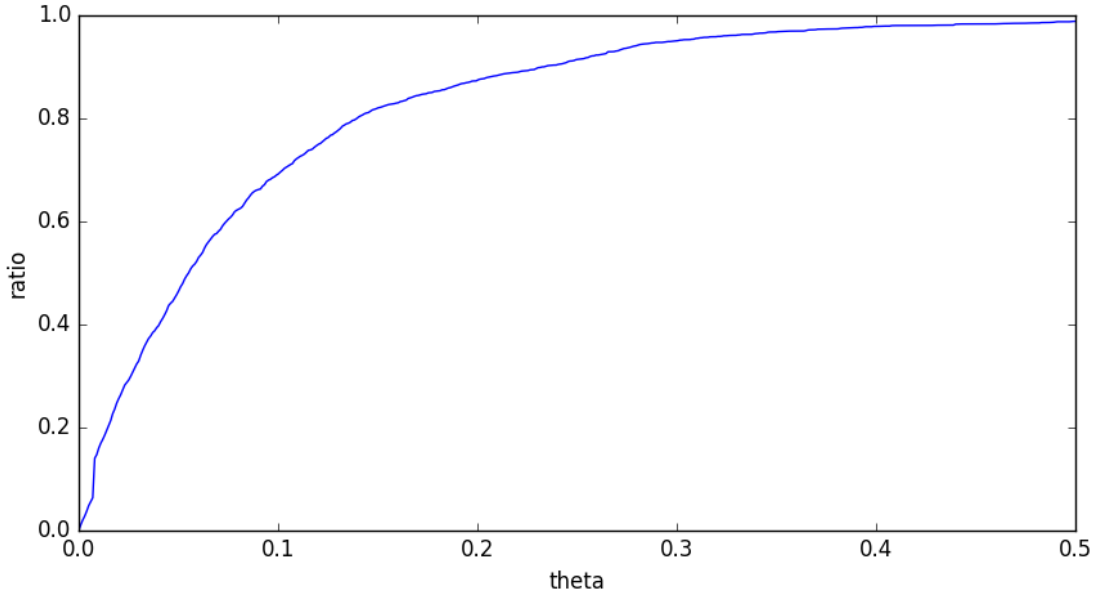
训练好模型后验证方式采用10折交叉验证：10折交叉验证是把样本数据分成10份，轮流将其中9份做训练数据，将剩下的1份当测试数据，10次结果的均值作为对算法精度的估计，通常情况下为了提高精度，还需要做多次10折交

叉验证。更进一步，还有 K 折交叉验证，10 折交叉验证是它的特殊情况。K 折交叉验证就是把样本分为 K 份，其中 K-1 份用来做训练建立模型，留剩下的一份来验证，交叉验证重复 K 次，每个子样本验证一次。

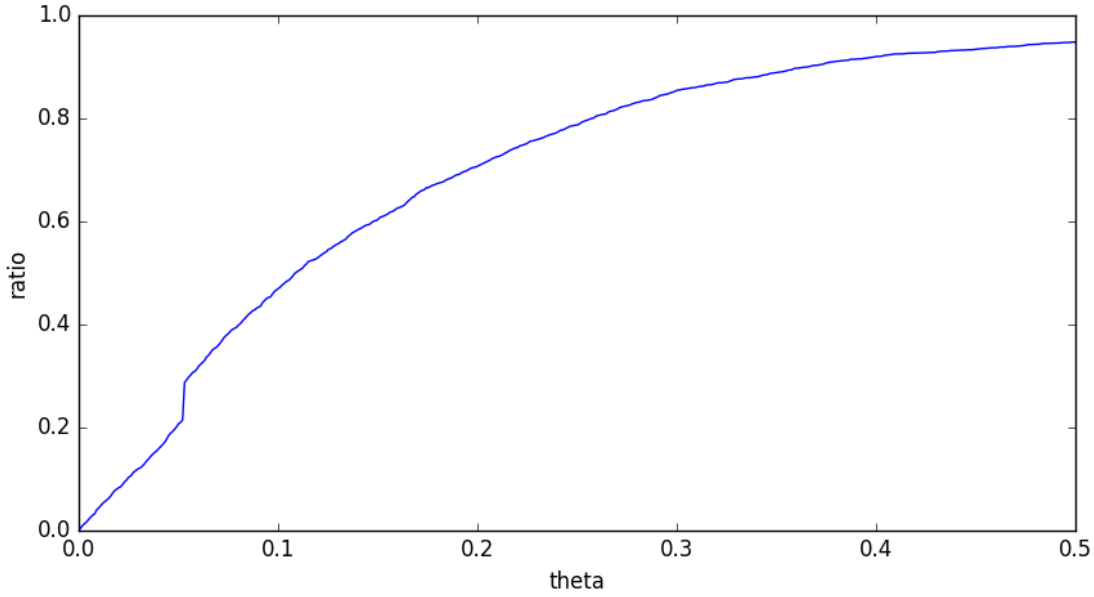
每次验证都将计算满足一定条件的样本数比率，该条件是指：|(预测值 - 真实值) / 真实值| <= Θ， Θ 是误差百分比。然后将这些比率做加权平均，权值是每次验证样本的数量/总样本数量。

模型效果如下：X 轴是 Θ 值，Y 轴是满足 Θ 的比率。

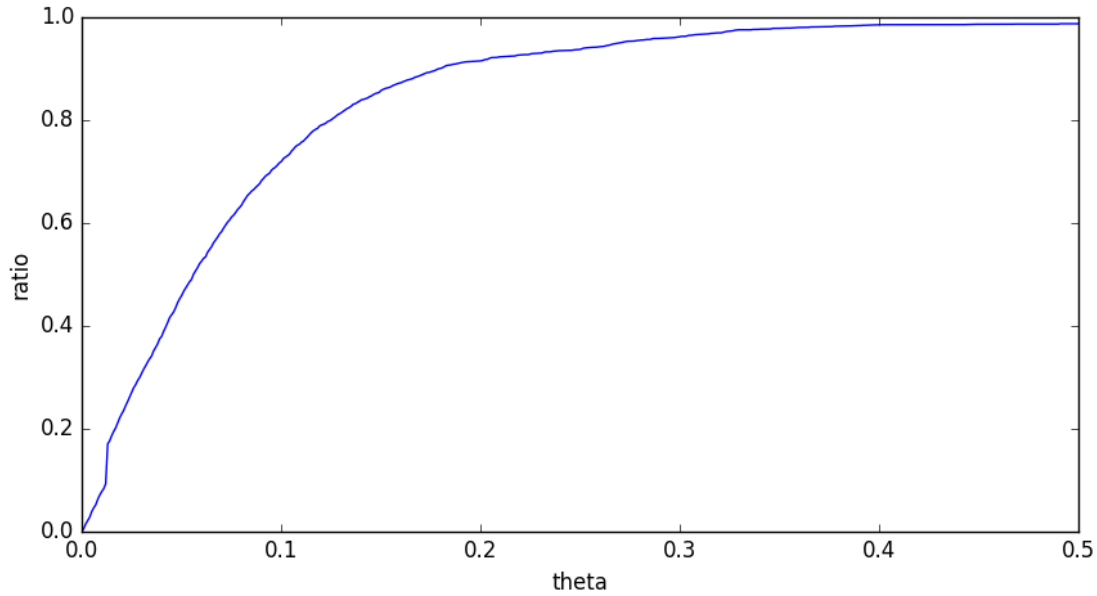
4 层神经网络，神经元个数（12,20,20,1）



3 层神经网络，神经元个数（12,60,1）



3 层神经网络，神经元个数（12,24,1）



3 层网络（12,20,1）的效果略好于 4 层网络（12,20,20，1），远好于 3 层（12,60,1）的效果。

原因分析：（12,60,1）严重过拟合，（12,20,20,1）轻度过拟合。

代码分析：

本代码是使用 python 所写，使用了一些库，运行前请提前安装这些库。分别是：

```
csv

numpy

tensorflow
sklearn
matplotlib
```

2.csv 存放样本特征，1.csv 存放样本标签，tmp/model.ckpt 文件是神经网络的模型。

代码中注释写的很清楚了，而且是 Chenglish，很好阅读。