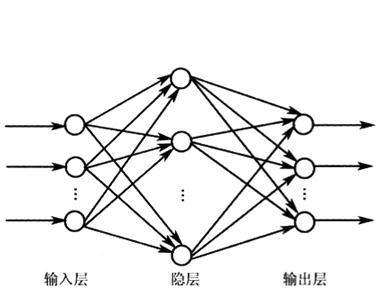
**利用前馈神经网络预测送转行情**

**一、多层前馈神经网络(Multilayer Feed-Forward Neural Network)**

多层前馈神经网络由以下部分组成（如下图）：



最左边输入层(input layer), 中间隐藏层 (hidden layer), 最右边输出层 (output layer)；每层由单元(unit)组成；输入层(input layer)是由训练集的实例特征向量传入，经过连接结点的权重(weight)传入下一层，一层的输出是下一层的输入。隐藏层的个数可以是任意的，输入层有一层，输出层有一层。根据生物学来源定义，每个单元(unit)也可以被称作神经元。以上称为2层的神经网络（输入层不算）。一层中对输入加权求和，然后根据非线性方程转化为输出。神经元对于输入的加权和通常采用以下两种sigmoid函数做非线性转化：

双曲正切函数(tanh) ：



逻辑函数(logistic function)：



作为多层前馈神经网络，理论上，如果有足够多的隐藏层(hidden layer) 和足够大的训练集, 可以模拟出任何方程。使用神经网络训练数据之前，必须确定神经网络的层数，以及每层单元的个数。特征向量在被传入输入层时通常被先标准化(normalize）到0和1之间 （为了加速学习过程）。离散型变量可以被编码成每一个输入单元对应一个特征值可能赋的值。比如：特征值A可能取三个值（a0, a1, a2), 可以使用3个输入单元来代表A；如果A=a0, 那么代表a0的单元值就取1, 其他取0；如果A=a1, 那么代表a1de单元值就取1，其他取0，以此类推。

神经网络即可以用来做分类(classification）问题，也可以解决回归(regression)问题。没有明确的规则来设计最好有多少个隐藏层，根据实验测试和误差，以及准确度来实验并改进。

**二、前馈神经网络参数设置**

**1、目标函数**

二次目标函数：



其中：c是目标函数（或者损失函数）,w是权重，b是偏置，n是训练样本数，y(s)是在输入x下的模型输出，a是在输入x下的真实值。

如果初始化的权重和偏置相对于真实权重和偏置相差太远，在sigmoid曲线图上就接近于0或者1，此时的偏导非常小。这使得学习过程在初始阶段会减慢。可以采用Cross-Entropy函数来加快学习速度。

Cross-Entropy函数：



因为Cross-Entropy函数相对于权重和偏置的偏导取决于模型输出值和真实值的差值，因此，差值很大的时候，更新量比较大；当差值比较小的时候，更新量也比较小。这一点要优于二次目标函数，学习起来也比二次目标函数快。二次目标函数更适用于神经元是线性的情况。其他情况几乎总是Cross-Entropy更优。

**2、 学习率**

学习率可以看成是梯度下降算法中所迈步子的大小。学习率过大，可能难以收敛，或者跳过极值点；学习率太小，学习速度慢。通常情况下，如果不考虑训练效率的问题，我会将学习率设的小一点。

**3、 正则化项**

神经网络训练的过程中，可能造成权重‘失真’（即，有的权重太大，放大了某些特征的作用；有的太小，掩盖了相应特征的作用）。解决的思路的是，在目标函数后面加一项，通过这一项使得权重和偏置在训练过程中变化较为稳定。通常称这一项为正则化项（Regularization）。正则化项有两种，分别称为L1，和L2。

L1:



其中，c是损失函数，λ作为正则化项的参数。上式在原来目标函数的基础上加上了一项。这一项很显然是权重加和。训练的目的是使目标函数越小越好，所以权重的加和也不会大，因而权重本身不会太大。

L2:



上式将正则化项换成了权重的平方再加和。本质上和L1具有一样的作用。本文实现上采用L2。

**4、其它改进方法**

**1）神经网络结构**

其实，最显然的一个参数是神经网络的结构。如果输入输出层是确定的，那么可以调整的是隐藏层的层数以及每一层的神经元个数。理论上，神经网络结构越复杂，越容易过拟合（overfitting）。实际上，神经网络的泛化效果挺好的。深度神经网络能够这么抢眼也证明了这一点。

**2）Softmax函数**



其中，，，分别为，第j-1层到第j层第i个神经元的权重，第j-1层到第j层第i个神经元的偏置，遍历第j层的k个神经元。Softmax函数通常用来替代最后的一层的激活函数，它具有概率的特性。Softmax函数更多地用在多层神经网络上。本文实现上还是采用sigmoid函数。

**3）扩充训练集**

为了避免过拟合，研究者们通常会采用一些方法从原来的训练集出发扩充出更多的训练样本。比如对于一幅图而言，通过微小的旋转得到一张新的图。但是很显然，我们不能随便生成证券市场的数据。

**4）Dropout**

Dropout方法的含义是，通过随机地去除神经网络上的隐藏层神经元来得到不同的训练结果，再将这些训练结果整合以避免过拟合。本质上的思路，类似于采用多个神经网络训练，再综合这些结果。目前学术界以及工业界采用的多分类器结合，或者深度森林等都借鉴了这种思想。

**5）初始参数设置**

权重初始化一般通过高斯分布（也就是正态分布）生成随机（样本）值来完成；默认的正态分布为N(0,1)，此时有大量的随机值分布在(-1,1)之外；这使得初始化的权重比较大，那么sigmoid函数的值大量地在(-1,1)之外，使得学习过程在初始阶段会非常慢。通常采用的改进策略是通过N(0, 1/sqrt(n\_in))正态分布来进行初始化，其中，sqrt(n\_in)是输入层输入个数n\_in的开方。这使得大量的随机值分布在(-1,1)之内，不会造成权重太大。本文默认采用后者。

**三、 利用前馈神经网络预测送转行情**

**1、对样本组合做初步筛选，设定基准日期为每年的9月30日。**

1.1、按T年6月30日读取数据库RHstock表wsd的变量div\_stocks，要求变量div\_stocks=0。

1.2、按T年基准日期读取数据库RHstock表wsd的变量total\_shares，要求变量total\_shares<=20亿。若基准日期不是交易日，取之前第一个交易日。

1.3、T年基准日期读取数据库RHstock表wsd的变量close\_u，要求变量close\_u≥10。若基准日期不是交易日，取之前第一个交易日。

1.4、按T年基准日期读取数据库RHstock表wsd的变量listdays\_t，要求变量listdays\_t≤720。若基准日期不是交易日，取之前第一个交易日。

1.5、按T年9月30日读取数据库RHstock表wsd的变量np\_parcomsh\_chg，要求变量np\_parcomsh\_chg≥10%。

1.6、按T年9月30日读取数据库RHstock表wsd的变量surplus\_sum，要求变量surplus\_sum≥1。

**2、对筛选后的样本，设定自变量如下：**

|  |  |
| --- | --- |
| **变量名** | **SQL字段名** |
| X1 | 按T年基准日期读取数据库RHstock表wsd的变量total\_shares；若基准日期不是交易日，取之前第一个交易日 |
| X2 | 按T年基准日期读取数据库RHstock表wsd的变量close\_u |
| X3 | 按T年基准日期读取数据库RHstock表wsd的变量listdays\_t |
| X4 | 按T年基准日期读取数据库RHstock表wsd的变量surplus\_sum |
| X5 | 按T年基准日期读取数据库RHstock表wsd的变量np\_parcomsh\_chg |
| X6 | 按T年9月30日读取数据库RHstock表wsd的变量profi\_style，若为预增，设为1，否则为0 |
| X7 | 按T年基准日期读取数据库RHstock表wsd的变量ipo\_lessoneyear，若为1，表示是次新股 |
|  |  |

因变量的设置如下：

按T年12月31日读取数据库RHstock表wsd的变量div\_stocks>=1，设为1，否则设为0。

**3、实验设置**

3.1、标准化

输入中有些特征本身值非常大，比如总股本total\_shares，将所有该特征除以阈值20亿，归一化到0-1之间。

3.2、离散变量处理

因为离散型变量本身只有相互区别的性质，没有大小可比的性质。如果直接赋值，就默认了离散型变量具有大小可比性。这可能会造成实验的不精准。在此，做如下设置：

对每一个离散型特征值(以X6（是否预增）为例)，采用两个输入量：



如果预增，则；否则。

对于输出值（是否送转，比例在1以上）同样采用两个变量：



如果送转，则；否则。

3.3、神经网络参数设置

经调试后采用如下参数设置：

|  |  |  |
| --- | --- | --- |
| 参数名 | 设置 | 描述 |
| 神经网络结构 | [9,10,10,2] | 三层神经网络，隐藏层神经元数各为10；输入层变量数为9，X6和X7为离散型变量，各增加一个变量；输出层变量数为2，输出为离散变量，增加一个变量 |
| 目标函数 | Cross-Entropy | 避免局部最优，加快学习 |
| 随机梯度下降轮动训练集 | 250 |  |
| 学习率 | 0.0000001 | 经试验，大于该值，初始训练即陷入局部最优 |
| 正则化项参数 | 0.8 | 大于或小于该值，初始训练即陷入局部最优 |

**4、将结果保存在数据库中，涉及的变量如下，每次运行时先清空表中的数据**

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | 数据库rhis1表E003b\_ stock2 | | |
| 字段 | 类型 | 描述 | 说明 |
| report\_date |  | 报告周期 | 格式为2006-12-31，自2005年开始计算 |
| report\_type |  | 报告周期类型 | 取为1231 |
| type\_group |  | 样本类型 | 若class\_weight=None，设为1；  若class\_weight= balanced，设为2 |
| group\_id |  | 分组编号 | 若p=0.5，设为0；  若p=0.6，设为1；  若p=0.7，设为2；  若p=0.8，设为3 |
| stockcode |  | 股票代码 | 每年预测得到的股票代码 |
| div\_stocks |  | 送转比例 | 所有筛选出的stockcode对应报告周期的变量div\_stocks，读取数据库RHstock表wsd\_yyyy的变量div\_stocks，若为NULL，设为0 |
|  |  |  |  |

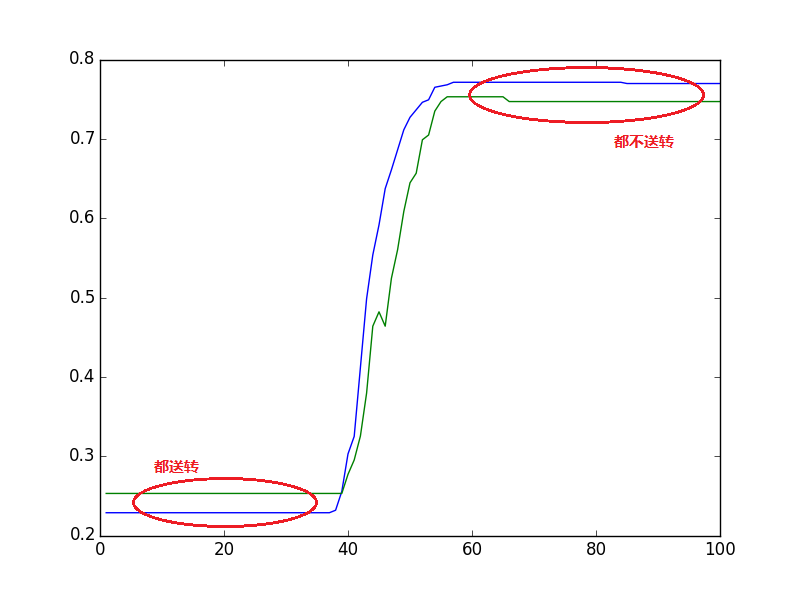
**5、统计各报告周期的送转股票占比，基于表E003b\_stock2做分析。每次运行时会先清空表中数据**

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | 数据库rhis1表E003b\_stock\_ sum2 | | |
| 字段 | 类型 | 描述 | 说明 |
| report\_date |  | 报告周期 | 读取数据库rhis1表E003b\_stock2的变量report\_date |
| type\_group |  | 样本类型 | 读取数据库rhis1表E003b\_stock2的变量type\_group |
| group\_id |  | 分组编号 | 读取数据库rhis1表E003b\_stock2的变量group\_id |
| num |  | 组合股票数量 | 将样本类型+分组编号作为一个组合，统计组合内的股票数量 |
| num\_delivery |  | 送转股票数量 | 将样本类型+分组编号作为一个组合，统计组合中变量div\_stocks大于等于1的股票数量 |
| accuracy\_rate |  | 精确率 | 将预测结果与当年变量div\_stocks>=1的股票做比较，可以得到重叠的部分。并计算准确率，其中精确率=重叠的数量÷当年预测y=1的股票数量；召回率=重叠的数量÷当年变量div\_stocks>=1的股票数量 |
| recall\_rate |  | 召回率 |  |

**四、实验结果及分析**

**1、局部最优问题**

实验过程中发现，神经网络训练过程容易陷入局部最优,如下图：



其中，蓝色的曲线是训练过程中训练集正确率（既考虑正例，也考虑负例，分母是所有实例）的变化，绿色的曲线是测试集正确率的变化。图中，红圈部分为两个局部最优解；上面的红圈训练得到的网络预测所有的股票都不送转，由于不送转占大多数，所以正确率接近80%；下面的红圈训练得到的网络预测所有的股票都送转，由于送转占极少数，所以正确率在25%左右。在这两个极端值之间，可以看到正确率曲线跳跃非常快，在不到10轮的学习之后，就从一个极端跳到另一个极端。

以上的问题由以下原因造成：

1）神经网络训练伊始，采用高斯分布随机赋初值（权重和偏置），不同的赋值极易使训练陷入局部最优；

2）正确率曲线跳跃非常快，说明送转类和不送转类之间的过度并不平滑

**2、解决方案**

计算出极端情况下的两个正确率，以这两个正确率为上下界；然后不断训练不同的神经网络，一旦正确率在上下界之内则认为是合理的神经网络。以此方法得到精确率（只考虑正例）最优的那个神经网络。预测的结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| report\_date | type\_group | group\_id | num | num\_delivery | accuracy\_rate | recall\_rate |
| 2012/12/31 | 1 | 0 | 24 | 10 | 0.416667 | 0.066225 |
| 2012/12/31 | 1 | 1 | 8 | 4 | 0.5 | 0.02649 |
| 2012/12/31 | 1 | 2 | 2 | 1 | 0.5 | 0.006623 |
| 2012/12/31 | 1 | 3 | 5 | 2 | 0.4 | 0.013245 |
| 2016/12/31 | 1 | 3 | 2 | 1 | 0.5 | 0.00565 |
| 2013/12/31 | 1 | 0 | 3 | 2 | 0.666667 | 0.011765 |
| 2013/12/31 | 1 | 1 | 5 | 2 | 0.4 | 0.011765 |
| 2013/12/31 | 1 | 2 | 2 | 1 | 0.5 | 0.005882 |
| 2013/12/31 | 1 | 3 | 143 | 40 | 0.27972 | 0.235294 |
| 2016/12/31 | 1 | 0 | 5 | 2 | 0.4 | 0.011299 |
| 2016/12/31 | 1 | 1 | 36 | 18 | 0.5 | 0.101695 |
| 2016/12/31 | 1 | 2 | 77 | 28 | 0.363636 | 0.158192 |

神经网络对class不平衡的问题有自适应性，另外随机梯度下降算法会平衡正负例个数，所以这里type\_group取1。从实验结果来看，精确率在50%上下；且预测数越多，精确率越小；反之则越大。

**3、总结**

实验结果表明送转类和不送转类之间的过度并不平滑，致使梯度下降容易陷入局部最优；不平滑在数学上的解释是：特征值差别不大，但是输出值差别很大，函数形态上表现为脉冲或者不可导的情况。

另外，最大的训练集数量为807，对于神经网络来说训练集规模较小，这也是容易陷入局部最优的原因之一。

相较于逻辑回归，神经网络的缺点在于初始化参数是随机赋值的，梯度下降的起点不稳定；逻辑回归可以直接求导，精确地计算出极值。相对于逻辑回归，神经网络对于训练集庞大、函数形态复杂但可导的情况更为有效。

**五、附录**

文件夹下文件：

data文件夹中是2011,2012,2013,2016年的送转数据；

results文件夹中是预测结果；

codes文件夹中是源码。