北京林业大学

_2019 学年─ _2020 学年第 2 学期 数据挖掘 实验报告书

专 业: 计算机(创新实验班) 班 级: 计创 17

姓 名: 黄奕超 学 号: 171002509

实验地点: ____电脑______任课教师: _ 王建新______

实验题目: 极限学习机

实验环境: Eclipse Java EE

一、实验要求

- 1. 掌握神经元网络中的几种常见的激活函数。
- 2. 掌握矩阵的常见操作,并能用程序实现。
- 3. 掌握最小二乘法,并能用程序实现。
- 4. 通过激活函数、矩阵操作、最小二乘法实现极限学习机,能够对任意变量数量的数据集进行训练,确定网络结构中的参数,并能够利用获得的确定网络进行预测。

二、实验内容和步骤

1. 实验背景介绍

相比较其他传统的神经元网络算法,极限学习机基于单隐层前馈神经网络,具有速度快、易于实现、泛化能力强等优势。对于任意输入集合,随机设定输入层结点的权重值以及隐含层偏差后,具有 h 个隐含层结点的单隐层前馈神经网络几乎能以任意区间、无限可导的非零激活函数,逼近任何一个连续的函数。极限学习机工作结构如图 1 所示。

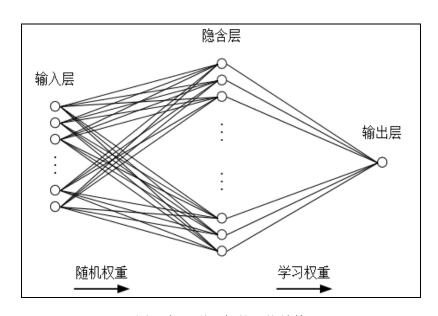


图 1 极限学习机的网络结构

激活函数可以是任意无限可到、值域在(0,1)之中的函数,如图 2 所示。

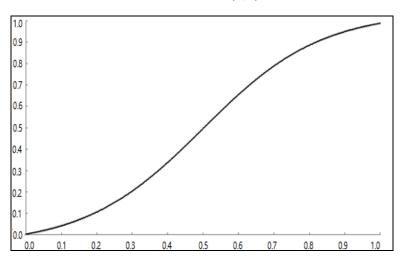


图 2 极限学习机的一种激活函数

我们这里把这个激活函数固定下来,如式(1)所示。

$$g(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

图 1 中,输入层和隐含层之间有权重;隐含层和输出层之间也有权重。但是,输入层和隐含层之间的权重可以随机设定,包括截距!这是极限学习机和普通的神经元网络最大的不同之处。

假设一共有n 行输入,而隐含层的节点数量(是输入变量数量的2~3 倍左右)有h 个,根据已经设定的权重,我们会有:

$$(H_{i,1}, H_{i,2}, \dots, H_i, Q_i), \quad i = 1,2,\dots, n$$
 (2)

对于数据集中的第 i 个输入, $H_{i,j}$ 是各个隐含节点的输出, Q_i 是第 i 个真实输出值。注意,所有输入节点汇聚到一个节点之后,除了乘以加权系数之外,还要有一个截距。

假设隐含节点的各个输出到最终输出之间的连接权重是 w_j , j=1, 2, ..., h,则 我们要求得一组 w,使得下式取得最小值:

$$\min_{w_1, w_2, \dots, w_h} \sum_{i=1}^n \left(Q_i - \sum_{j=1}^h w_j H_{i,j} - d \right)^2$$
(3)

求解这一最小值问题,恰好就是最小二乘法的功能。其中 d 是截距。求解这一问题的解法是:

$$\widehat{W} = (H^T H)^{-1} H^T Q \tag{4}$$

式(4)中的 H 要比(3)中的 H 多一个变量 d。

2. 输入输出及具体要求

建立极限学习机之后,对附件中训练数据进行训练,获得极限学习机的网络结构和模型,然后对预测数据进行预测。

预测数据一共 10 条,通过预测,会得到 10 个预测结果,这些结果要用文本的形式在一行中显示。为了和真实结果进行对比,不能用图片的形式显示预测结果。

实验报告中要有训练数据的模型结果和真实结果差的平方的平均值,然后求平方根之后的结果。

实现方法、实验结果及结论分析等:

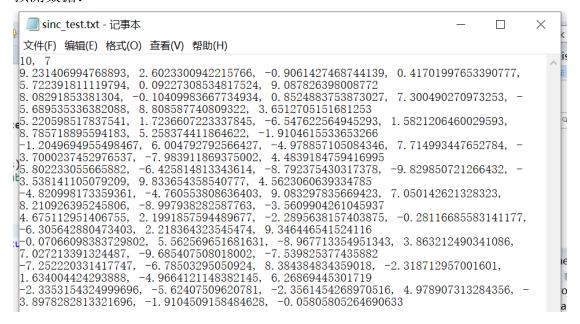
1. 实现方法

数据处理上,预先得到训练数据和预测数据集的行列数,并在文件中写入方便程序运行。在读取数据时分行读取,使用 split()和正则表达式将字符转换成浮点数。同时预先设定好隐含层数量。

输入层和隐含层之间的权重随机生成。运用给定的激活函数和训练数据生成模型,求解训练数据差的平方的平均值开根的值。然后再用得到的模型计算预测数据。

2. 实验结果

预测数据:



训练数据:

```
🥘 train.txt - 记事本
                                                                                                                                                            X
 文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
4.542315721460088, 0.6189090172684821, -6.79586870133948, -6.230426604445654, 0.33204160253691484, -4.64267583159883, -4.772779311432071, -2.041021682636613
-8.982652858886201, 6.0743108992079975, 5.241331623116569, 0.6085920802035201,
0. 5919085134919797, -0. 2563383430927466, 6. 055886481305414, 0. 15546833824005285
-9. 529696762346774, 2. 4919010520961145, 9. 73555362157148, -0. 3206310439077793,
9. 815057753359715, -1. 9064123073225563, -4. 551871879833465, -0. 015514720005059532 -6. 779699395252672, 7. 478875353099859, 4. 048509021263055, 3. 2971047692147177,
7. 471634302479806, 1. 8588271019253106, 2. 8757312593228406, -0. 1186082523682146
-6. 203244589367172, -3. 546759494124303, 4. 090437975557688, 2. 0681282492389705, 1. 4374575936352212, 8. 192129976814254, -5. 757406817445632, -1. 864346934528962
1. 721074525202426, 5. 42643802242437, 5. 287439934029415, -1. 0230717740171918
0. 7533957851260613, -7. 388785008117313, -3. 3129884873818227, -4. 3199162598079965,
4. 935838350761427, 9. 603838215753893, 8. 614719674431804, 0. 03418310922022301
9. 136821456028912, -5. 239154225041711, -9. 316874392704783, 2. 8248710169171822, -
0. 7958199873702974, -7. 5743919006297356, -8. 06074996231596, -0. 2862990876772977
 -8. 127465279756976, -2. 2741733864622375, 4. 779824411527265, 9. 603155249217398,
9. 383499360739204, 6. 552080065135748, -2. 080234823150942, -2. 901866637041264
1. 9844156736051985, -0. 6347385129532874, -9. 675932410701682, 5. 72408755042202, 6. 161542767479698, 2. 289504542192965, -5. 482769936649616, -0. 00554639220809338
 -5. 463407124945354, 6. 28325252074459, 9. 058271631890385, -5. 44223762507825,
2. 7637130854992726, -0. 9693686046258208, -9. 746404861867013, 0. 06617390822436862 4. 579502842545818, 9. 095257213561329, 3. 606524055728668, -2. 1589213746006486,
0. 2676789507721473, 5. 110031135046301, -1. 4401141081672115, 0. 035272832339319354  
0. 8215395530688401, -0. 760173950947479, -6. 097751772709592, -4. 180752659459534,
3. 161497322864829, -5. 456894628513915, 6. 81961619275857, -3. 588068534900894
3. 500839979215371, 6. 197935963926032, -7. 8988502969552705, -7. 47169867549045,
3. 373340772844882, -9. 064561499081465, 1. 3224708700505339, -4. 849318012759726

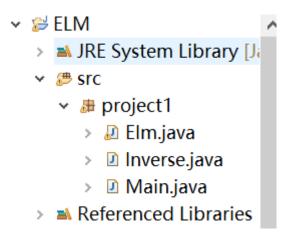
7. 173040061765018, -3. 6973957253485645, 7. 001432992292642, 9. 586375948675013,

1. 9085801858754934, 9. 56614350797173, 4. 497961116470568, -3. 811461486295754
-4. 048761600503825, -4. 8960573396184355, -6. 381448721682654, -7. 413977600989208, 7. 183876514262831, 7. 561627822318666, 2. 7054264314466216, -0. 036672129348016905 -5. 638787728349555, -9. 912295099000108, -2. 381922581856603, -6. 335188956139897,
8. 57297026379063, 8. 11122347974755, 0. 6030760385674245, 0. 16396929931252935

-9. 67847642383787, -6. 455125487815776, 7. 347319435652313, -8. 090042798034817,

7. 071194839407447, 8. 83742738983537, -2. 71224127490078, 0. 005220527496925305
-3. 5308736181609612, 4. 56720967453214, 8. 244748207773458, -4. 923842594164906, 1. 6652831461247857, 2. 865178189201231, -5. 719907863938765, -0. 1278025022721975 8. 506623037537643, 7. 46603966613668, 2. 9459430539671256, -1. 3949046525686004, 2. 6441653248005537, 8. 564797787825277, 6. 229564329584129, -0. 05624029233074812 5. 541639840264308, 7. 696878446891656, -7. 499107494795081, -6. 490895218580417,
```

具体函数实现:



输出结果:

- -0.760898993059638, -0.7525005750940392, -0.7832734035173019,
- -1.746500835414412, -0.057260995547640636, 0.32602761749665027,
- -0.8179412493986741, -0.26760419407746733, -0.34208396449980416,

-0.08826833548622881

Training_Accuracy = 2.1859267441190324

3. 结论分析

设定的隐含层数量会影响得到的结果,同时因为输入到隐含层的权值随机产 生,最终得到的结果也有所不同。

不足:

算法的数学原理理解仍有不足,因此查阅了相关资料和一些博客的经验。对于具体应用上的概念还比较模糊,同时功能、界面上可以进一步完善和加强。