

# 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。

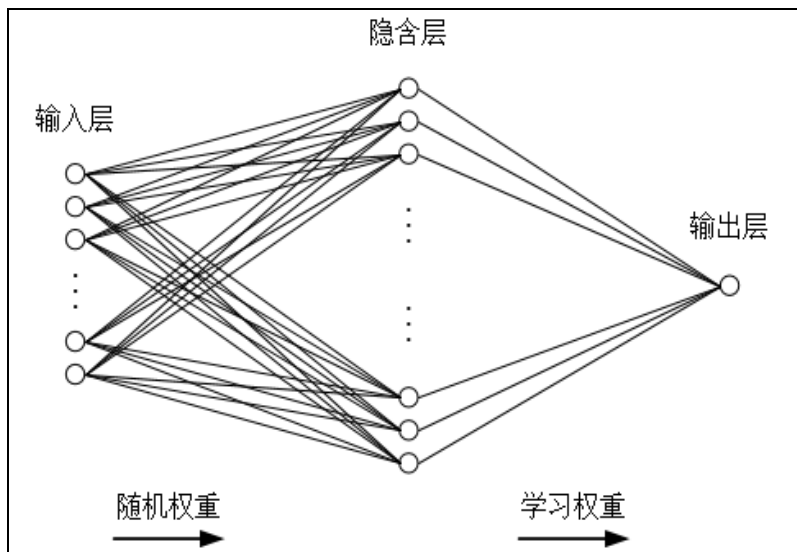


图 1 极限学习机的网络结构

激活函数可以是任意无限可导、值域在(0,1)之中的函数，如图 2 所示。

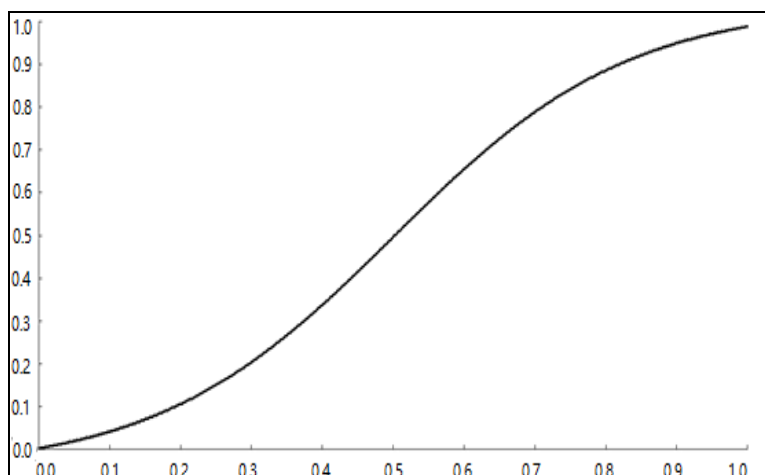


图 2 极限学习机的一种激活函数

我们这里把这个激活函数固定下来，如式(1)所示。

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

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：

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

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  $w_j, j=1, 2, \dots, h$ ，则我们要求得一组  $w$ ，使得下式取得最小值：

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

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：

$$\hat{w} = \left( H^T H \right)^{-1} H^T Q \quad (4)$$

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。

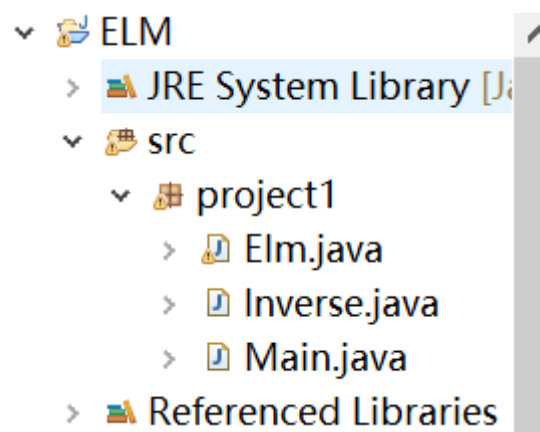
## 2. 实验结果

预测数据:

```
sinc_test.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
10, 7
9.231406994768893, 2.6023300942215766, -0.9061427468744139, 0.41701997653390777,
5.722391811119794, 0.09227308534817524, 9.087826398008772,
8.08291853381304, -0.10409983667734934, 0.8524883753873027, 7.300490270973253, -
5.689535336382088, 8.808587740809322, 3.6512705151681253
5.220598517837541, 1.7236607223337845, -6.547622564945293, 1.5821206460029593,
8.785718895594183, 5.258374411864622, -1.9104615533653266
-1.2049694955498467, 6.004792792566427, -4.978857105084346, 7.714993447652784, -
3.7000237452976537, -7.983911869375002, 4.4839184759416995
5.802233055665882, -6.425814813343614, -8.792375430317378, -9.829850721266432, -
3.538141105079209, 9.833654358540777, 4.5623060639334785
-4.820998173359361, -4.760553808636403, 9.083297835669423, 7.050142621328323,
8.210926395245806, -8.997938282587763, -3.5609904261045937
4.675112951406755, 2.1991857594489677, -2.2895638157403875, -0.28116685583141177,
-6.305642880473403, 2.218364323545474, 9.346446541524116
-0.07066098383729802, 5.562569651681631, -8.967713354951343, 3.863212490341086,
7.027213391324487, -9.685407508018002, -7.539825377435882
-7.252220331417747, -6.78503295050924, 8.384384834359018, -2.318712957001601,
1.634004424293888, -4.9664121148382145, 6.26869445301719
-2.3353154324999696, -5.62407509620781, -2.3561454268970516, 4.978907313284356, -
3.8978282813321696, -1.9104509158484628, -0.05805805264690633
```

训练数据:

具体函数实现：

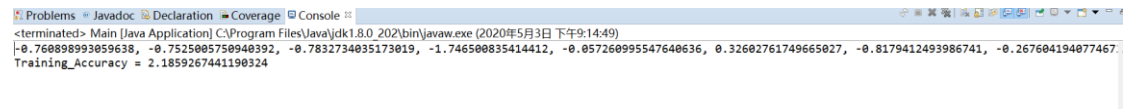


输出结果：

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

-0.08826833548622881

Training\_Accuracy = 2.1859267441190324



The screenshot shows an IDE console window with the following text:

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (2020年5月3日 下午9:14:49)
|-0.760898993059638, -0.7525005750940392, -0.7832734035173019, -1.746500835414412, -0.057260995547640636, 0.32602761749665027, -0.8179412493986741, -0.267604194077467
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。