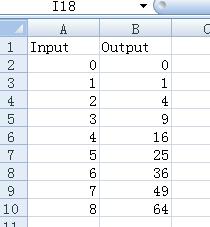
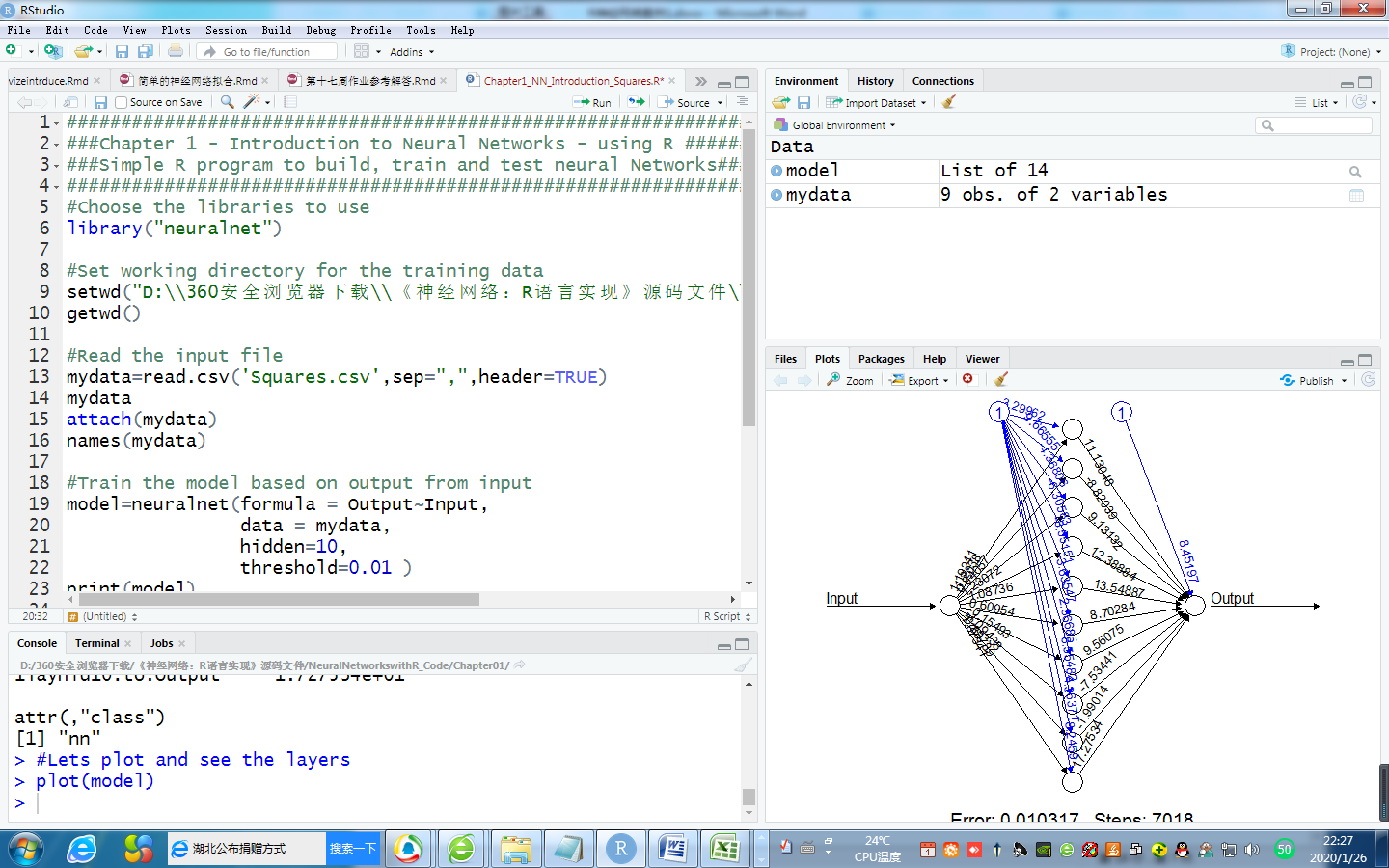
# 神经网络的基本实验(基于R语言)

## 用神经网络拟合一个平方函数

考虑一个数字平方的简单数据集，用R中的neuralnet函数训练，测试该神经网络的准确性。





先安装库：> install.packages("neuralnet")

代码如下：

#Choose the libraries to use

library("neuralnet")

#Set working directory for the training data

setwd("D:\\360安全浏览器下载\\《神经网络：R语言实现》源码文件\\NeuralNetworkswithR\_Code\\Chapter01")

getwd()

#Read the input file

mydata=read.csv('Squares.csv',sep=",",header=TRUE)

mydata

attach(mydata)

names(mydata)

#Train the model based on output from input

model=neuralnet(formula = Output~Input,

data = mydata,

hidden=10,

threshold=0.01 )

print(model)

#Lets plot and see the layers

plot(model)

#Check the data - actual and predicted

final\_output=cbind (Input, Output,

as.data.frame(model$net.result) )

colnames(final\_output) = c("Input", "Expected Output",

"Neural Net Output" )

print(final\_output) #打印最终输出，将预测值和实际值进行比较

#########################################################################

> mydata

Input Output

1 0 0

2 1 1

3 2 4

4 3 9

5 4 16

6 5 25

7 6 36

8 7 49

9 8 64

> names(mydata)

[1] "Input" "Output"

> model=neuralnet(formula = Output~Input,

+ data = mydata,

+ hidden=10,

+ threshold=0.01 )

> print(model) #打印模型

$call

neuralnet(formula = Output ~ Input, data = mydata, hidden = 10,

threshold = 0.01)

$response

Output

1 0

2 1

3 4

4 9

5 16

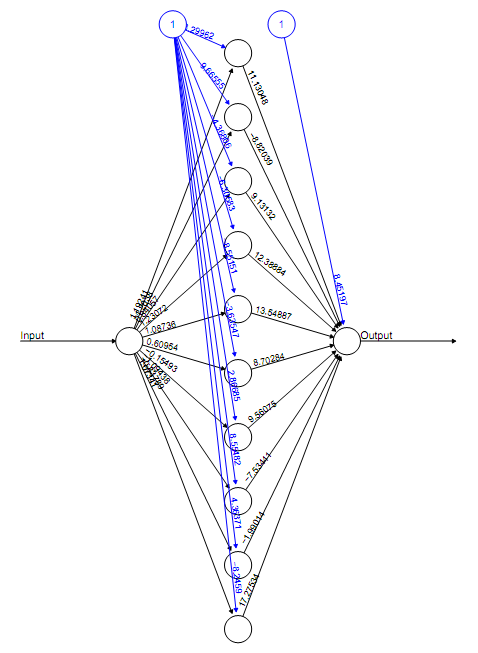
6 25

7 36

8 49

9 64

|  |
| --- |
| $covariate    [1,] 0  [2,] 1  [3,] 2  [4,] 3  [5,] 4  [6,] 5  [7,] 6  [8,] 7  [9,] 8  $model.list  $model.list$response  [1] "Output"  $model.list$variables  [1] "Input"  $err.fct  function (x, y)  {  1/2 \* (y - x)^2  }  <bytecode: 0x000000000ca16e00>  <environment: 0x000000000ca13500>  attr(,"type")  [1] "sse"  $act.fct  function (x)  {  1/(1 + exp(-x))  }  <bytecode: 0x000000000ca1ccb0>  <environment: 0x000000000ca1e028>  attr(,"type")  [1] "logistic"  $linear.output  [1] TRUE  $data  Input Output  1 0 0  2 1 1  3 2 4  4 3 9  5 4 16  6 5 25  7 6 36  8 7 49  9 8 64  $exclude  NULL  $net.result  $net.result[[1]]  [,1]  [1,] -0.05602375  [2,] 1.10323319  [3,] 3.92560753  [4,] 9.03472799  [5,] 15.99026209  [6,] 25.00143358  [7,] 36.00007584  [8,] 48.99966958  [9,] 63.99991791  $weights  $weights[[1]]  $weights[[1]][[1]]  [,1] [,2] [,3] [,4] [,5]  [1,] -3.299624 9.665552 -4.3680622 -6.305827 -8.551514  [2,] 1.192408 -1.203804 0.8405749 1.230725 1.087357  [,6] [,7] [,8] [,9] [,10]  [1,] -3.635474 2.8668504 8.554821 4.3637084 -8.245900  [2,] 0.609537 -0.1549282 -1.094378 -0.8378853 1.073415  $weights[[1]][[2]]  [,1]  [1,] 8.451967  [2,] 11.130478  [3,] -8.820390  [4,] 9.131319  [5,] 12.388840  [6,] 13.548872  [7,] 8.702839  [8,] 9.560748  [9,] -7.534412  [10,] -1.990137  [11,] 17.275339  $generalized.weights  $generalized.weights[[1]]  [,1]  [1,] -11.259104102  [2,] -15.920224648  [3,] -0.345563427  [4,] -0.084230042  [5,] -0.032838289  [6,] -0.016888006  [7,] -0.009387028  [8,] -0.006092388  [9,] -0.003661938  $startweights  $startweights[[1]]  $startweights[[1]][[1]]  [,1] [,2] [,3] [,4] [,5]  [1,] -0.1093914 0.7053004 0.2122929 0.7289706 -0.2024983  [2,] -0.4422097 -2.0688075 0.6743820 1.0867434 0.8678274  [,6] [,7] [,8] [,9] [,10]  [1,] 0.5163396 0.5873401 -0.1106267 -0.579395758 -0.09911735  [2,] -1.6632861 -1.2461571 -1.8254337 0.002722529 1.67032930  $startweights[[1]][[2]]  [,1]  [1,] -0.42615881  [2,] 1.04767042  [3,] -1.30975875  [4,] -0.95135636  [5,] 1.09069890  [6,] -0.19115110  [7,] 1.77920916  [8,] 0.55333498  [9,] -0.02407916  [10,] -0.68715660  [11,] -0.30494979  $result.matrix  [,1]  error 1.031651e-02  reached.threshold 9.861602e-03  steps 7.018000e+03  Intercept.to.1layhid1 -3.299624e+00  Input.to.1layhid1 1.192408e+00  Intercept.to.1layhid2 9.665552e+00  Input.to.1layhid2 -1.203804e+00  Intercept.to.1layhid3 -4.368062e+00  Input.to.1layhid3 8.405749e-01  Intercept.to.1layhid4 -6.305827e+00  Input.to.1layhid4 1.230725e+00  Intercept.to.1layhid5 -8.551514e+00  Input.to.1layhid5 1.087357e+00  Intercept.to.1layhid6 -3.635474e+00  Input.to.1layhid6 6.095370e-01  Intercept.to.1layhid7 2.866850e+00  Input.to.1layhid7 -1.549282e-01  Intercept.to.1layhid8 8.554821e+00  Input.to.1layhid8 -1.094378e+00  Intercept.to.1layhid9 4.363708e+00  Input.to.1layhid9 -8.378853e-01  Intercept.to.1layhid10 -8.245900e+00  Input.to.1layhid10 1.073415e+00  Intercept.to.Output 8.451967e+00  1layhid1.to.Output 1.113048e+01  1layhid2.to.Output -8.820390e+00  1layhid3.to.Output 9.131319e+00  1layhid4.to.Output 1.238884e+01  1layhid5.to.Output 1.354887e+01  1layhid6.to.Output 8.702839e+00  1layhid7.to.Output 9.560748e+00  1layhid8.to.Output -7.534412e+00  1layhid9.to.Output -1.990137e+00  1layhid10.to.Output 1.727534e+01  attr(,"class")  [1] "nn"  > #Lets plot and see the layers  > plot(model) #绘制神经网络 |
|  |
| |  | | --- | | > | |



包neuralnet简介：

版本1.33,2016年8月

neuralnet(formula, data, hidden = 1, threshold = 0.01,

stepmax = 1e+05, rep = 1, startweights = NULL,

learningrate.limit = NULL, learningrate.factor = list(minus = 0.5,

plus = 1.2), learningrate = NULL, lifesign = "none",

lifesign.step = 1000, algorithm = "rprop+", err.fct = "sse",

act.fct = "logistic", linear.output = TRUE, exclude = NULL,

constant.weights = NULL, likelihood = FALSE)

formula：需要拟合的模型的符号描述

data：包含公式中指定的变量的数据集

hidden：一个整型变量，指定每个层中隐含的神经元的数量

threshold：指定误差函数的偏导数的阈值的数值，即为停止条件

stepmax：训练神经网络的最大步骤，到达这个最大值，神经网络的训练过程会停止

rep：神经网络训练的重复次数

startweights：包含权重的初始值向量。权重不会随机初始化。

learningrate.limit：一个包含学习率最低和最高限制的向量或列表

learningrate.factor：一个包含用于上限和下限学习率的乘法因子的向量或列表。

learningrate：指定传统反向传播所使用的学习率数值，仅用于传统的反向传播。

lifesign：一个字符串，指定神经网络计算过程中的打印功能

algorithm：包含算法类型的字符串，用来计算神经网络；

err.fct：用于计算误差的可微分函数

linear.output：逻辑值

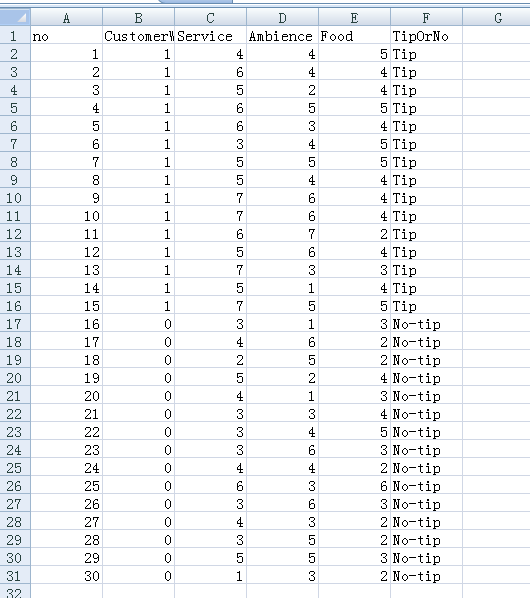
exclude：一个向量或矩阵，指定从计算中排除的权重

constant.weights：一个向量，指定从训练过程中排除的固定的权重值

likelihood：逻辑值

## 第二个例子，客户分类

客户被要求对以下方面打分，服务，氛围，食物，结果变量为是否付小费tip。



这是3个输入和一个分类输出的分类问题。

代码如下：

###Choose the libraries to use

library(NeuralNetTools)

library(nnet)

###Set working directory for the training data

setwd("D:\\360安全浏览器下载\\《神经网络：R语言实现》源码文件\\NeuralNetworkswithR\_Code\\Chapter01")

getwd()

###Read the input file

mydata=read.csv('RestaurantTips.csv',sep=",",header=TRUE)

mydata

attach(mydata)

names(mydata)

##Train the model based on output from input

model=nnet(CustomerWillTip~Service+Ambience+Food,

data=mydata,

size =5,

rang=0.1,

decay=5e-2,

maxit=5000)

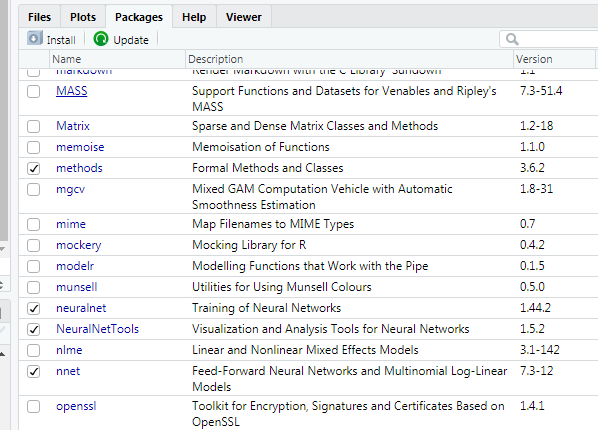
print(model)

plotnet(model)

garson(model)

########################################################################

安装记载包：



> mydata=read.csv('RestaurantTips.csv',sep=",",header=TRUE)

> mydata

no CustomerWillTip Service Ambience Food TipOrNo

1 1 1 4 4 5 Tip

2 2 1 6 4 4 Tip

3 3 1 5 2 4 Tip

4 4 1 6 5 5 Tip

5 5 1 6 3 4 Tip

6 6 1 3 4 5 Tip

7 7 1 5 5 5 Tip

8 8 1 5 4 4 Tip

9 9 1 7 6 4 Tip

10 10 1 7 6 4 Tip

11 11 1 6 7 2 Tip

12 12 1 5 6 4 Tip

13 13 1 7 3 3 Tip

14 14 1 5 1 4 Tip

15 15 1 7 5 5 Tip

16 16 0 3 1 3 No-tip

17 17 0 4 6 2 No-tip

18 18 0 2 5 2 No-tip

19 19 0 5 2 4 No-tip

20 20 0 4 1 3 No-tip

21 21 0 3 3 4 No-tip

22 22 0 3 4 5 No-tip

23 23 0 3 6 3 No-tip

24 24 0 4 4 2 No-tip

25 25 0 6 3 6 No-tip

26 26 0 3 6 3 No-tip

27 27 0 4 3 2 No-tip

28 28 0 3 5 2 No-tip

29 29 0 5 5 3 No-tip

30 30 0 1 3 2 No-tip

> attach(mydata)

> names(mydata)

[1] "no" "CustomerWillTip" "Service"

[4] "Ambience" "Food" "TipOrNo"

> ##Train the model based on output from input

> model=nnet(CustomerWillTip~Service+Ambience+Food,

+ data=mydata,

+ size =5,

+ rang=0.1,

+ decay=5e-2,

+ maxit=5000)

# weights: 26

initial value 7.527679

iter 10 value 6.050387

iter 20 value 5.324842

iter 30 value 5.197885

iter 40 value 5.160726

iter 50 value 5.142697

iter 60 value 5.141790

iter 70 value 5.141779

final value 5.141778

converged

> print(model)

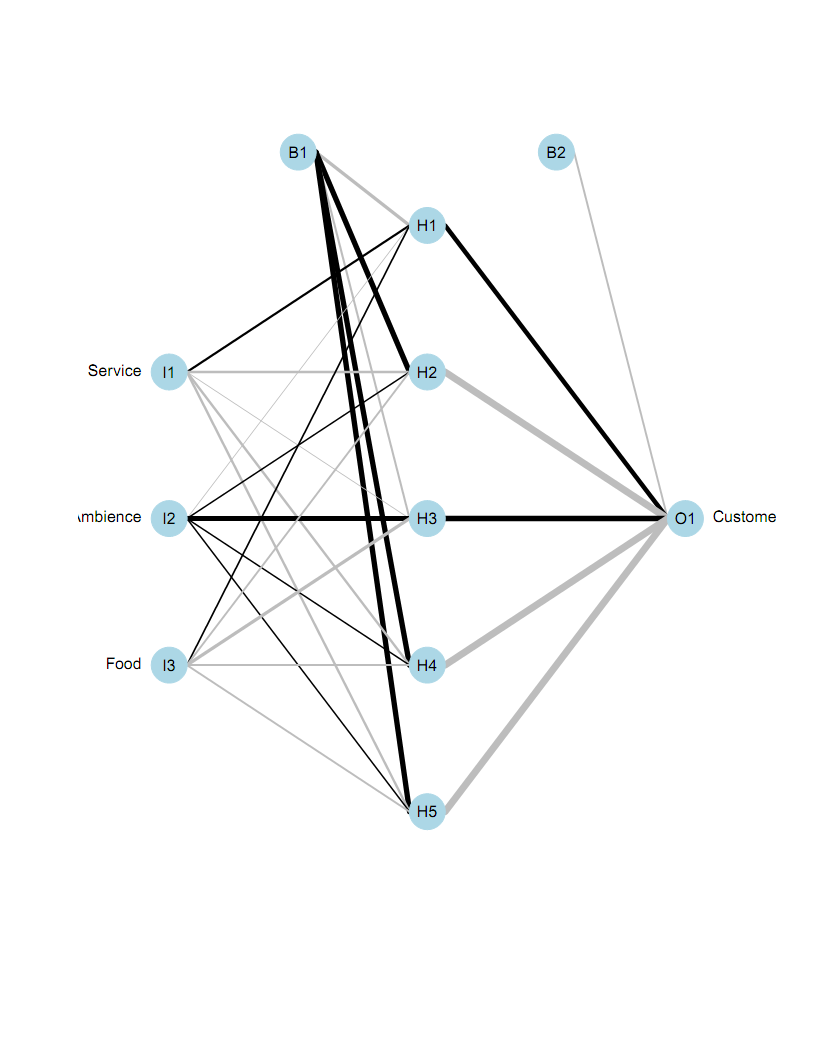
a 3-5-1 network with 26 weights

inputs: Service Ambience Food

output(s): CustomerWillTip

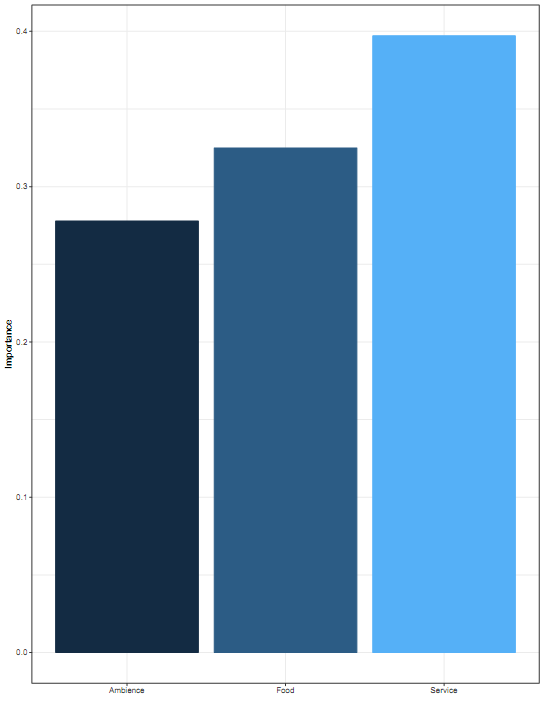
options were - decay=0.05

> plotnet(model)



> garson(model)

使用工具包NeuralNetTools，可以通过garson算法获取神经网络中的输入变量的相对重要性，下图中可以看出客户享受的服务对给不给小费影响大。



nnet包：前馈神经网络和多项对数线性模型

用于具有单个隐含层的前馈神经网络和多项对数线性模型的软件

nnet(formula, data, weights, subset, na.action, contrasts = NULL)

formula：形如~ X1+X2+...的公式

data：公式中指定的变量所在的数据框

subset:指定训练样本中要使用的案例的索引向量

na.action:指定如果找到NA时要执行的函数

contrasts：模型公式中作为变量出现的一些或全部因子的对比列表。

## 神经网络的优缺点

神经网络实现非常简单，但内部结果相对复杂。可以归纳为一个通用的数学函数逼近器。神经网络是一种机器学习技术，是数据驱动的。神经网络以构建MLP为基础，包括权重，偏置，激活函数，前向和后向传播等基础知识。神经网络是非线性的和非参数的。神经网络可以用于有监督的和无监督的学习。

神经网络作为数据科学，机器学习和预测的首选建模技术。

优点：灵活，可用于分类和回归问题，任何可以数值化的数据都可以用在神经网络模型中，因为神经网络是具有逼近函数的数学模型；对大量输入的非线性数据具有很好的建模，例如图像；一旦经过训练，预测就会变得相当快；可以使用任意数量的输入和层来训练神经网络。

缺点：神经网络是黑盒子，意味着不知道每个独立变量对因变量的影响程度。

使用CPU进行训练，计算成本非常高；依赖于训练数据，会导致过拟合和泛化的问题。

参考文献，《神经网络，R语言实现》，朱塞佩.查博罗著，李洪成译，机械工业出版社，201807.