Guide of Python

内容概要： 数学建模算法

创建时间： 2022/4/7 13:41 **更新时间：** 2022/4/17 15:27

作者： **TwinkelStar**

BP 神经网络

BP Neural Network

1、操作系统相关环境

1）硬件环境：

- 电脑

2）软件环境：

- Python3.7(向下兼容 Python3)(程序设计语言)
- Numpy1.19.5(兼容大部分版本)(科学计算库)
- Tensorflow2.3（深度学习库）
- Matplotlib3.4.2 （数据可视化库）

3）操作系统(2 选 1)：

- Windows7
- Windows10
- Windows11

2、BP 神经网络

神经网络在特征提取和建模上都有着相较于浅层模型显然的优势。深度学习善于从原始输入数据中挖掘越来越抽象的特征表示，而这些表示具有良好的泛化能力。不仅仅在数学建模中有着广泛的应

用，在目标检测、计算机视觉、自然语言处理等领域成效卓然。

1）基本原理

生物神经系统是一个有高度组织和相互作用的数量巨大的细胞组织群体。神经细胞也称为神经元，是神经系统的基本单元，他们按照不同的结合方式构建了复杂的神经网络。而我们人工神经网络就是模拟生物的神经网络，通过模拟出像生物神经网络那样，使得我们的智能体具有学习、记忆和认知等各种智能，图 1 为最基础全连接网络，只含有一个输入层，一个隐藏层，一个输出层的 BP 神经网络：

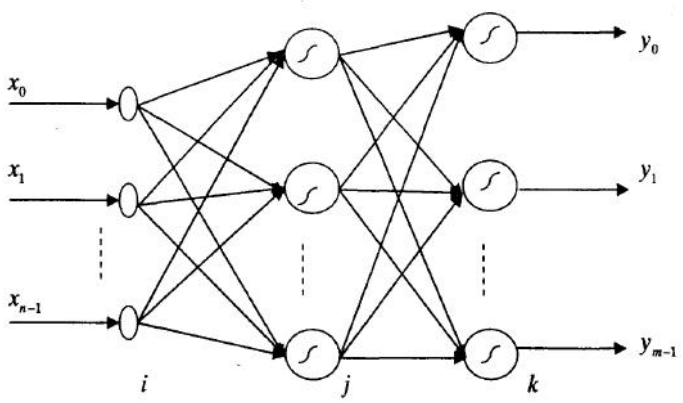

图 1.BP 神经网络的拓扑结构
fig1.BP Neural Network and Topology

但就针对数学建模而言，复杂的神经网络并不是都能用在建模中，我们可以搭建一些简单的网络求解问题。

BP 神经网络的学习过程是信号的正向传播（图像也是一种信号）与误差的反向传播两个过程组成。正向传播时，就像图 1 所示，由 x_0 作为输入，输入信号从输入层经过隐藏层，最后传向输出层，输出 y_0 在输出端产生输出信号。如果在输出层不能得到期望值，则转入误差信号反向传播，网络的权值由误差反馈调节，通过权值的不断迭代更新，最终修正网络的实际输出更接近期望输出，图 2 为算法流程图：

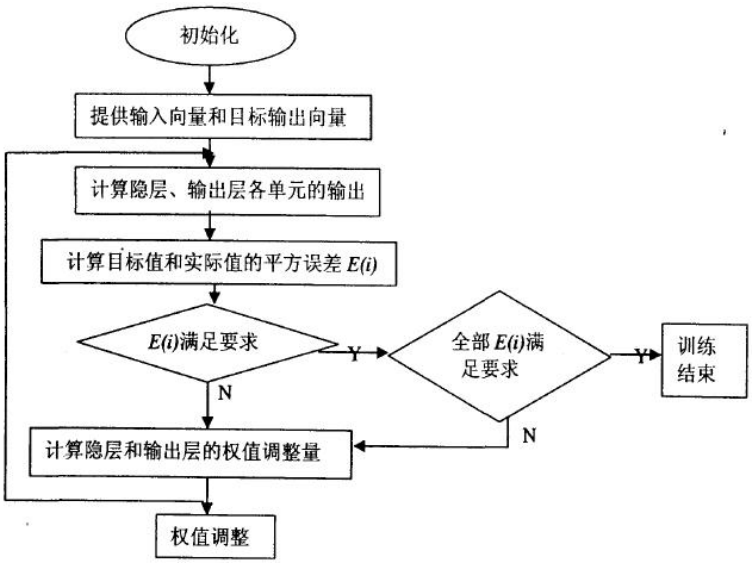

图 2.BP 神经网络流程图
fig2.BP Neural Network Charts

2）BP 算法步骤

第一步：权值初始化:w_{sq}=Random(")，sq 为 ij, jk;

第二步：依次输入 P 个学习样本。设当前输入为 P 个样本;

第三步：依次计算各层的输出:x_j，y_k，j=0，l，2，...，n_1，k=0，l，2，...，m-1。

第四步:求各层的反传误差:

$$\delta_{jk}^{(p)} = (d_k^{(p)} - y_k^{(p)}) y_k^{(p)} (1 - y_k^{(p)}), k = 0,1,\cdots,m-1$$

$$\delta_{ij}^{(p)} = \sum_{k=0}^{m-1} \delta_{jk}^{(p)} w_{jk}' x_j^{(p)} (1 - x_j^{(p)}), \quad j = 0,1,\cdots,n_1$$

图 3.BP 神经网络算法步骤
fig3.BP Neural Network Algorithm Steps

第五步：记录已学习过的样本数 P。如果 p<P，转到第二步继续计算；如果 p=P，转到第六步；

第六步：按权值修正公式修正各层的权值和阂值；

第七步：按新的权值再计算，若达到最大学习次数。则终止学习。否则转到第二步，继续新一轮的学习。

3、开源工具

因为从零开始搭建网络的过程及其复杂，数学建模时间有限，我们使用一些开源的工具可以更快的搭建网络模型，得到实验结果，以开源的深度学习框架 Tensorflow 为例：Tensorflow 是由谷歌公司开源的神经网络框架，时一个基于数据流编程的符号数学系统，被广泛应用于各类机器学习算法的编程实现，同时，TensorFlow 也是一个端到端开源机器学习平台。

1）安装方法

打开 Python 编辑器，在命令行处输入命令，如图 4 所示：

图 4.Tensorflow 安装
fig4.Tensorflow Installing

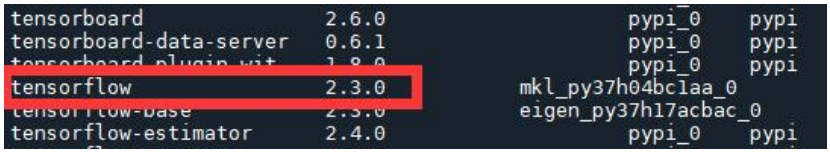

图 5.Tensorflow 安装成功
fig5.Tensorflow Installed

2）测试

在命令行处输入命令查看是否安装成功，如图 6 所示：

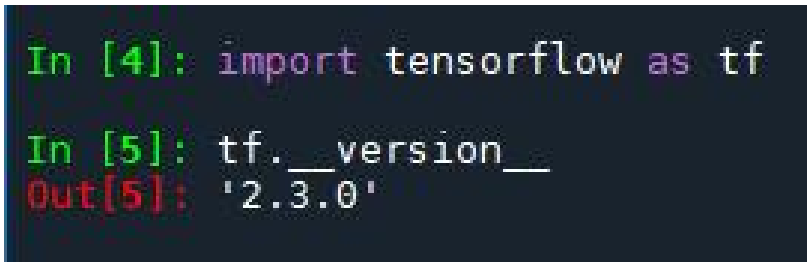

图 6.Tensorflow 安装测试
fig6.Tensorflow Installed Test

Tensorflow 拥有一个全面而灵活的生态系统，其中包含各种工具、库和社区资源，可助力研究人员推动先进机器学习技术的发展，并使开发者能够轻松地构建和部署由机器学习提供支持的应用。对于参加数学建模的同学来说很有必要掌握，利用好开源工具更好的实现对数据的挖掘和可视化。

4、例题与程序设计

1）以 2021 年高教社杯全国大学生数学建模竞赛题目为例，如图 7 所示，为题目详细内容：

2021 年高教社杯全国大学生数学建模竞赛题目

（请先阅读“全国大学生数学建模竞赛论文格式规范”）

B 题　乙醇偶合制备 C4 烯烃

C4 烯烃广泛应用于化工产品及医药的生产，乙醇是生产制备 C4 烯烃的原料。在制备过程中，催化剂组合（即：Co 负载量、Co/SiO2 和 HAP 装料比、乙醇浓度的组合）与温度对 C4 烯烃的选择性和 C4 烯烃收率将产生影响（名词解释见附录）。因此通过对催化剂组合设计，探索乙醇催化偶合制备 C4 烯烃的工艺条件具有非常重要的意义和价值。

某化工实验室针对不同催化剂在不同温度下做了一系列实验，结果如附件 1 和附件 2 所示。请通过数学建模完成下列问题：

(1) 对附件 1 中每种催化剂组合，分别研究乙醇转化率、C4 烯烃的选择性与温度的关系，并对附件 2 中 350 度时给定的催化剂组合在一次实验不同时间的测试结果进行分析。

(2) 探讨不同催化剂组合及温度对乙醇转化率以及 C4 烯烃选择性大小的影响。

(3) 如何选择催化剂组合与温度，使得在相同实验条件下 C4 烯烃收率尽可能高。若使温度低于 350 度，又如何选择催化剂组合与温度，使得 C4 烯烃收率尽可能高。

(4) 如果允许再增加 5 次实验，应如何设计，并给出详细理由。

图 7.2021 年数学建模 B 题

fig7.2021 year mathematical modeling of B

部分实验数据如表 1 所示：

表 1.部分实验数据

table1.Partial experimental data

催化剂组合编号	催化剂组合	温度	乙醇转化率(%)	乙烯选择性（%）
A1	200mg 1wt%Co/SiO2-200mg HAP-乙醇浓度1.68ml/min	250	2.07	1.17
		275	5.85	1.63
		300	14.97	3.02
		325	19.68	7.97
		350	36.80	12.46
A2	200mg 2wt%Co/SiO2-200mg HAP-乙醇浓度1.68ml/min	250	4.60	0.61
		275	17.20	0.51
		300	38.92	0.85
		325	56.38	1.43
		350	67.88	2.76
A3	200mg 1wt%Co/SiO2-200mg HAP-乙醇浓度0.9ml/min	250	9.7	0.13
		275	19.2	0.33
		300	29.3	0.71
		325	37.6	1.83
		350	48.9	2.85
		400	83.7	6.76
		450	86.4	14.84

该题提供了部分的实验数据，需要答题人利用数据挖掘的方法，将剩余残差的数据拟合出来，模型主要受温度和反应物的影响，根据不同反应物和催化剂的组合做对比实现，我们搭建 BP 神经网络对实验数据进行拟合。

本次实验的最终目的时为了预测出实验数据，将所有的实验数据进行非线性回归之后，完善所给的实验数据，对每一组实验数据进行 C4 烯烃收率的计算，求解出在温度高于或低于 350 度时催化剂与温度的最优组合得到 C4 烯烃收率的最大化。

2）程序设计

为了方便编程，对数据进行预处理，将数据存入数组，方便模型的训练以及调用，构建方式如图 8 所示：

```
data_imp[:, 4] = data[:, 0]
data_imp[0:5, 0] = 200
data_imp[0:5, 1] = 1
data_imp[0:5, 2] = 200
data_imp[0:5, 3] = 1.68
data_imp[5:10, 0] = 200
data_imp[5:10, 1] = 2
data_imp[5:10, 2] = 200
data_imp[5:10, 3] = 1.68
data_imp[10:17, 0] = 200
data_imp[10:17, 1] = 1
data_imp[10:17, 2] = 200
data_imp[10:17, 3] = 0.9
data_imp[17:23, 0] = 200
```

图 8.构建实验数据数组
fig8.Build Test Data Array

搭建 BP 神经网络函数，设定了一个输入层，一个隐藏层，一个输出层，Tensorflow 的函数以及 API 如图 9 所示：

```
def mod():
    model = tf.keras.Sequential()
    model.add(keras.layers.BatchNormalization())
    model.add(tf.keras.layers.Dense(units=150,
                                    activation=tf.keras.activations.relu))
    model.add(keras.layers.BatchNormalization())
    model.add(tf.keras.layers.Dense(units=100,
                                    activation=tf.keras.activations.relu))
    model.add(keras.layers.BatchNormalization())
    model.add(keras.layers.AlphaDropout(rate=0.5))
    model.add(tf.keras.layers.Dense(units=2,
                                    activation=tf.keras.activations.relu))
    return model
```

图 9.构建 BP 神经网络函数
fig9.Build BP Neural Network Function

导入模型，采用均方误差作为误差反向传播的修正函数，通过反向传播，不断的去修正神经网络的权值，得到期望值，如图 10 所示：

```
for i in range(1000):
    with tf.GradientTape(persistent=True) as tape:
        out = modd(data_imp)
        loss = tf.reduce_mean(tf.sqrt(tf.square(out - data_out) + 0.00001))  # 均方误差
        loss_regularization = []
    train_loss(loss)
    grad = tape.gradient(loss, modd.trainable_variables)
```

图 10.均方误差
fig10.MSE

训练模型，将实验数据依次喂入神经网络进行非线性拟合，并保存实验结果，如图 11 所示：

```
for i in range(x[0], x[1]):
    data_test[i, 0:4] = np.array(([200, 1, 200, 1.68]))
    data_test[i, 4] = i
```

图 11.训练 BP 神经网络函数
fig11.Trains BP Neural Network Function

设置迭代次数 1000 次，实验结果，如图 12 所示：

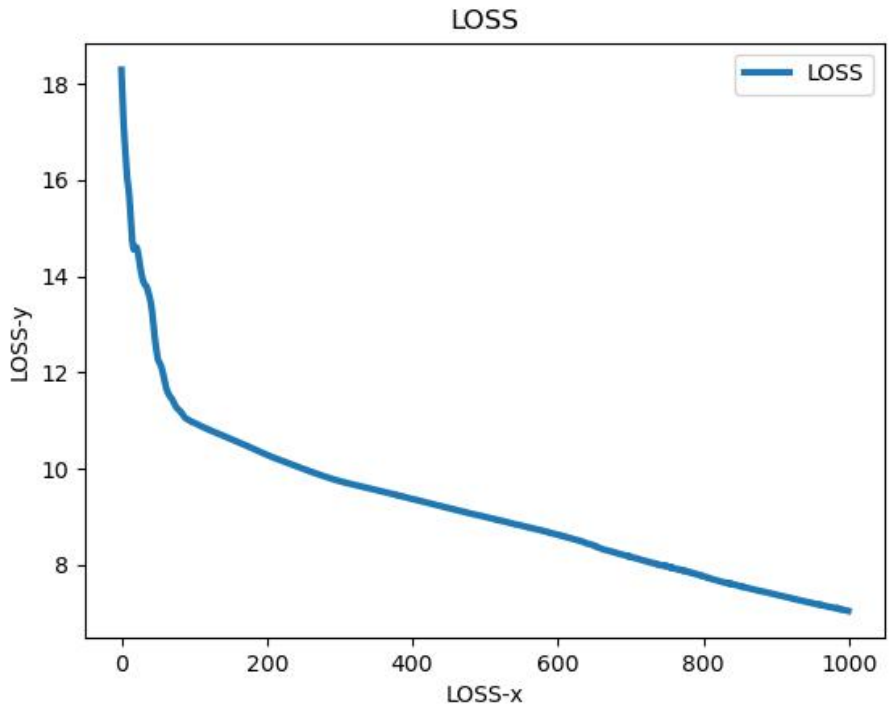

图 12.实验结果
fig12.Test Result

将 BP 拟合的实验结果与原始数据进行对比：

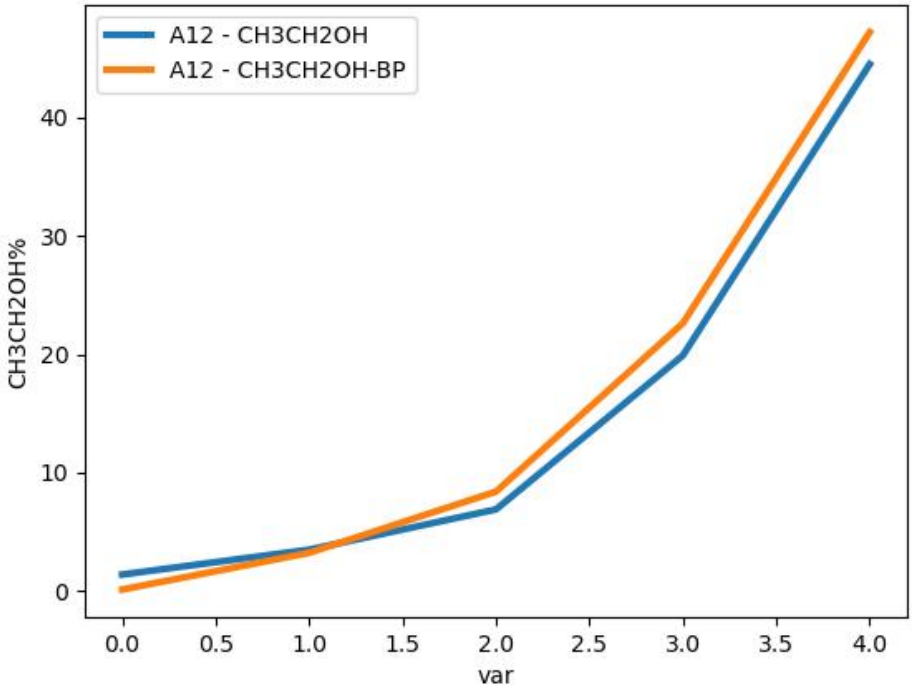

图 13.实验对比结果
fig13.Contrast Experiment

从图 13 中可以看到，当反应温度超过 100 摄氏度时，拟合值小于实际值，图 12 中误差逐渐下降，但因为数据量实在是过于少的原因，对拟合的数据的 loss 值无法继续下降，提供一种解决思路就是插值，或者对实验数据继续扩充，增强模型的泛化性。

并且，BP 神经网络要解决的是一个复杂非线性化问题，且收敛速度很慢对初始网络权重非常敏感，以不同的权重初始化网络，其往往会收敛于不同的局部极小，有可能每次训练得到不同结果，从而导致网络训练失败，需要合理的设置初始化参数，确保神经网络不会出现过拟合、梯度消失、梯度爆炸等问题。

3）函数调用说明

在本文中，提供了 BP 神经网络的代码实现方法，对于 B 题的解题代码，在表 2 中也一并给出，包含函数的说明及调用方法。

表 2.函数说明

table2.Function Description

Function	Description
bpnetword.py	BP 神经网络函数
createIMG.py	实验结果对比
kendall.py	kendall 系数的计算
relation.py	相关性分析
result3.py	求解最优值函数
Sklean.py	Sklearn 机器学习函数例程

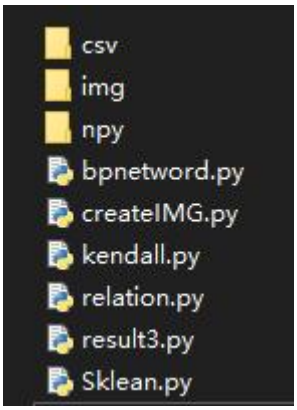

图 14.文件目录

fig14.File Path

csv 文件：实验数据图表；

img 文件：实验结果可视化图形存储；

npy 文件：实验结果数据存储。

直接运行代码即可得到实验结果，在对应的文件中查询实验数据与图表。

4）Tensorflow 常用函数说明

表 3.基础运算

table3.Basic operation

操作	描述
tf.add(x, y, name=None)	求和
tf.sub(x, y, name=None)	减法
tf.mul(x, y, name=None)	乘法
tf.div(x, y, name=None)	除法
tf.mod(x, y, name=None)	取模
tf.abs(x, name=None)	求绝对值
tf.neg(x, name=None)	取负 (y = -x).
tf.sign(x, name=None)	返回符号 y = sign(x) = -1 if x < 0; 0 if x == 0; 1 if x > 0.
tf.inv(x, name=None)	取反
tf.square(x, name=None)	计算平方 (y = x * x = x^2).
tf.round(x, name=None)	舍入最接近的整数 # ‘a’ is [0.9, 2.5, 2.3, -4.4] tf.round(a) ==> [1.0, 3.0, 2.0, -4.0]
tf.sqrt(x, name=None)	开根号 (y = \sqrt{x} = x^{1/2}).
tf.pow(x, y, name=None)	幂次方 # tensor ‘x’ is [[2, 2], [3, 3]] # tensor ‘y’ is [[8, 16], [2, 3]] tf.pow(x, y) ==> [[256, 65536], [9, 27]]
tf.exp(x, name=None)	计算 e 的次方
tf.log(x, name=None)	计算 log，一个输入计算 e 的 ln，两输入以第二输入为底
tf.maximum(x, y, name=None)	返回最大值 (x > y ? x : y)

操作	描述
tf.minimum(x, y, name=None)	返回最小值 (x < y ? x : y)
tf.cos(x, name=None)	三角函数 cosine
tf.sin(x, name=None)	三角函数 sine
tf.tan(x, name=None)	三角函数 tan
tf.atan(x, name=None)	三角函数 ctan

表 4.激活函数

table4.Activation Functions

操作	描述
tf.nn.relu(features, name=None)	整流函数：max(features, 0)
tf.nn.relu6(features, name=None)	以 6 为阈值的整流函数 min(max(features, 0), 6)
tf.nn.elu(features, name=None)	elu 函数，exp(features) - 1 if < 0,否则 features Exponential Linear Units (ELUs)
tf.nn.softplus(features, name=None)	计算 softplus：log(exp(features) + 1)
tf.nn.dropout(x, keep_prob, noise_shape=None, seed=None, name=None)	计算 dropout，keep_prob 为 keep 概率 noise_shape 为噪声的 shape
tf.nn.bias_add(value, bias, data_format=None, name=None)	对 value 加一偏置量 此函数为 tf.add 的特殊情况，bias 仅为一维， 函数通过广播机制进行与 value 求和, 数据格式可以与 value 不同，返回为与 value 相同格式
tf.sigmoid(x, name=None)	y = 1 / (1 + exp(-x))
tf.tanh(x, name=None)	双曲线切线激活函数

表 5.卷积函数

table5.Convolution Functions

操作	描述
tf.nn.conv2d(input, filter, strides, padding, use_cudnn_on_gpu=None, data_format=None, name=None)	在给定的 4D input 与 filter 计算 2D 卷积输入 shape 为 [batch, height, width, in_channels]
tf.nn.conv3d(input, filter, strides, padding, name=None)	在给定的 5D input 与 filter 计算 3D 卷积输入 shape 为[batch, in_depth, in_height, in_width, in_channels]

表 5.池化函数

table5.Pooling Functions

操作	描述
tf.nn.avg_pool(value, ksize, strides, padding, data_format='NHWC', name=None)	平均方式池化
tf.nn.max_pool(value, ksize, strides, padding, data_format='NHWC', name=None)	最大值方法池化
tf.nn.max_pool_with_argmax(input, ksize, strides, padding, Targmax=None, name=None)	返回一个二维元组(output,argmax),最大值 pooling，返回最大值及其相应的索引
tf.nn.avg_pool3d(input, ksize, strides, padding, name=None)	3D 平均值 pooling
tf.nn.max_pool3d(input, ksize, strides, padding, name=None)	3D 最大值 pooling