

神经网络实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.11 |

目录

[一、实验目的 1](#_Toc60250319)

[二、实验内容 1](#_Toc60250320)

[三、实验环境 1](#_Toc60250321)

[四、 实验步骤 2](#_Toc60250322)

[五、神经网络实验报告表 2](#_Toc60250323)

[六、编写简单的感知器学习程序 4](#_Toc60250324)

[七、实验结论 6](#_Toc60250325)

# 一、实验目的

理解反向传播网络的结构和原理，掌握反向传播算法对神经元的训练过程，了解反向传播公式。通过构建BP网络实例，熟悉前馈网络的原理及结构。

# 二、实验内容

1. 通过BP网络各项参数的不同设置，观察BP算法的学习效果。观察比较BP网络拓朴结构及其它各项参数变化对于训练结果的影响。观察并分析不同训练数据对相同拓朴结构的BP网络建模的影响。
2. 设计简单的感知器，实现简单的逻辑运算（与、或）等，也可做其他更复杂的问题。

# 三、实验环境

以下两种实验环境可供选择：

1. 神经网络可视化实验环境，如图3所示。

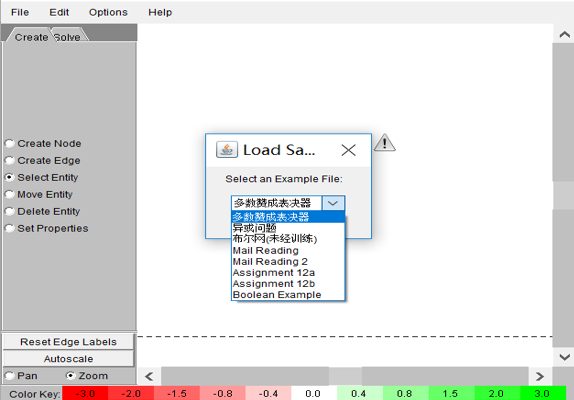


图 1 神经网络可视化实验环境

1. C＋＋语言编程环境，语言环境可以自选。

# 四、 实验步骤

1. 进入神经网络可视化实验环境，基本实验步骤如下：
   1. 进入实验环境；
   2. 选择相关的实验模块；
   3. 设置相应地实验参数（如设置初始权值为随机值）；
   4. 选择实验运行方式；
   5. 观测运行过程；
   6. 修改相应地参数重复第1~5步直到满意为止。

**对设计型实验可参照以下步骤：**

1. 进入实验环境；
2. 进入创建新模型工作窗；
3. 创建网络拓朴结构；
4. 设置相应的网络参数；
5. 输入相应的训练数据集；
6. 设置实验环境参数；
7. 选择相应的运行方式；
8. 观察实验过程
9. 修改相关参数并重复1~8步直至满意为止。
10. 编写简单的感知器学习程序，训练感知器执行操作。

# 五、神经网络实验报告表

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 王启乐 | 指导老师:钟萍 | 日期:2020.11.23 |
| 实验目的 | 理解反向传播网络的结构和原理，掌握反向传播算法对神经元的训练过程，了解反向传播公式。通过构建BP网络实例，熟悉前馈网络的原理及结构。 | | |
| 网络拓扑图 | 图 1.1 初始状态    图 1.2 目标状态 | 图 2.1 初始状态    图 2.2 目标状态 | 图 3.1 初始状态    图 3.2 目标状态 |
| 训练数据集 | T:节点 0,节点 1,节点 2,节点 5;  A:0.0,0.0,0.0,0.0;  A:0.0,0.0,1.0,0.0;  A:0.0,1.0,0.0,0.0;  A:0.0,1.0,1.0,1.0;  A:1.0,0.0,0.0,0.0;  A:1.0,0.0,1.0,1.0;  A:1.0,1.0,0.0,1.0;  A:1.0,1.0,1.0,1.0; | T:节点 0,节点 1,节点 4;  A:0.0,0.0,0.0;  A:0.0,1.0,1.0;  A:1.0,0.0,1.0;  A:1.0,1.0,0.0; | T:节点1,节点2,与,或,非  A:0.0,0.0,0.0,0.0,1.1;  A:0.0,1.1,0.0,1.1,0.0;  A:1.1,0.0,0.0,1.1,0.0;  A:1.1,1.1,1.1,1.1,1.1; |
| 训练误差 | 0.09875 | 0.09875 | 0.09996 |
| 模拟问题或函数 | 多数赞成表决器 | 异或问题 | 布尔网 |
| 观测结果 | 图 1.3 误差值表    图 1.4 训练前    图 1.5 训练后 | 图 2.3 误差值表    图 2.4 训练前    图 2.5 训练后 | 图 3.3 误差值表    图 3.4 训练前    图 3.5 训练后 |
| 学生结论 | 经过多次训练反馈修改权值，最终神经网络能够完成多数赞成表决任务。 | 训练之后准确性明显提高但是由于存在0.1的误差所以不能完全准确，赋的值不同，学习速度不同，到达目标的时间长短也不同，r越小，同等时间下所需训练的次数也就越多且近似成反比关系。 | 经过多次训练反馈修改权值，最终神经网络能够完成布尔网求解任务。此外，初始权值的设定会影响训练的过程和结果，不合适的权值会导致难以训练出结果。 |

# 六、编写简单的感知器学习程序

反向传播（英语：Backpropagation，缩写为BP）是“误差反向传播”的简称，是一种与最优化方法（如梯度下降法）结合使用的，用来训练人工神经网络的常见方法。该方法对网络中所有权重计算损失函数的梯度。这个梯度会反馈给最优化方法，用来更新权值以最小化损失函数。反向传播要求有对每个输入值想得到的已知输出，来计算损失函数梯度。因此，它通常被认为是一种监督式学习方法。反向传播要求人工神经元（或“节点”）的激励函数可微。

MNIST数据集的每一张图片是一个28x28的矩阵，每个元素是其位置的灰度信息。我们首先将二维的矩阵打平成一维的，也就是说变成一个784x1的矩阵。上面介绍的简单的例子是单个数字，这里成了矩阵，所以我们的参数也应该是矩阵。

假设我们先将784x1的矩阵乘以一个512x784的矩阵参数，将得到一个512x1的矩阵；再将512x1的矩阵乘以一个256x512的矩阵参数，将得到一个256x1的矩阵；再将256x1的矩阵乘以一个10x256的矩阵参数，将得到一个10x1的矩阵。经过上述操作，我们将一个784x1的矩阵转换成了一个10x1的矩阵。为什么最后要转换成10个元素的矩阵？因为我们的手写体识别问题中有10个数字，最后我们得到的矩阵的每个元素代表的是可能是相应数字的概率。根据得到的概率和实际情况来构造损失函数，再利用梯度下降的方法来更新参数。

上述的好几步的矩阵转换操作实际上就是神经网络中的层，最后一步的10个元素属于输出层，第一次的784x1是输入层，中间的则是隐藏层。

该部分完整代码如下：

1. #下载、导入数据用到的函数包
2. **import** input\_data
3. #tensorflow 2.x没有placeholder所以要用1.x的API
4. **import** tensorflow.compat.v1 as tf
5. #禁用tensorflow 2.x行为
6. tf.disable\_v2\_behavior()
7. #读取数据
8. mnist = input\_data.read\_data\_sets("MNIST\_data", one\_hot=True)
9. #placeholder用于定义过程，在执行的时候再赋具体的值
10. # 我们在TensorFlow运行计算时输入这个值。我们希望能够输入任意数量的MNIST图像，每一张图展平成784维的向量。
11. x = tf.placeholder("float", [None, 784])
12. #W的维度是[784，10]，因为我们想要用784维的图片向量乘以它以得到一个10维的证据值向量
13. W = tf.Variable(tf.zeros([784,10]))
14. #b的形状是[10]，所以我们可以直接把它加到输出上面
15. b = tf.Variable(tf.zeros([10]))
16. #softmax模型
17. y = tf.nn.softmax(tf.matmul(x,W) + b)
18. #为了计算交叉熵，我们首先需要添加一个新的占位符用于输入正确值
19. y\_ = tf.placeholder("float", [None,10])
20. #计算交叉熵
21. cross\_entropy = -tf.reduce\_sum(y\_\*tf.log(y))
22. #使用反向传播算法(backpropagation algorithm)进行训练
23. train\_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross\_entropy)
24. #初始化tensorflow
25. init = tf.initialize\_all\_variables()
26. sess = tf.Session()
27. sess.run(init)
28. #让模型循环训练1000次
29. **for** i **in** range(1000):
30. batch\_xs, batch\_ys = mnist.train.next\_batch(100)
31. sess.run(train\_step, feed\_dict={x: batch\_xs, y\_: batch\_ys})
32. correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1))
33. accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, "float"))
34. **print**(sess.run(accuracy, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels}))

# 七、实验结论

BP神经网络的原理并不难，可以自行手动实现，但使用TensorFlow不仅方便，而且有更高的效率。但Python的数值计算效率并不高，要实现高效的数值运算一般会使用Numpy这样的库，将运算放到外部其他语言封装好的程序中，以此来完成高效运算。但是从外部计算切换回Python的每一个操作，仍然是一个很大的开销。TensorFlow也把复杂的计算放在Python之外完成，但是为了避免前面说的那些开销，它做了进一步完善。Tensorflow不单独地运行单一的复杂计算，而是让我们可以先用图描述一系列可交互的计算操作，然后全部一起在Python之外运行。