



网络空间先进技术研究院

# 机器学习与人工智能

## 课程报告

项目名称 手写数字的分类预测

学 院 计算机科学与教育软件学院

专业班级 网研院

学生姓名 谢禹舜

学生学号 2111806060

任课教师 仇晶

2018 年 11 月

课程报告评定表

学生姓名	谢禹舜	学 号	2111806060	成绩	
专业班级	网研院				
项目名称	手写数字的分类预测				
指导教师评语	<div>指导教师： 年 月 日</div>				

## 目 录

1	项目目的 .....	1
2	项目环境和条件 .....	1
3	项目原理 .....	1
3.1	发展历程 .....	1
3.2	原理部分 .....	2
3.3	具体实现 .....	2
4	项目内容 .....	5
5	项目环境和条件 .....	5
5.1	任务分析 .....	5
5.2	数据分析 .....	5
5.3	开发步骤 .....	6
5.4	关键问题 .....	6
5.5	实验结果分析 .....	6
6	项目总结与心得体会 .....	7
7	参考文献 .....	8

## 1 项目目的

熟悉朴素神经网络算法及其在手写数字任务的应用，理解 sigmoid 函数、反向误差传播算法、梯度下降算法和全连接神经网络模型的相关知识，了解和使用 python 平台/开发环境。

## 2 项目环境和条件

软硬件环境和条件：

硬件： PC 机，处理器为英特尔酷睿 i5-3230M。

软件： python3 环境，需要用到的库有：scipy、matplotlib、numpy

## 3 项目原理

### 3.1 发展历程：

在人工神经网络的发展历史上，感知机(Multilayer Perceptron, MLP)网络曾对人工神经网络的发展发挥了极大的作用，也被认为是一种真正能够使用的人工神经网络模型，它的出现曾掀起人们研究人工神经网络的热潮。单层感知网络(M-P 模型)作为最初的神经网络，具有模型清晰、结构简单、计算量小等优点。但是，随着研究工作的深入，人们发现它的不足，例如无法处理非线性问题，即使计算单元的作用函数不用阀函数而用其他较复杂的非线性函数，仍然只能解决线性可分问题。不能实现某些基本功能，从而限制了它的应用。增强网络的分类和识别能力、解决非线性问题的唯一途径是采用多层前馈网络，即在输入层和输出层之间加上隐藏层，构成多层前馈感知器网络。

20 世纪 80 年代中期，David Rumelhart、Geoffrey Hinton 和 Ronald Williams、David Parker 等人分别独立发现了误差反向传播算法(Error Back Propagation Training)，简称 BP，系统解决了多层神经网络隐藏层链接权重学习问题，并在数学上给出了完整推导。人们将采用这种算法进行误差校正的多层前馈网络称为 BP 网。

BP 神经网络具有任意复杂的模式分类能力和优良的多维函数映射能力，解决了简单感知器不能解决的异或(Exclusive OR, XOR)和一些其他问题。从结构上讲，BP 网络具有输入层、隐藏层和输出层；从本质上讲，BP 算法就是以网络误差平方为目标函数、采用梯度下降法来计算目标函数的最小值。

### 3.2 原理部分：

人工神经网络无需事先确定输入与输出之间映射关系的数学方程，仅通过自身的训练，学习某种规则，在给定输入值的情况下得到最接近期望输出值的结果。作为一种智能信息处理系统，人工神经网络实现其功能的核心是算法。BP 神经网络是一种按误差反向传播(简称误差反传)训练的多层前馈网络，其算法称为 BP 算法，它的基本思想是梯度下降法，利用梯度搜索技术，以使网络的实际输出值和期望输出值的误差均方差为最小。

基本 BP 算法包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。误差反传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐藏节点的链接强度和隐藏节点与输出节点的链接强度以及阈值，使误差沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数(权值和阈值)，训练即告停止。此时经过训练的神经网络即能对类似样本的输入信息，自行处理输出误差最小的经过非线性转换的信息。

### 3.3 具体实现：

3.3.1 创建一个简单的三层神经网络，分别是输入层、隐藏层和输出层。本次课程数据是手写数字图片，手写数字图片（28×28）看成是一个 784×1 的列向量  $i$ ，所以输入层的节点数目是 784；隐藏层节点数目是一个超参数，需要根据实验结果进行调节，本次实验中设置为 200；根据手写数字的输入是 0~9 将输出层节点数目设置为 10。

3.3.2 神经元对于输入是有条件的，对于微小的信号不会立即反应，而是抑制输入，直到输入增强到可以触发输出。为了模拟神经元这一特性，使用 sigmoid 函数：

$$y = \frac{1}{1 + e^{-x}}$$

由于一个神经元可以接受许多输入，所以只需要将他们相加，作为 S 函数的输入，然后输出结果就可以了。

3.3.3 输入层与隐藏层、隐藏层与输出层之间使用全连接，任意两个结点  $i, j$  之

间的权重表示为 $w_{ij}$ 。根据输入层节点与隐藏层节点的数目，定义输入层与隐藏层之间的权重用一个  $200 \times 764$  的权重矩阵 $W_{input\_hidden}$ ；同理隐藏层与输出层之间的权重用一个  $10 \times 200$  的权重矩阵 $W_{hidden\_output}$ 表示。所以，当输入为一个列向量矩阵 $I$ 时：

$$X_{hidden} = W_{input\_hidden} \cdot I$$

$$O_{hidden} = \text{sigmoid}(X_{hidden})$$

$$X_{output} = W_{hidden\_output} \cdot O_{hidden}$$

所以，当一个节点接收到上一层所有节点的输入值时，该节点会将所有输入相加，然后经过激活函数处理后输出。

3.3.4 将神经网络输出值与真实值进行比较，使用反向误差传播进行权重更新。

由于误差值 $e_1 = (t_1 - o_1)$ ，各部分的权重为 $\frac{w_{1,1}}{w_{1,1}+w_{2,1}}$ 和 $\frac{w_{2,1}}{w_{1,1}+w_{2,1}}$ ，所以

$e_{hidden,1}$  = 链接 $w_{1,1}$ 和 $w_{2,1}$ 上的分割误差之和

$$= e_{output,1} * \frac{w_{1,1}}{w_{1,1} + w_{2,1}} + e_{output,2} * \frac{w_{2,1}}{w_{1,1} + w_{2,1}}$$

将上述过程进行矢量化：

$$e_{hidden,1} = e_{output,1} * \frac{w_{1,1}}{w_{1,1} + w_{2,1}} + e_{output,2} * \frac{w_{2,1}}{w_{1,1} + w_{2,1}}$$

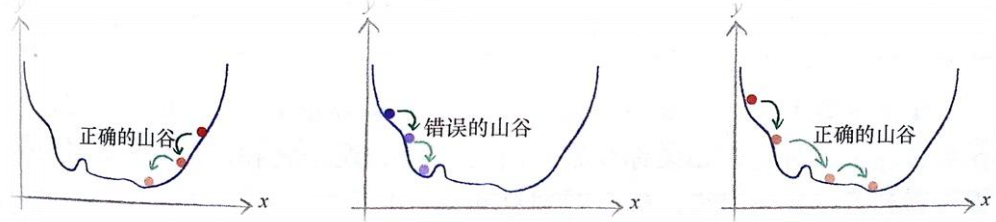
所以最后总结可得：

$$error_{hidden} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

$$error_{hidden} = W_{hidden\_output}^T \cdot error_{output}$$

3.3.5 为了调整权重使得误差最小，我们使用梯度下降求最值的方法。形象的说明一下，就好比处于山顶上的一个人需要下山，但是他只能看到周围有限范围内的情况。此时他先走到这个范围内的最低点，由于其位置的变化，他所目及的范围也跟着发生了变化。同理在目前范围内仍然走到最低点，就这样反复进行，直到走到山底。在这个过程中，我们把这个人每次移动的距离叫做学习率。学习率也是一个参数，设置过大，容易在山底来回震荡，而取不到最低值；设置过小，又会使得下山的速度太慢，影响了算法的收敛速度。由上可知，这是一个由局部最优逐渐达到全局最优的算法，

所以很容易陷入一个局部最优解。还是拿下山来的例子来做说明，因为这个人只能看到其周围有限的范围，那么当他走到一个半山腰的凹地时，他就会认为自己已经到达山底，从而作出错误的判断。因此，我们需要反复训练，以期求得最优的结果，如下图所示：



在本次课程中，因为神经网络的输出真实值和理想值之间存在差值，所以可以看成误差函数，用数学表示为：

$$\frac{\partial E}{\partial w_{j,k}}$$

将误差函数展开，就是针对所有  $n$  个输出节点的和：

$$\frac{\partial E}{\partial w_{j,k}} = \frac{\partial}{\partial w_{j,k}} \sum_n (t_n - o_n)^2$$

但是误差函数根本就不需要对所有输出节点求和，因为节点的输出只取决于所连接的链接：

$$\begin{aligned} \frac{\partial E}{\partial w_{j,k}} &= \frac{\partial}{\partial w_{j,k}} (t_n - o_n)^2 = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{j,k}} \\ &= -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{j,k}} \\ &= -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{j,k}} \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) \\ &= -2(t_k - o_k) \cdot \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) \cdot \\ &\quad (1 - \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)) \frac{\partial}{\partial w_{j,k}} (\sum_j w_{j,k} \cdot o_j) \\ &= -2(t_k - o_k) \cdot \text{sigmoid}(\sum_j w_{j,k} \cdot o_j) \cdot \\ &\quad (1 - \text{sigmoid}(\sum_j w_{j,k} \cdot o_j)) \cdot o_j \end{aligned}$$

因为权重改变方向和梯度方向相反，使用学习因子后：

$$\text{new } w_{j,k} = \text{old } w_{j,k} - \alpha \cdot \frac{\partial E}{\partial w_{j,k}}$$

最后转换为矩阵形式，如下：

$$\begin{aligned}\Delta w_{j,k} &= \alpha \cdot E_k * S_k \cdot (1 - S_k) \cdot O_j \\ &= \alpha \cdot E_k * O_k \cdot (1 - O_k) \cdot O_j^T\end{aligned}$$

3.3.6 神经网络的输出是一个 10 维的列向量，每一维的大小表示的是该维度对应数字的可能性大小，我们总是取概率值最大的那一维对应的数字作为神经网络对输入的手写数字的分类。

## 4 项目内容

- (1) 了解神经网络如何工作，并且制作出自己的神经网络，在构造神经网络模型中，使用反向传播算法来更新权重，在实现更新权重的过程中使用梯度下降法进行求解。
- (2) 本次课程任务是手写数字的自动分类，根据网上的数据集训练出一个神经网络的模型，使用反向传播算法和梯度下降法对神经网络模型进行更新修正，以提高分类的正确率。
- (3) 本次课程任务主要使用 Python 语言，应用到 numpy、scipy、matplotlib 库。本次使用 numpy 库是将运算转化为矩阵运算，这样运算会更加高效，并且节省内存；使用 scipy 库是为了构造出 sigmoid 函数用来模拟神经元的工作原理；使用 matplotlib 库是为了最后会用图表更加直观显示正确率。

## 5 项目过程与内容（例如：包括任务分析、数据分析、开发步骤、关键问题、实验结果分析等）

### 5.1 任务分析

输入一张手写数字的图片，通过训练得到的模型，将手写数字图片根据数字正确分类

### 5.2 数据分析

5.2.1 使用网上的 MINIST 数据集，使用数据集来训练模型，使用测试集来验证模型。

5.2.2 MINIST 数据集对于每张手写数字图片都转换为一个 785 长度的列表，列表第



一位是数字，可作为标签，后 784 位是图片的灰度值。

5.2.3 输出十个类别，依次对应 0~9 数字。

## 5.3 开发步骤

5.3.1 构造神经网络框架：初始化函数、训练模型和查询数据。

5.3.2 初始化函数中设置输入层节点、隐藏层节点和输出层节点的数量用来规定神经网络的形状和尺寸；训练模型中创建网络的节点和链接，用于之后最重要的链接权重和更新权重；查询数据中接受神经网络的输入，返回网络的输出。

5.3.3 读取 MNIST 数据集，根据数据集和课程任务需求，设置输入层节点为 784，隐藏层节点为 200，输出层节点为 10。读取相对较小的 MNIST 训练数据集训练更新神经网络并进行验证。

5.3.4 使用完整数据集进行训练和测试。

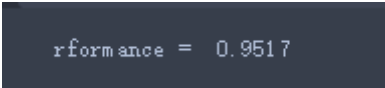
## 5.4 关键问题

5.4.1 隐藏层节点数目并不是通过科学方法得到的。在验证神经网络模型时多次更改隐藏层节点数目，使用输出正确率最高所对应的隐藏层节点数目。

5.4.2 输入层、隐藏层和输出层之间的矩阵更新修正。根据 MNIST 训练集，使用反向传播算法和梯度下降法对链接权重不断的训练以达到最优。

5.4.3 MNIST 数据集过大，不方便验证程序正确性。先使用相对较小的 MNIST 数据集进行验证，验证成功后再使用完整的 MNIST 数据集进行训练和测试。

## 5.5 实验结果分析



```
rformance = 0.9517
```

如上图所示，使用 60000 个训练样本训练简单的 3 层神经网络，然后使用 10000 条记录对网络进行测试，能达到 95%以上的正确率。使用神经网络得到的结果相对于其他模型要略胜一筹。

## 6 项目总结与心得体会

在本次课程实验中，对于神经网络的学习是一个从无到有的过程，不仅仅是学会了如何使用神经网络，还将神经网络的发展以及里面涉及到的知识点全部学习了一遍。

首先是所有有用的计算机系统都有一个输入和一个输出，并在输入和输出之间进行某种类型的计算。神经网络也是如此。当我们不能精确知道一些事情如何运作时，我们可以尝试使用模型来估计其运作方式，在模型中，包括了我们可以调整的参数。如果我们不知道如何将千米转换为英里，那么我们可以使用线性函数作为模型，并使用可调节的梯度值作为参数。改进这些模型的一种好方法是，基于模型和已知真实示例之间的比较，得到模型偏移的误差值，调整参数。我们使用简单的数学，理解了线性分类器输出误差值和可调节斜率参数之间的关系。也就是说，我们知道了在何种程度上调整斜率，可以消除输出误差值。但是使用朴素的调整方法会出现一个问题，即改进后的模型只与最后一次训练样本最匹配，“有效地”忽略了所有以前的训练样本。解决这个问题的一种好方法是使用学习率，调节改进速率，这样单一的训练样本就不能主导整个学习过程。来自真实世界的训练样本可能充满噪声或包含错误。适度更新有助于限制这些错误样本的影响。如果数据本身不是由单一线性过程支配，那么一个简单的线性分类器不能对数据进行划分。例如，由逻辑 XOR 运算符支配的数据说明了这一点。但是解决方案很容易，你只需要使用多个线性分类器来划分由单一直线无法分离的数据即可。

虽然比起现代计算机，生物大脑看起来存储空间少得多，运行速度比较慢，但是生物大脑却可以执行复杂的任务，如飞行、寻找食物、学习语言和逃避天敌。相比于传统的计算机系统，生物大脑对损坏和不完善信号具有难以置信的弹性。由互相连接的神经元组成的生物大脑是人工神经网络的灵感来源。通过神经网络向前馈送信号所需的大量运算可以表示为矩阵乘法。不管神经网络的规模如何，将输入输出表达为矩阵乘法，使得我们可以更简洁地进行书写。更重要的是，一些计算机编程语言理解矩阵计算，并认识到潜在的计算方法的相似性。这允许计算机高速高效地进行这些计算。

神经网络通过调整链接权重进行学习。这种方法由误差引导，误差就是训练数据所给出正确答案和实际输出之间的差值。简单地说，在输出节点处的误差等于所需值与实际值之间的差值。然而，与内部节点相关联的误差并不显而易见。一种方法是按照链接权重的比例来分割输出层的误差然后在每个内部节点处重组这些误差。反向传播误差可以表示为矩阵乘法。无论网络规模大小，这使我们能够简洁地表达反向传播误差，同时也允许理解矩阵计算的计算机语言更高效、更快速地完成工作。这意味着前向馈送信号和反向传播

误差都可以使用矩阵计算而变得高效。梯度下降法是求解函数最小值的一种很好的办法，当函数非常复杂困难，并且不能轻易使用数学代数求解函数时，这种方法却发挥了很好的作用。更重要的是，当函数有很多参数，一些其他方法不切实际，或者会得出错误答案，这种方法依然可以适用。这种方法也具有弹性，可以容忍不完善的数据，如果我们不能完美地描述函数，或我们偶尔意外地走错了一步，也不会错得离谱。神经网络的误差是内部链接权重的函数。改进神经网络，意味着通过改变权重减少这种误差。直接选择合适的权重太难了。另一种方法是，通过误差函数的梯度下降，采取小步长，迭代地改进权重。所迈出的每一步的方向都是在当前位置向下斜率最大的方向，这就是所谓的梯度下降。使用微积分可以很容易地计算出误差斜率。

如果输入、输出和初始权重数据的准备与网络设计和实际求解的问题不匹配，那么神经网络并不能很好地工作。一个常见的问题是饱和在这个时候，大信号(这有时候是由大权重带来的)导致了应用在信号上的激活函数的斜率变得非常平缓。这降低了神经网络学习到更好权重的能力。另一个问题是零值信号或零值权重。这也可以使网络丧失学习更好权重的能力。内部链接的权重应该是随机的，值较小，但要避免零值。如果节点的传入链接较多，有一些人会使用相对复杂的规则，如减小这些权重的大小。输入应该调整到较小值，但不能为零。一个常见的范围为  $0.01 \sim 0.99$ ，或  $-1.0 \sim 1.0$ ，使用哪个范围，取决于是否匹配了问题。输出应该在激活函数能够生成的值的范围内。逻辑 S 函数是不可能生成小于等于 0 或大于等于 1 的值。将训练目标值设置在有效的范围之外，将会驱使产生越来越大的权重，导致网络饱和。所以一个合适的范围为  $0.01 \sim 0.99$ 。

## 7 参考文献

- [1] Tariq Rashid, 林赐.Python 神经网络编程.第 1 版.北京: 人民邮电出版社, 2018