人工智能实验报告
BP学习算法

小组　lgd-cs
赖启东*　14347058
黎丁嘉　15336077
李振宇　15336092
刘键涵　15336110
洪培衔　15336059
2015级计算机科学与技术
School of Data and Computer Science, Sun Yat-Sen University, China
laiqd@mail2.sysu.edu.cn

January 11, 2018

Abstract

本次实验我们采用BP(Error Back Propagation Trainning, BP)学习算法解决手写数字识别问题。我们在基础的三层神经网络结构上加入并行计算的优化，对mnist数据集进行训练和测试，识别成功率高达92%以上。同时我们还进行现场手写图片识别，准确度高。

1 Introduction

1.1 Problem Description

使用BP学习算法进行手写数字识别。对数据集MNIST进行训练和测试，然后给含有一个数字的图片，识别出图片上数字。

1.2 Algorithm Review

误差反向传播算法(Error Back Propagation Trainning, BP)是1986年提出的概念[1]，是一种按照误差逆向传播训练的多层次前馈神经网络。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层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，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形转换的信息。

2 Algorithm Design

2.1 定义

我们采用三层神经元结构，分为输入层，隐层和输出层。输入层神经元个数为n个，隐层神经元个数m，输出层l个。我们采用单极性sigmoid函数

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

其导数为

$$f'(x) = \frac{e^x}{(1-e^{-x})^2} = f(x)(1-f(x)) \tag{2}$$

输入向量

$$\vec{X} = (x_1, x_2, \ldots, x_n)^T \tag{3}$$

隐层输出向量

$$\vec{Y} = (y_1, y_2, \ldots, y_m)^T \tag{4}$$

输出层输出向量

$$\vec{O} = (o_1, o_2, \ldots, o_l)^T \tag{5}$$

期望输出向量

$$\vec{D} = (d_1, d_2, \ldots, d_l)^T \tag{6}$$

输入层到隐层之间权值矩阵V和隐层到输出层权值矩阵W。

我们把输入的像素矩阵有笔迹的像素点定为1,没有笔迹的定为0,然后把二维展开成一维，作为输入向量。输出d_i表示为数字i的概率，最后取最大概率的作为识别出来的数字。

2.2 推导过程

总误差为

$$E = \frac{1}{2}\sum_{k=1}^{l}(d_k - o_k)^2 = \frac{1}{2}\sum_{k=1}^{l}(d_k - f(\sum_{j=1}^{m} w_{j,k} f(\sum_{i=1}^{n} V_{i,j} x_i))) \tag{7}$$

设

$$net_j = \sum_{i=1}^{n} v_{i,j} x_i \quad net_k = \sum_{j=1}^{m} w_{j,k} y_j \tag{8}$$

则

$$y_j = f(net_j) \quad o_k = f(net_k) \tag{9}$$

由式子(2)可以得到

$$\frac{\mathrm{d}o_k}{\mathrm{d}net_k} = o_k(1-o_k) \quad \frac{\mathrm{d}y_j}{\mathrm{d}net_j} = y_j(1-y_j) \tag{10}$$

令

$$\Delta w_{j,k} = -\eta\frac{\partial E}{\partial w_{j,k}} \quad \Delta v_{i,j} = -\eta\frac{\partial E}{\partial v_{i,j}} \tag{11}$$

$$\delta_k^o = -\frac{\partial E}{\partial net_k} \quad \delta_j^y = -\frac{\partial E}{\partial net_j} \tag{12}$$

可得

$$\delta_k^o = -\frac{\partial E}{\partial o_k}\frac{\mathrm{d}o_k}{\mathrm{d}net_k} = -(o_k - d_k)o_k(1-o_k) \tag{13}$$

$$\delta_j^y = -\sum_{k=1}^{l}\frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial y_j}\frac{\mathrm{d}y_j}{\mathrm{d}net_j} = \sum_{k=1}^{l}\delta_k^o w_{j,k} y_j(1-y_j) \tag{14}$$

权值调整

$$\Delta w_{j,k} = -\eta\frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{j,k}} = \eta\delta_k^o y_j = -\eta(o_k - d_k)o_k(1-o_k)y_j \tag{15}$$

$$\Delta v_{i,j} = -\eta\frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial v_{i,j}} = \eta\delta_j^y x_i = y_j(1-y_j)x_i\sum_{k=1}^{l}\delta_k^o w_{j,k} \tag{16}$$

2.3 算法设计

第$t+1$次迭代我们让

$$\delta_k^o(t+1) = -\frac{\partial E(t+1)}{\partial net_k(t+1)} + \mu\delta_k^o(t) \tag{17}$$

$$\delta_j^y(t+1) = -\frac{\partial E(t+1)}{\partial net_j(t+1)} + \mu\delta_j^y(t) \tag{18}$$

训练算法框图如下

Algorithm 1 train

给W, V矩阵每个元素赋$[-0.5, 0.5]$的随机值
repeat
  $i \leftarrow 0$
  repeat
    前向传播计算$\vec{Y}$和$\vec{O}$
    计算误差E
    计算δ_k^o和δ_j^y
    计算ΔW和ΔV并误差反转改变网络权值
    $i \leftarrow i+1$
  until $i > iteration$
until 训练完训练集

框图中的$iteration$是一个数据的训练次数，是参数。
测试函数如下

Algorithm 2 test(int** image)

1: 把二维数组image均匀分块，每块代表一个像素点
2: 统计每块中有笔迹的个数，确定每块是0还是1
3: 把每块信息传入输入层
4: 前向传播，得出$\vec{O}$
5: $ans \leftarrow$ 输出层$\vec{O}$中最大值的下标
6: **return** ans

2.4 优化

可以看到一次前向传播时间复杂度为$O(nm+ml)$，一次误差反传时间复杂度为$O(nm+ml)$，那么训练一次为$O(nm+ml)$,如果训练上万次程序运行时间非常久。于是我们考虑加入优化。由式子(7),(15),(16),(17),(18)可以看到计算实际上是做简单的高度相似的重复性工作。于是我们考虑使用**向量计算**来加速程序。实验中加入向量计算后速度显著增加。原本训练时间需要30分钟以上，加入并行计算后不到4分钟就可以训练完成，速度快了8倍。

3 Computational Experiments

3.1 Parameter Settings

输入层神经元个数为$784(=28*28)$，隐层神经元个数100,输出层神经元个数10。$\eta = 0.35$,$\mu = 0.05$。一个数据训练次数$iteration = 1$。

3.2 Datasets

选取的训练集与测试集为mnist,训练集规模为60000组，测试集规模10000。均是28*28的矩阵表示一个图片。此外我们还进行了现场人工手写数字图片识别。

3.3 Environment

我们的程序使用python3.62编写，实验运行在Intel Core i5-4200H 2.8GHz的CPU，4GB RAM，Windows 10操作系统的个人电脑上。

3.4 Results

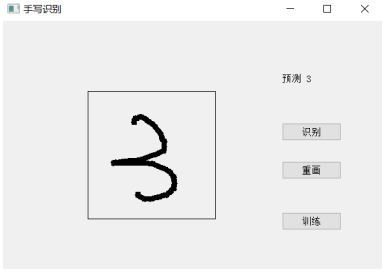
在mnist测试集上进行测试，可以看到识别成功率达到92.57%。

Figure 1: 图1

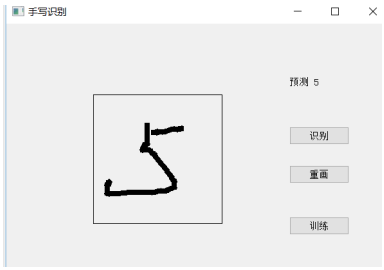
```
8134    8800
8233    8900
8333    9000
8421    9100
8519    9200
8618    9300
8716    9400
8809    9500
8904    9600
8999    9700
9079    9800
9166    9900
9257    10000
success_rate is  0.9257
```

进行手写识别实验截图如下

Figure 2: 图2



(a) 数字3



(b) 数字5

python任务台输出如下

Figure 3: 图3

```
we get  2
we get  3
we get  4
we get  5
we get  6
we get  8
we get  7
we get  4
we get  9
we get  6
we get  9
we get  0
we get  1
```

4 Conclusion

通过这次实验，我们学习人工神经网络的基本思路，了解BP学习算法思路，认真推导梯度函数，并将该算法应用到手写数字识别问题上。BP的效果神奇，在以后学习中期待进一步探究人工神经网络其他模型。

5 实验分工

赖启东负责核心算法实现
李振宇手写UI
黎丁嘉手写UI
洪培衔负责核心算法实现，并行优化，梯度函数推导
刘键涵调试修改程序
实验报告由赖启东撰写，平时实验过程记录报告由刘键涵完成。

References

[1] 朱福喜, "人工智能基础教程(第二版)," 2011.