深 圳 大 学 实 验 报 告

课程名称 机器学习

项目名称 实验六：深度神经网络

学　　院 计算机与软件学院

专　　业 软件工程

指导教师 赖志辉

报 告 人 郑杨 学号 2020151002

实验时间 2022 年 6 月 14 日至 2022 年 6 月 16 日

实验报告提交时间 2022 年 6 月 17 日

教务处制

目录

一、实验目的与要求

1、全方面比较 CNN 与全连接网络的异同。
2、推导神经网络前向后向传播算法的优化迭代公式。
3、熟练掌握一种深度神经网络的算法与应用，并给出在人脸识别、身份证识别、通用手写体识别等方面的 2 个以上应用案例与效果。

二、实验内容与方法

1、查阅相关资料，叙述 CNN 与全连接网络的异同。
2、参考西瓜书与网上博客资料，详细推导优化神经网络前后向传播算法。
3、使用神经网络进行通用手写体识别与人脸识别实验。

三、实验步骤与过程

1 CNN 与全连接网络的异同

相同点：

1. 都涉及了前向/后向传播算法
2. 都可以存在多层结构，如 CNN 具有输入层/卷积层/激励层/池化层/全连接层，全连接神经网络具有输入层/隐藏层/输出层
3. 都使用了激活函数，常用的有 sigmoid，softmax 等

不同点：

1. CNN 实际上内含了全连接神经网络
2. CNN 多层结构能够多元组合，即使是每一层也有很多可选的设定
3. CNN 参数通常比全连接神经网络多，但优化的效率会比后者更快，因为 CNN 有池化层显著降低数据量
4. CNN 能够有效进行特征提取，并且是自发性学习特征
5. CNN 需要较多的调参
6. 全连接神经网络常常会陷入局部最优解，且可能存在梯度消失等问题

2 本文涉及的符号说明

L 神经网络的层数

n_l 表示第 l 层的神经元个数

$f(\cdot)$ 表示神经元的激活函数

$W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ 表示第 $l-1$ 层到第 l 层的权重矩阵

$w_{ij}^{(l)}$ 是权重矩阵 $W^{(l)}$ 中的元素，表示第 $l-1$ 第 j 个神经元到第 l 层第 i 个神经元的权重

$b^{(l)}=(b_1^{(l)},b_2^{(l)},\cdots,b_{n_l}^{(l)})^T\in\mathbb{R}^{n_l}$ 表示$l-1$层到第l层的偏置

$z^{(l)}=(z_1^{(l)},z_2^{(l)},\cdots,z_{n_l}^{(l)})^T\in\mathbb{R}^{n_l}$ 表示第l层神经元的状态

$a^{(l)}=(a_1^{(l)},a_2^{(l)},\cdots,a_{n_l}^{(l)})^T\in\mathbb{R}^{n_l}$ 表示第l层神经元的激活值（$a^{(l)}=f(z^{(l)})$）

3 神经网络基础

一个神经网络模型由神经元、神经元之间的联系与激活函数$f(\cdot)$组成，感知机模型其实就是一个简单的神经网络模型，如下图所示：

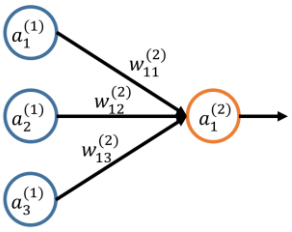

图 1：感知机模型

其计算输出值的方法是：

$$a_1^{(2)}=w_{11}^{(2)}a_1^{(1)}+w_{12}^{(2)}a_2^{(1)}+w_{13}^{(2)}a_3^{(1)}+b_1^{(2)}$$

如果对感知机模型进行推广，加入隐藏层，则会得到如图 2 所示的神经网络模型。该模型每一层的计算方法与感知机模型类似，具体前向传播算法在下一节进行叙述。

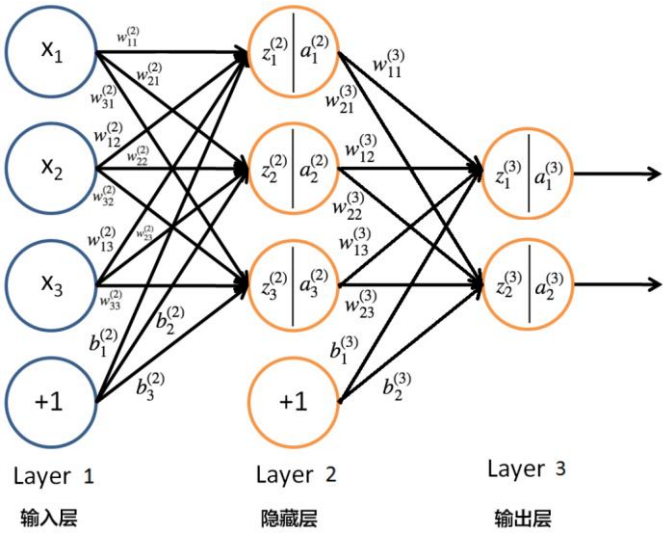

图 2：简单的神经网络模型

4 前向传播算法优化推导

以图 2 所示的简单模型为例，推导前向传播算法。

在已知每一层的权重矩阵$W^{(l)}$与偏置值向量$b^{(l)}$之后，显然，第 2 层第 1 个神经元的状

态与激活值可以通过以下计算得到：

$$z_{1}^{(2)} = w_{11}^{(2)}a_1^{(1)} + w_{12}^{(2)}a_2^{(1)} + w_{13}^{(2)}a_3^{(1)} + b_1^{(2)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

第 2 层的其他神经元与第 3 层的所有神经元激活值都可以通过类似的方法计算得到。写成矩阵与向量运算形式就是：

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$
$$a^{(2)} = f(z^{(2)})$$

故推广以下可以得到，第 $l(2 \le l \le L)$ 层神经元的状态及激活值为：

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = f(z^{(l)})$$

故，神经网络前向传播算法的迭代过程如下：

$$x = a^{(1)} \to z^{(2)} \to \cdots \to a^{(L-1)} \to z^{(L)} \to a^{(L)} = y$$

5 后向传播算法优化推导

5.1 后向传播算法的优化问题

前向传播算法讲的是已知每一个神经元的参数后，得到神经网络的输出，但事实上，每一个神经元的参数通常没法事先确定，所以需要一个后向传播算法优化加速每一个神经元参数的求解过程。

假设训练数据为 $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(N)}, y^{(N)})\}$，又假设输出数据为 n_L 维的，即 $y^{(i)} = (y_1^{(i)}, \cdots, y_{n_L}^{(i)})^T$，则对于某一个训练数据 $(x^{(i)}, y^{(i)})$ 来说，其代价函数可写为：

$$E_i = \frac{1}{2} \| y^{(i)} - o^{(i)} \| = \frac{1}{2} \sum_{j=1}^{n_L} (y_j^{(i)} - o_j^{(i)})^2$$

其中，$o^{(i)}$ 为神经网络对输入 $x^{(i)}$ 产生的实际的输出。代价函数采用均方误差表示，其中的 $\frac{1}{2}$ 是非必要的，只是为了后续计算时比较方便。以图 2 所示神经网络为例，$n_L = 2$，$y^{(i)} = (y_1^{(i)}, y_2^{(i)})^T$，从而有 $E_i = \frac{1}{2}(y_1^{(i)} - a_1^{(3)})^2 + \frac{1}{2}(y_2^{(i)} - a_2^{(3)})^2$；可以将这个误差展开到隐藏层，得到：

$$E_i = \frac{1}{2}(y_1^{(i)} - f(w_{11}^{(3)}a_1^{(2)} + w_{12}^{(3)}a_2^{(2)} + w_{13}^{(3)}a_3^{(2)} + b_1^{(3)}))^2$$
$$+ \frac{1}{2}(y_2^{(i)} - f(w_{21}^{(3)}a_1^{(2)} + w_{22}^{(3)}a_2^{(2)} + w_{23}^{(3)}a_3^{(2)} + b_2^{(3)}))^2$$

进一步，可以展开到输入层（替换掉$a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$即可），故代价函数$E_i$仅和权重矩阵$W^{(l)}$和偏置向量$b^{(l)}$相关，调整权重和偏置可以减小或增大神经网络的代价。

故，所有训练样本的平均代价可以写为：

$$E_{total} = \frac{1}{N}\sum_{i=1}^{N} E_i$$

于是，我们的目标就是求出使得上面这个代价函数最小的调整权重和偏置，可以使用一个最简单的方法——梯度下降（批量梯度下降）进行优化求解。由于我们求解的参数为$W^{(l)}$和$b^{(l)}$，$2 \le l \le L$，故可以使用下面的公式更新参数$W^{(l)}$和$b^{(l)}$：

$$W^{(l)} = W^{(l)} - \mu \frac{\partial E_{total}}{\partial W^{(l)}}$$
$$= W^{(l)} - \frac{\mu}{N}\sum_{i=1}^{N} \frac{\partial E_i}{\partial W^{(l)}}$$
$$b^{(l)} = b^{(l)} - \mu \frac{\partial E_{total}}{\partial b^{(l)}}$$
$$= b^{(l)} - \frac{\mu}{N}\sum_{i=1}^{N} \frac{\partial E_i}{\partial b^{(l)}}$$

故，只需要求得每一个训练数据的代价函数对参数$W^{(l)}$和$b^{(l)}$的偏导数$\frac{\partial E_i}{\partial W^{(l)}}$和$\frac{\partial E_i}{\partial b^{(l)}}$即可使用梯度下降法更新参数以优化原问题。下文将详细推导偏导数的求解方法，由于样本与样本之间的代价相互独立，于是我们只需要分别计算每一个E_i对参数的偏导数即可，下面把E_i记为E进行推导。

5.2 输出层的权重参数$W^{(L)}$更新

仍以图 2 作为例子进行推导，再进行推广得到公式。首先把E展开到隐藏层有：

$$
\begin{aligned}
E &= \frac{1}{2}\| y-o \| \\
&= \frac{1}{2}\| y-a^{(3)} \| \\
&= \frac{1}{2}((y_1-a_1^{(3)})^2+(y_2-a_2^{(3)})^2) \\
&= \frac{1}{2}((y_1-f(z_1^{(3)}))^2+(y_2-f(z_2^{(3)}))^2) \\
&= \frac{1}{2}((y_1-f(w_{11}^{(3)}a_1^{(2)}+w_{12}^{(3)}a_2^{(2)}+w_{13}^{(3)}a_3^{(2)}+b_1^{(3)}))^2+ \\
&(y_2-f(w_{21}^{(3)}a_1^{(2)}+w_{22}^{(3)}a_2^{(2)}+w_{23}^{(3)}a_3^{(2)}+b_2^{(3)}))^2)
\end{aligned}
$$

由求导的链式法则，我们对输出层的第 1 个神经元与隐藏层的第一个神经元的权重系数 $w_{11}^{(3)}$ 求偏导得：

$$
\begin{aligned}
\frac{\partial E}{\partial w_{11}^{(3)}} &= \frac{1}{2}2(y_1-a_1^{(3)})(-\frac{\partial a_1^{(3)}}{\partial w_{11}^{(3)}}) \\
&= -(y_1-a_1^{(3)})f'(z_1^{(3)})\frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}} \\
&= -(y_1-a_1^{(3)})f'(z_1^{(3)})a_1^{(2)} \\
&= \frac{\partial E}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}}\frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}}
\end{aligned}
$$

记 $\delta_1^{(3)} = \frac{\partial E}{\partial a_1^{(3)}}\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = \frac{\partial E}{\partial z_1^{(3)}} = -(y_1-a_1^{(3)})f'(z_1^{(3)})$，则上式可写为：

$$
\frac{\partial E}{\partial w_{11}^{(3)}} = \delta_1^{(3)}\frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}} = \delta_1^{(3)}a_1^{(2)}
$$

则用类似的方法可以求得输出层的其他权重参数。

而推广到一般情况有，

$$
\begin{aligned}
\delta_i^{(L)} &= -(y_i-a_i^{(L)})f'(z_i^{(L)}), 1\le i\le n_L \\
\frac{\partial E}{\partial w_{ij}^{(L)}} &= \delta_i^{(L)}a_j^{(L-1)}, 1\le i\le n_L, 1\le j\le n_{L-1}
\end{aligned}
$$

表示为矩阵（向量）形式为：

$$
\begin{aligned}
\delta^{(L)} &= -(y-a^{(L)})\odot f'(z^{(L)}), 1\le i\le n_L \\
\frac{\partial E}{\partial W^{(L)}} &= \delta^{(L)}a^{(L-1)}, 1\le i\le n_L, 1\le j\le n_{L-1}
\end{aligned}
$$

其中，$\odot$ 表示 Hadamard product 运算，规则是把矩阵（向量）的对应位置相乘。

5.3 隐藏层的权重参数$W^{(l)}(2\le l\le L-1)$更新

对隐藏层的权重参数求偏导，利用$\delta_i^{(l)}$的定义，有：

$$\frac{\partial E}{\partial w_{ij}^{(l)}}=\frac{\partial E}{\partial z_i^{(l)}}\frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}=\delta_i^{(l)}a_j^{(l-1)}$$

其中，$\delta_i^{(l)}=\frac{\partial E}{\partial z_i^{(l)}}=\sum_{j=1}^{n_{l+1}}\frac{\partial E}{\partial z_j^{(l+1)}}\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}=\sum_{j=1}^{n_{l+1}}\delta_j^{(l+1)}\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}$。

值得注意的是，上述式子在求$\delta_i^{(l)}$时多了个求和符号，这是因为当E展开到$l+1$层时，E可以看成是$z^{(l+1)}$的函数；当E展开到l层，E可以看成是$z^{(l)}$的函数。当E对l层的某一个$z_i^{(l)}$求导时，由于使用的是链式求导法，并且$l+1$层每一个神经元都和$z_i^{(l)}$所在的神经元有连接，故在E中，自变量$z_i^{(l)}$出现了n_{l+1}次，出现的每一次对应一个$l+1$层的$z_j^{(l+1)},1\le j\le n_{l+1}$，故由函数和的求导法则和链式求导法有：

$$\begin{aligned}\delta_i^{(l)}&=\frac{\partial E}{\partial z_i^{(l)}}=\frac{\partial E}{\partial z_1^{(l+1)}}\frac{\partial z_1^{(l+1)}}{\partial z_i^{(l)}}+\frac{\partial E}{\partial z_2^{(l+1)}}\frac{\partial z_2^{(l+1)}}{\partial z_i^{(l)}}+\cdots+\frac{\partial E}{\partial z_{n_{l+1}}^{(l+1)}}\frac{\partial z_{n_{l+1}}^{(l+1)}}{\partial z_i^{(l)}}\\&=\sum_{j=1}^{n_{l+1}}\frac{\partial E}{\partial z_j^{(l+1)}}\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}=\sum_{j=1}^{n_{l+1}}\delta_j^{(l+1)}\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}\end{aligned}$$

又因为$z_j^{(l+1)}=\sum_{k=1}^{n_l}w_{jk}^{(l+1)}a_k^{(l)}+b_j^{(l+1)}=\sum_{k=1}^{n_l}w_{jk}^{(l+1)}f(z_k^{(l)})+b_j^{(l+1)}$，故有：

$$\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}=w_{ji}^{(l+1)}f'(z_i^{(l)})$$

代入得到：

$$\delta_i^{(l)}=\sum_{j=1}^{n_{l+1}}\delta_j^{(l+1)}\frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}}=\left(\sum_{j=1}^{n_{l+1}}\delta_j^{(l+1)}w_{ji}^{(l+1)}\right)f'(z_i^{(l)})$$

这就实现了误差项的反向传播，即通过$\delta_j^{(l+1)}$计算$\delta_i^{(l)}$。把其写为矩阵形式得：

$$\delta^{(l)}=\left(\left(W^{(l+1)}\right)^T\delta^{(l+1)}\right)\odot f'(z^{(l)}),2\le l\le L-1$$

$$\frac{\partial E}{\partial W^{(l)}}=\delta^{(l)}a^{(l-1)},2\le l\le L-1$$

5.4 输出层和隐藏层的偏置参数$b^{(l)}(2 \le l \le L)$更新

对偏置参数求导得：

$$\frac{\partial E}{\partial b_i^{(l)}} = \frac{\partial E}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial b_i^{(l)}} = \delta_i^{(l)}$$

写成矩阵（向量）形式有：

$$\frac{\partial E}{\partial b^{(l)}} = \delta^{(l)}$$

5.5 神经网络算法训练流程

- 初始化参数$W^{(l)}, b^{(l)}, 2 \le l \le L$，一般初始化为比较小的接近零的随机值。
- 通过前向传播算法计算每一层的状态与激活值：

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = f(z^{(l)})$$

- 计算每一层的$\delta^{(l)}$
  - 首先计算输出层的$\delta^{(L)} = -(y - a^{(L)}) \odot f'(z^{(L)}), 1 \le i \le n_L$
  - 然后通过后向传播算法计算隐藏层的$\delta^{(l)}, 2 \le l \le L-1$

$$\delta^{(l)} = \left(\left(W^{(l+1)} \right)^T \delta^{(l+1)} \right) \odot f'(z^{(l)}), 2 \le l \le L-1$$

- 计算代价函数关于$W^{(l)}, b^{(l)}, 2 \le l \le L$的偏导数：

$$\frac{\partial E}{\partial W^{(l)}} = \delta^{(l)} a^{(l-1)}$$
$$\frac{\partial E}{\partial b^{(l)}} = \delta^{(l)}$$

- 之后使用批量梯度下降法进行参数的更新

$$W^{(l)} = W^{(l)} - \frac{\mu}{N} \sum_{i=1}^{N} \frac{\partial E_i}{\partial W^{(l)}}$$
$$b^{(l)} = b^{(l)} - \frac{\mu}{N} \sum_{i=1}^{N} \frac{\partial E_i}{\partial b^{(l)}}$$

- 迭代进行上述步骤，直到收敛为止。

6 相关实验

6.1 神经网络在手写体识别的应用

使用三层神经网络在如图 3 所示的手写体数据集上进行识别率测试。该数据集中，共有十类数字图片，每一类图片都有 500 个图像。首先进行测试集与训练集的划分，我们分别采用训练集与测试集比例为 9:1、8:2、7:3、6:4、5:5 进行识别率的测试。对于神经网络的设置，隐藏层设置了 25 个神经元，使用全连接网络进行参数训练，梯度下降迭代次数为 100 次。

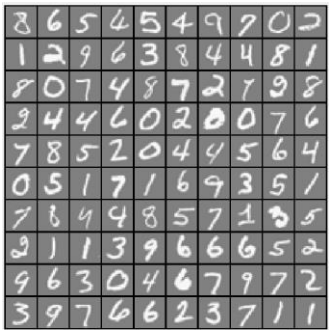

图 3：手写体数据集

测试结果如表 1 所示，可以看到，随着训练集比例的增加，使用神经网络预测的识别率较高，且基本稳定在某个高度，稳定性较好，说明训练出来的参数具有一定的鲁棒性。

表 1：神经网络算法在手写体识别上应用的效果随不同训练与测试比例的变化

Prop	9:1	8:2	7:3	6:4	5:5
Rate	92.8%	**92.9%**	**92.9%**	92.8%	91.6%

另外，固定训练集与测试集的比例为 7:3，改变算法的迭代次数（100、150、200、250、300、350），测试其识别率变化，测试结果如表 2 所示。可以看到，算法的识别率变化并没有很大，观察迭代过程的损失函数值变化可以知道（如图 4 所示），算法在一开始就已经把损失函数值减低到了很小的范围，这个前提下，损失函数下降会非常的缓慢，增加迭代次数对识别率的提升并没有明显的效果。

表 2：神经网络算法在手写体识别上应用的效果随不同迭代次数的变化

Times	100	150	200	250	300	350
Rate	92.0%	92.8%	93.1%	92.7%	92.8%	93.2%

```
Iteration    94 | Cost: 3.981388e-01
Iteration    95 | Cost: 3.980122e-01
Iteration    96 | Cost: 3.978223e-01
Iteration    97 | Cost: 3.972712e-01
Iteration    98 | Cost: 3.965356e-01
Iteration    99 | Cost: 3.963585e-01
Iteration   100 | Cost: 3.962607e-01
Iteration   101 | Cost: 3.961876e-01
Iteration   102 | Cost: 3.959102e-01
Iteration   103 | Cost: 3.957322e-01
Iteration   104 | Cost: 3.956313e-01
Iteration   105 | Cost: 3.953732e-01
Iteration   106 | Cost: 3.951445e-01
Iteration   107 | Cost: 3.943634e-01
Iteration   108 | Cost: 3.934577e-01
```

图 4：损失函数下降缓慢

6.2 神经网络在人脸数据集上的实验

在 ORL、Yale 与 AR 数据集上使用神经网络进行人脸识别实验，首先使用 PCA 降维到不同的维度，再使用神经网络（固定迭代次数为 10000 次）训练参数进行预测得到识别率 PCA+KNN 的识别率进行比较，测试结果如表 3 所示。

表 3：神经网络算法在人脸数据集上的 PCA 降低到不同维度的识别率

	20	40	60	80	100	120	140	160
ORL	84.3%	90.0%	91.7%	91.7%	**95.0%**	92.5%	**95.0%**	91.7%
AR	32.2%	50.9%	35.9%	43.4%	45.6%	**65.3%**	46.9%	46.3%
Yale	96.7%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

表 4：KNN 在人脸数据集上的 PCA 降低到不同维度的识别率

	20	40	60	80	100	120	140	160
ORL	93.3%	94.2%	92.5%	92.5%	93.3%	93.3%	93.3%	93.3%
AR	41.9%	50.3%	52.2%	54.4%	54.7%	54.7%	54.7%	54.7%
Yale	98.3%	96.7**%**	98.3%	96.7%	98.3%	98.3%	98.3%	96.7%

可以看到，神经网络模型在 ORL 与 Yale 数据集上的表现较好，特别是在 Yale 数据集上，当 PCA 降低维度达到饱和时，神经网络模型能够精准的预测出所有人脸。而在 AR 数据集上，由于该数据集受光线与角度影响较大，神经网络模型也没有很好的效果，但一定程度上也比 KNN 的识别率高。

四、实验结论或体会

详细推导了神经网络的前向传播算法与后向传播算法的优化过程，掌握了使用函数加法的求导法则与链式求导法推导后向传播算法的过程，使用了神经网络模型在手写体识别上与人脸识别上进行应用，测试了识别率，并与之前学习过的方法进行比较。

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。