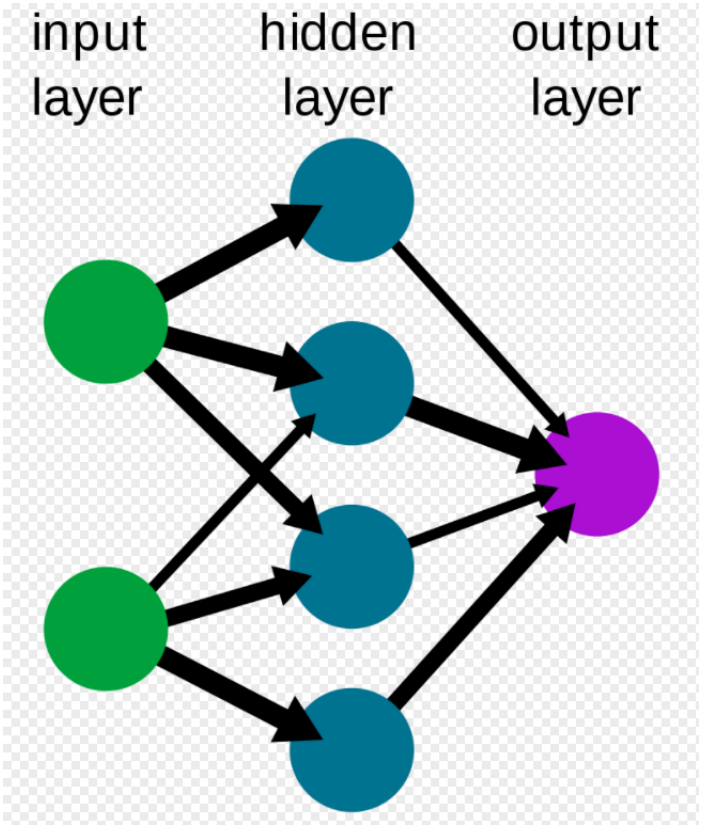

图 1　Neural Network

假设我们定义隐藏层的第一个神经元为 h_1，输入层从上到下依次为 $i_1, i_2, ..., i_n$。则

$$h_1 = A(i_1 \times a_{11} + i_2 \times a_{2_1} + ... + i_n \times a_{n_1} + b_1) \tag{1}$$

其中，a_{ij} 表示输入层的第 i 个神经元与隐藏层的第 j 个神经元的权值，b_1 为偏置项，A 为一个非线形激活函数。同理，

$$h_2 = A(i_1 \times a_{12} + i_2 \times a_{2_2} + ... + i_n \times a_{n_2} + b_2) \tag{2}$$

我们可以发现，$h_1, h_2, h_3, ...$ 有着非常紧密的形式，我们进一步使用矩阵的形式来表示他们：

$$\vec{h} = A(\vec{i} \cdot a + \vec{b})$$

如果在神经网络的存在第二层隐藏层的话，那么可以表示为：

$$\vec{h}' = A'(\vec{h} \cdot a + \vec{b})$$

需要注意的是，对于神经网络而言，每一层的激活函数可以是不一样的。本实验首先定义了一个没有隐藏层的神经网络，即直接对输入做简单处理，然后接入输出层。如果需

要扩展隐藏层，只需要修改相对应的代码即可，这就要求本实验的代码有着极高的要求。否则再修改的时候可能需要大量的地方重写。

3 网络和参数的初始化

首先我们对没有隐藏层的网络做一个数学形式的表达。

$$\vec{l_0} = A_1(data + b_0)$$
$$output = A_2(\vec{l_0} \cdot W_1 + b_1)$$

1. data 是一个长度为 784 的向量，直接将输入的图片进行拉平即可。
2. b_0 作为一个偏置向，也是一个长度为 784 的向量。
3. 激活函数 A_1 使用双曲正切 $tanh$ 函数。
4. W_1 为一个 784×10 的矩阵。
5. b_1 也是长度为 784 的向量。
6. 激活函数 A_2 使用 $SoftMax$ 函数[7]。

一般而言，神经网络输出层的激活函数在处理不同的任务的时候采用不同的激活函数。对于回归问题，一般使用恒等函数，而分类问题使用 $SoftMax$ 函数。$SoftMax$ 的计算如图 2 所示：

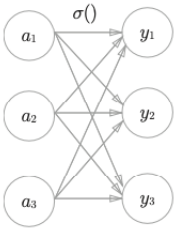

图 2　SoftMax 函数

SoftMax 函数的数学表达为：

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)} \tag{3}$$

虽然 SoftMax 函数的实现很简单，但是在计算机的运算上有一定的缺陷。这个缺陷就是溢出问题，因为要进行大量的指数运算，而仅仅 e^{100} 就会变成一个后面带 40 多个 0 的超大值。当指数为 1000 的时候，会直接返回一个表示无穷大的 inf。所以需要做一个小

的优化。

$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)} \\ &= \frac{C\exp(a_k)}{C\sum_{i=1}^{n}\exp(a_i)} \\ &= \frac{\exp(a_k+\log C)}{\sum_{i=1}^{n}\exp(a_i+\log C)} \\ &= \frac{\exp(a_k+C')}{\sum_{i=1}^{n}\exp(a_i+C')} \end{aligned} \tag{4}$$

这里的 C' 可以使用任何值，但是为了防止溢出，一般会使用输入中的最大值。

对于当前网络的参数[8]，我们对 b_0, b_1 都采用全 0 的初始化[9]，而 W_1 我们采取在区间 $[-\sqrt{\frac{6}{784+10}}, \sqrt{\frac{6}{784+10}}]$ 内随机初始化。

4 MNIST 数据集

MNIST 是一个手写体数字的图片数据集，该数据集来由美国国家标准与技术研究所（National Institute of Standards and Technology (NIST)）发起整理，一共统计了来自 250 个不同的人手写数字图片，其中 50% 是高中生，50% 来自人口普查局的工作人员。该数据集的收集目的是希望通过算法，实现对手写数字的识别。

官网上提供了数据集的下载，主要包括四个文件：

文件下载	文件用途
train-images-idx3-ubyte.gz	训练集图像
train-labels-idx1-ubyte.gz	训练集标签
t10k-images-idx3-ubyte.gz	测试集图像
t10k-labels-idx1-ubyte.gz	测试集标签

在上述文件中，训练集一共包含了 60,000 张图像和标签，而测试集一共包含了 10,000 张图像和标签。测试集中前 5000 个来自最初 NIST 项目的训练集，后 5000 个来自最初 NIST 项目的测试集。前 5000 个比后 5000 个要规整，这是因为前 5000 个数据来自于美国人口普查局的员工，而后 5000 个来自于大学生。在实验中，我们将 60,000 张的训练集切分为 50,000 张的训练集和 10,000 张的验证集。

5 网络训练

5.1 偏差估计

对于一张图片，我们喂入网络中后，定义得到的输出为 O_1。假设我们输入图片的标签是 9, 网络经过若干次训练后，输出的 $O_1 = (0.01, 0.01, ..., 0.91)$。虽然已经可以很大几率得到正确的标签，但是离我们完美的输出 $O_2 = (0, 0, ..., 1)$ 还是有一定的差异。即：

$$(O_2 - O_1)^2 = (0.01)^2 + (0.01)^2 + ... + (1 - 0.91)^2 \tag{5}$$

我们记这个偏差为 $Loss\ Function$。我们最终的目的就是尽可能的减少这个偏差值。其中取平方的原因是因为为了平衡正负误差之间的相互抵消。O_1 是通过网络的参数计算得到的，所以要优化 $Loss\ Function$，便需要去优化参数。

5.2 梯度下降

本实验采取随机梯度下降法去进行优化，所以我们需要去手动计算各种导。首先给出一些常用的普通导数性质：

$$\begin{aligned}
\frac{d}{dx}(f+g) &= \frac{d}{dx}f + \frac{d}{dx}g \\
\frac{d}{dx}(af) &= a\frac{d}{dx}f \\
\frac{d}{dx}(fg) &= (\frac{d}{dx}f)g + f(\frac{d}{dx}g) \\
\frac{d}{dx}(\frac{f}{g}) &= \frac{(\frac{d}{dx}f)g - (\frac{d}{dx}g)f}{g^2} \\
\frac{d}{dx}f(g(x)) &= (\frac{d}{dx}f)(g(x))(\frac{d}{dx}g) \\
\frac{d}{dx}f(g_1(x), g_2(x)) &= (\partial_1 f)\frac{d}{dx}g_1(x) + (\partial_2 f)\frac{d}{dx}g_2(x)
\end{aligned} \tag{6}$$

首先去求 $L(Loss\ Function)$ 的梯度:

$$\begin{aligned}
L &= (y - y_{pred})^2 \\
&= (y_0 - y_{pred0})^2 + (y_1 - y_{pred1})^2 + ... + (y_9 - y_{pred9})^2 \\
&= (y - A_2(\vec{l_0} \cdot W_1 + \vec{b_1}))^2 \\
&= (y - A_2(A_1(x + \vec{b_0}) \cdot W_1 + \vec{b_1}))
\end{aligned} \tag{7}$$

为了方便后续的计算，记 $L_0 = (y_0 - y_{pred0})^2, L_1 = (y_1 - y_{pred1})^2, ..., L_9 = (y_9 - y_{pred9})^2$。随机梯度下降要做的是先求出 $\frac{\partial L}{\partial b_{0i}}, \frac{\partial L}{\partial b_{1i}}, \frac{\partial L}{\partial W_{1ij}}$。

对于 $\frac{\partial L}{\partial b_{1i}}$:

$$\frac{\partial L}{\partial b_{1i}} = \frac{\partial L_0}{\partial b_{1i}} + ... + \frac{\partial L_9}{\partial b_{1i}} \tag{8}$$

我们首先计算 $\frac{\partial L_0}{\partial b_{1i}}$:

$$\begin{aligned}
\frac{\partial L_0}{\partial b_{1i}} &= \frac{\partial}{\partial b_{1i}}(y_0 - y_{pred0})^2 \\
&= 2(y_0 - y_{pred0})\frac{\partial}{\partial b_{1i}}(y_0 - y_{pred0}) \\
&= -2(y_0 - y_{pred0})\frac{\partial}{\partial b_{1i}}(y_{pred0}) \\
&= -2(y_0 - y_{pred0})\frac{\partial}{\partial b_{1i}}A_{2,0}(\vec{l_0}\cdot W_1 + \vec{b_1}) \qquad \text{第 0 个分量} \\
&= -2(y_0 - y_{pred0})\sum_m \partial_m[A_{2,0}(\vec{l_0}\cdot W_1 + \vec{b_1})]\frac{\partial}{\partial b_{1i}}(\vec{l_0}\cdot W_{1,m} + \vec{b_{1,m}})
\end{aligned} \tag{9}$$

其中 $\frac{\partial}{\partial b_{1i}}\vec{l_0}\cdot W_{1,m} = 0$，而 $\frac{\partial}{\partial b_{1i}}\vec{b_{1,m}}$ 只有当 $i = m$ 的时候为 1，其他情况下为 0。所以公式 (9) 可以优化为:

$$\begin{aligned}
\frac{\partial L_0}{\partial b_{1i}} &= -2(y_0 - y_{pred0})\sum_m \partial_m[A_{2,0}(\vec{l_0}\cdot W_1 + \vec{b_1})]\frac{\partial}{\partial b_{1i}}(\vec{l_0}\cdot W_{1,m} + \vec{b_{1,m}}) \\
&= -2(y_0 - y_{pred0})\partial_i[A_{2,0}(\vec{l_0}\cdot W_1 + \vec{b_1})]
\end{aligned} \tag{10}$$

同理有：

$$\begin{aligned}
\frac{\partial L_1}{\partial b_{1i}} &= -2(y_1 - y_{pred1})\partial_i[A_{2,1}(\vec{l_0}\cdot W_1 + \vec{b_1})] \\
&..... \\
\frac{\partial L_1}{\partial b_{1i}} &= -2(y_9 - y_{pred9})\partial_i[A_{2,9}(\vec{l_0}\cdot W_1 + \vec{b_1})]
\end{aligned} \tag{11}$$

把所有的都累加起来可以得到:

$$\frac{\partial L_1}{\partial b_{1i}} = -2(\vec{y} - \vec{y_{pred}})\cdot\partial_i[A_2(\vec{l_0}\cdot W_1 + \vec{b_1})] \tag{12}$$

到此为止，我们已经推导完了 $\frac{\partial L_0}{\partial b_{1i}}$ 的计算过程。接下来我们推导 $\frac{\partial L_0}{\partial W_{1,ij}}$。

$$\begin{aligned}
\frac{\partial L_0}{\partial W_{1,ij}} &= -2(y_0 - y_{pred0})\sum_m \partial_m[A_{2,0}(\vec{l_0}\cdot W_1 + \vec{b_1})]\frac{\partial}{\partial W_{1,ij}}(\vec{l_0}\cdot W_{1,m} + \vec{b_{1,m}}) \\
&\text{其中}\frac{\partial}{\partial W_{1,ij}}(\vec{l_0}\cdot W_{1,m} + \vec{b_{1,m}}) = \frac{\partial}{\partial W_{1,ij}}(\vec{l_0}\cdot W_{1,m}) = \frac{\partial}{\partial W_{1,ij}}\sum_n l_{0,n}W_{1,nm}
\end{aligned} \tag{13}$$

当 $i = n, j = m$ 的时候，$\frac{\partial}{\partial W_{1,ij}}\sum_n l_{0,n}W_{1,nm} = 1$，否则为 0。此时：

$$\frac{\partial L_0}{\partial W_{1,ij}} = -2(y_0 - y_{pred0})[\partial_j A_{2,0}]\vec{l_{0,i}} \tag{14}$$

进一步得到:

$$\frac{\partial L}{\partial W_{1,ij}} = -2\vec{l_{0,i}} A_2 \cdot (y - y_{pred}) \tag{15}$$

接下来进行 $\frac{\partial L_0}{\partial b_{0,i}}$ 的推导。和公式 (9) 的推导类似，可以轻易得出：

$$\begin{aligned}\frac{\partial L_0}{\partial b_{0,i}} &= -2(y_0 - y_{pred0}) \sum_m \partial_m [A_{2,0}(\vec{l_0} \cdot W_1 + \vec{b_1})] \frac{\partial}{\partial b_{0,i}} ((\vec{l_0} \cdot W_1)_m + \vec{b_{1,m}}) \\ &= -2(y_0 - y_{pred0}) \sum_m \partial_m [A_{2,0}(\vec{l_0} \cdot W_1 + \vec{b_1})] ((\frac{\partial}{\partial b_{0,i}} \vec{l_0}) \cdot W_1)_m\end{aligned} \tag{16}$$

对于 $\frac{\partial}{\partial b_{0,i}} \vec{l_0}$，可以拿出来单独先计算。

$$\begin{aligned}\frac{\partial}{\partial b_{0,i}} \vec{l_0} &= \frac{\partial}{\partial b_{0,i}} A_1(x + b_0) \\ &= \sum_n [\partial_n A_1(x + b_0)] \frac{\partial}{\partial b_{0,i}} (x + b_{0,n}) \\ &= [\partial_i A_1(x + b_0)]\end{aligned} \tag{17}$$

再将公式 (17) 带回公式 (16) 中：

$$\begin{aligned}\frac{\partial L_0}{\partial b_{0,i}} &= -2(y_0 - y_{pred0}) \sum_m \partial_m [A_{2,0}(\vec{l_0} \cdot W_1 + \vec{b_1})] \frac{\partial}{\partial b_{0,i}} ((\vec{l_0} \cdot W_1)_m + \vec{b_{1,m}}) \\ &= -2(y_0 - y_{pred0}) \sum_m \partial_m [A_{2,0}(\vec{l_0} \cdot W_1 + \vec{b_1})] \partial_i A_1 \cdot W \\ &= -2(y_0 - y_{pred0})(\partial_i A_1 \cdot W_1 \cdot \vec{\partial} A_{2,0})\end{aligned} \tag{18}$$

于此我们可以推导出：

$$\frac{\partial L}{\partial b_{0,i}} = -2\partial_i A_1 \cdot W_1 \cdot \vec{\partial} A_2 \cdot (\vec{y} - \vec{y_{pred}})$$

5.3 网络性能

在实验中，我们共进行了 8 个 Epoch 的训练，其中学习率选择的是 1。最终在测试集上的准确率为 92.71%。图 3 为每个 Epoch 在训练时训练集和验证集的准确率和 Loss 值。

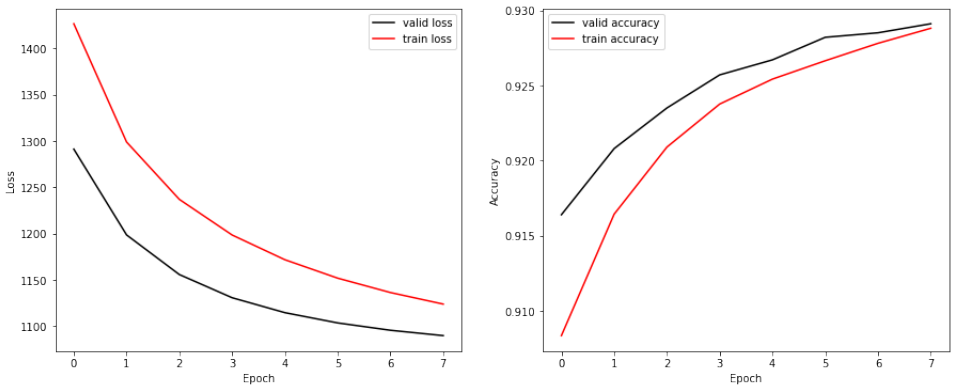

图 3　训练时的性能

但是可以发现的是，验证集的准确率却比训练集的准确率要高。出现这个情况的很大原因可能是数据在做切分的时候没有随机打乱。但是这并不影响我们在测试集上的准确率。

6 总结

如果将网络再扩展一层，便可以在测试集上达到高于 98% 的准确率。但是追求准确率不是本文的重点，通过本文可以清楚的理解反响传播的实质。但同时也体现出深度学习框架如 Pytorch，TensorFlow 等的便利性。因为对于反向传播而言，深度学习的框架会自动帮我们计算，而不需要人工手动的做大量的推导。

对于技术而言，由于我们的网络结构相对简单。本文没有使用数据增强[10]，Dropout[11] [12]，Batch Normalization[13]，惩罚项等技术去防止过拟合。同时相比于卷积神经网络[14] 而言，在输入的时候直接将图片拉伸，由于图片的二值化的，所以不存在丢失空间信息等问题。但是如果输入的 RGB 三通道图，卷积神经网络有着它独特的优势[15]。

参考文献

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2] Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J. Learning representations by back-propagating errors. nature, 1986, 323(6088): 533-536.

[3] Cohen, Gregory and Afshar, Saeed and Tapson, Jonathan and Van Schaik, Andre. EMNIST: Extending MNIST to handwritten letters. IEEE, 2017: 2921-2926.

[4] Xiao, Han and Rasul, Kashif and Vollgraf, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.