实验报告

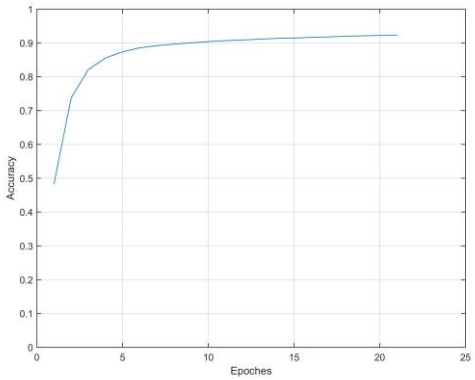
姓名：张家治　　学号：1601110508

网络结构

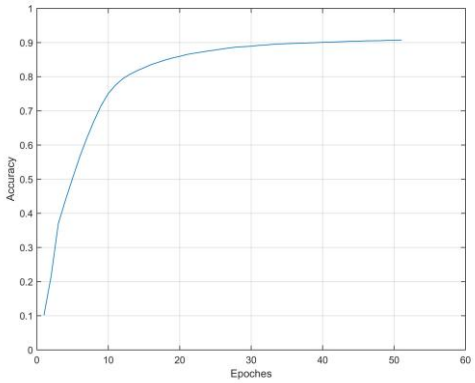
输入层：784(28*28) 个神经元
隐层：一个隐层，100 个神经元
输出层: Softmax 输出，10 个神经元

测试的两种激活函数

1. ReLU: performs good (accuracy reached 90% in 10 epoches)



2. Sigmoid: not so effecient (accuracy reached 90% in 50 epoches)



关于 MLP 的推导

详见下文，或查看我的博客：
http://www.cosmozhang.com/2016/12/11/mathematics-of-bp-nn.html

BP 神经网络的推导

姓名：张家治　　学号：1601110508

（原文出自我的博客：http://www.cosmozhang.com/2016/12/11/mathematics-of-bp-nn.html）

神经网络的数学描述

作如下假设：

网络共有n层

激活函数为$f(x)$

对第$l+1$层：

neuron 数目为$s_{(l+1)}$

第j个 neuron 接收上一级第 i 个 neuron 的输入为：$x_{ij}^{(l+1)}$

第j个 neuron 与上一级第i个 neuron 的连接权值为：$w_ij^{(l+1)}$

第j个 neuron 的偏置为：$b_j^{(l+1)}$

第j个 neuron 的输入和为：$u_j^{(l+1)}=\sum_{i=1}^{sl} w_ij^{(l+1)}x_ij^{(l+1)}+b_j^{(l+1)}$

第j个 neuron 的激活输出为：$t_j^{(l+1)}=f(u_j^{(l+1)})$

梯度下降法

首先说梯度下降（gradient descent）方法，梯度下降是一个最常用的数学优化方法，要优化一个损失函数$f(x)$，就是要找到它的极小值，也就是要让x向负梯度方向移动，梯度下降法每次迭代将x向负梯度方向移动η，η为学习率，也就是：

$$x_{n+1}=x_n-\eta f'(x_n)\ ,\ \eta>0$$

BP 算法

然后说 **BP 算法**，在 BP 算法中，我们要优化的也是损失函数，损失函数取成残差的模方，即

$$E=\frac{1}{2}\left\|y-u^{(n)}\right\|^2$$

将 BP 算法看作问题g，即

$E=g(W^{(2)},W^{(3)},\cdots,W^{(n)},b^{(1)},b^{(2)},\cdots,b^{(n)};x^{(1)})$，优化权值$W^{(i)}$和偏置$b^{(i)}$。

δ 规则

BP 的精髓在于 **δ 规则**，而 δ 规则的精髓在于它建立了相邻两层神经元的损失函数的偏导的关系，这个关系可以解释为：后级神经元的误差依网络权重传递给前级神经元。下面给出 δ 规则的推导，注意偏导数的链式法则（chain rule）会在推导中起到重要的作用。

首先，我们定义一个 δ 值，δ 的含义是 “误差对偏置项的偏导”

$$\delta_i^{(l)} = \frac{\partial E}{\partial b_i^{(l)}} = \frac{\partial E}{\partial u_i^{(l)}} \cdot \frac{\partial u_i^{(l)}}{\partial b_i^{(l)}} = \frac{\partial E}{\partial u_i^{(l)}} \cdot 1 = \frac{\partial E}{\partial u_i^{(l)}}$$

上面这个式子非常重要，他告诉我们$\delta_i^{(l)} = \frac{\partial E}{\partial b_i^{(l)}} = \frac{\partial E}{\partial u_i^{(l)}}$，原因很简单，因为$b$在$u$中是偏置项，也就是$u$说对于$b$的偏导是 1。

下面推导相邻两层之间 δ 的关系。

$$\begin{aligned}
\delta_i^{(l)} &= \frac{\partial E}{\partial u_i^{(l)}} \\
&= \sum_{j=1}^{s_{(l+1)}} \frac{\partial E}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial u_i^{(l)}} \\
&= \sum_{j=1}^{s_{(l+1)}} \delta_i^{(l+1)} \frac{\partial u_j^{(l+1)}}{u_i^{(l)}}
\end{aligned}$$

其中

$$\begin{aligned}
\frac{\partial u_j^{(l+1)}}{\partial u_i^{(l)}} &= \frac{\partial \left(b_j^{(l+1)} + \sum_{k=1}^{s_l} w_{kj}^{(l+1)} f\left(u_k^{(l)}\right)\right.}{\partial u_i^{(l)}} \\
&= \frac{\partial \left(w_{ij}^{(l+1)} f\left(u_i^{(l)}\right)\right)}{\partial u_i^{(l)}} \\
&= w_{ij}^{(l+1)} f'\left(u_i^{(l)}\right)
\end{aligned}$$

因此得到

$$\delta_i^{(l)} = f'\left(u_i^{(l)}\right) \sum_{j=1}^{s_{(l+1)}} w_{ij}^{(l+1)} \delta_j^{(l+1)}$$

观察上式的形式，我们可以给出如下解释：预测误差在神经网络中反向传递，前一个神经元对于后一个神经元的误差的贡献就是它们之间的连接权值，而某一个神经元所造成的误差是它对所有下级神经元造成的误差的总和。

有了 δ 规则，我们可以确定所有的$\frac{\partial E}{\partial b_i^{(l)}}$，也就是说确定了所有的偏置项的优化规则，而权值的优化规则也很好确定：

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \frac{\partial E}{\partial u_j^{(l)}} \cdot \frac{\partial u_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} x_{ij}^{(l)}$$

上面所有的推导形成了这样一个优化思路：通过在层与层之间的权值关系确定所有 δ，通过所有 δ 确定所有权值和偏置的梯度方向，进而确定优化规则。

在层之间递归确定 δ 时，还缺少最后一环，即最后层也就是输出层的 δ 值，这也不难求：

$$\delta_i^{(n)} = \frac{\partial E}{\partial u_i^{(n)}} = (y_i^{(n)} - f(u_i^{(n)}))(-f'(u_i^{(n)}))$$

看，一切都是链式法则。现在我们有了递归的初值，有了递归规则，有了递归项对于优化规则的决定规则，就可以逐次迭代优化网络了。