Lab1 实验报告

一、　识别液晶数字

首先实现基本的 BP 网络，sigmoid 函数选择的是单极性 S 型激活函数。该部分输入层为 7，输出层为 10。初始的隐藏层设为 18。

第一次尝试时，学习率设为 0.05，输入层到隐藏层的权重、隐藏层到输出层的权重都初始化为-0.1 到 0.1 间的随机数，隐藏层与输出层的阈值设置为-0.02 到 0.02 间的随机数，训练 2000w 次后，用验证集测试，均方误差为 0.1869709207694931，其中训练集的内容都能正确判定，但是对 2、6 的残缺值的判断，差距有点大，分别判定为 8 和 4。

查看上次实验输出的权重值和偏置值，发现权重值量级在个位，但是偏置值并未在训练中调整，所以仍然保持着初始的百分位量级的值，这样的话，偏置值对结果基本不会产生任何影响，所以修改 BP 网络，在训练中增加了对偏置值的调整。调整算法在文末图片中。再次实验，此次均方误差降低到 0.0588957966226782，对一个残缺 6 仍然判断成 4，说明 BP 网络在训练必须同时调整权重与偏置。

观察到，此时输入层到隐藏层的权重很少出现-0.1 到 0.1 的数字，大多在 0.5 到 3 之间，这样的话，那初始的权重可能设置的较小。而现在两层的偏置也多为 1 左右。于是调整了初始的权重和偏置的设置。分别设置成(-1,1)/num，(0,1)+0.2,(0,1)*0.2。这种设置方法是老师推荐的初始超参数，但是一开始进行试验的时候忘记了这个提醒，所以进行到第三次试验才设置了这样的参数。试验结果，误差进一步下降，均方误差为 0.03678205855488113，同时所有残缺数字都能正确预测。

想进一步降低误差，尝试调整学习率，降低学习率为 0.03，调整的结果不太理想，总误差提高到了 0.0991196626152209，并且有残缺数字 6 判断错误。再次调整学习率为 0.07，总误差仍然相比 0.05 提高到了 0.0529676986758371，并且对残缺 6 的判断还是出错。再次增大学习率为 0.1，均方误差提高到 0.10804556961644696。所以舍弃了学习率的调整，设置为 0.05。

后来采取了一种动态学习率调整的算法，具体如下。但是训练过程过长，最后放弃。

$$\eta\ (k+1)=\begin{cases}1.05\eta(k), & \text{当}E(k+1)<E(k)\\ 0.7\eta(k), & \text{当}E(k+1)>1.04E(k)\\ \eta(k), & \text{其他}\end{cases}$$

二、　拟合正弦函数

输出层为 1，输出层为 1，隐藏层设为 5。权重和偏置、学习率的初始值采用了识别液晶数字中最后相应的超参数开始实验。实验结果发现，对正数的拟合效果很好，但是对负数的拟合结果全为正数。意识到这时的 sigmoid 函数值域在（0,1），所以得重新写一个 BP 网络，选择新的 sigmoid 函数为双曲正切函数，该函数的值域为（-1,1）。

实验后得到均方误差为 0.3398208731333422，其中明显地，对负数的预测值和实际值相差很大。改变隐藏层个数至 7 与 10，均方误差相比 5 个有提升，负数问题没有得到解决。于是改回到 5 个。发现所有负数的预测值都无限接近于 0，所以应该是出现了过拟合现象。首先对激活函数进行修改为 tanh(x)=2σ(2x)−1，超参数不变。同时修改权重调整中的导数为 1-σ(2x)^2。结果得到明显变化。均方误差为 5.023117105449782E-6。对验证集每一个样本的预测误差都在 0.001 左右。改变学习率至 0.02，均方误差提高到 7.545596474596222E-6。改变学习率至 0.1，均方误差降低到 4.954255933661359E-6，多次尝试后，可以降低到 1.928079791397015E-6。动态改变学习率，使得在训练 1000w 次后，学习率降为 0.035；1500w 次后，降为 0.02，实验后均方误差提高到 3.917456831306431E-6。故不再调整学习率。

三、　手写体识别---BP 网络

首先对每个图片进行了二值化处理，将每个文件处理成只由 0,1 组成的 28*28 的行向量，然后将每个字母对应文件下所有图片的行向量添加到一个 all.txt 中，然后读取到程序中。

二值化的方法采用了一种自适应阈值算法，算法名为最大类间方差法（又叫大津法,简称 OTSU）。该算法是按照图像的灰度，将图像分为了背景和目标两部分，利用类间方差区分。类间方差越大，两部分的区别越明显。若两部分分类错误，则类间方差会变小。

算法简要：

记 T 为目标与背景的分割阈值，目标像素点数占整幅图像的比例记为 ω0， 平均灰度为 μ0；背景像素点数占整幅图像的比例为 ω1,其平均灰度为 μ1。则图像的总平均灰度为 μ=ω0*μ0+ω1*μ1。类间方差记为 g。目标和背景图象的方差：g=ω0*(μ0-μ)*(μ0-μ)+ ω1*(μ1-u)*(μ1-μ)= ω0*ω1*(μ0-μ1)^2，采用遍历的方法，当方差 g 最大时，可以认为此时目标和背景差异最大，也就是此时的灰度是最佳阈值。

输入层为 28*28，输出层为 8，隐藏层第一次设为 5，学习率设为 0.05。遍历 10000 次后结果通过验证集测试，对 A 至 H 的识别率都在 80%至 85%之间。

因为训练时间过长，将遍历次数减少到 500 次。因为之前层数太少，所以讲层数提高到 28 层，希望可以提高识别率。此次试验，出现了局部最小值的情况，对 B、E 的识别率降低到 50%下，互相判定为对方的情况居多。同时对 F 的识别率也降低到 60%。增加遍历次数到 1000，对除了 B 之外的其他字母，识别率均提高到 85-92%间。B 的识别率仅仅提高到 68%。增加学习率到 0.1。B 的识别率是上升到 80%，但是 A、C 的识别率下降到 76%左右。所有验证集总的识别率为 84.53%。

尝试减少隐含层个数，降低为 20 个，训练后识别率大为下降，降低到 68%。尝试增加隐含层个数，增加为 40 个，识别率为 85.59%。此时主要识别错误的为 B，于是在训练过程中单独增加对 B 的训练，这次调整后对 B 的识别率大大提高，但是与 B 相似的 E 识别率却大为降低，说明出现了过拟合。然后调整了训练的方式，在一次遍历中，分为两次遍历完一个文件夹，也就是两个 for 循环，每一个训练一半的样本，并且在训练前计算当前权重下当前样本的误差，若小于 0.001，则不再进行权重调整。结果使得验证集的识别率提高到了 88.16%。继续细分为 3 块，验证集识别率提高到 89.125%。分别修改隐藏层个数为 28 与 35 个，识别率保持在 87-88%。提高到 50 个，验证集识别率为 89.72%。

由于对同样结果的样本连续识别仍然达到 300 次，担心出现过拟合。于是更改了训练方式，将所有训练集全部添加到了一个 Map 内，乱序进行识别。在隐含层为 50 个节点，学习率为 0.1 的情况下，验证集识别率达到了 90.52%。修改学习率为 0.05，验证集识别率达到 91.09%，训练集识别率达到 98.24%。

四、　手写体识别—CNN 网络

卷积神经网络采用了 Keras 框架，后台使用 Theano。首先在 data.py 中处理数据，读取所有的图片，然后转为数组形式，同时分割文件名字符串设置目标。在 cnn.py 中，搭建卷积神经网络。从输入层出来，第一层为卷积层，采用 6 个卷积核，每个卷积核大小 5*5，将 28*28 变为 24*24,；第二层为采样层，变为 12*12；第三层继续为卷积层，12 个卷积核，每个卷积核大小 5*5 变为 8*8；第四层再次采样，变为 4*4；然后全连接层，为 128 个节点；最后输出到 8 个节点。除了最后一层激活函数为 softmax 外，其他层采用 sigmoid 函数。同时设置 Dropout 层防止过拟合。

首先设置学习率为 0.1，迭代 10 次，在 epoch 为 10 时，训练集识别率可达到 84%，验证集可达到 88%。继续增大迭代次数为 50 次，在 epoch 为 50 时，训练集和验证集的识别率

可达到90%左右，增大到150次，当epoch为145次以上时，训练集和验证集的识别率可达到93%左右。改变学习率为0.05，与之前相比，在epoch为10与50时，识别率变化不大。

改变激活函数为tanh，收敛速度明显加快，在epoch为13时，识别率即可达到90%，在80-150次之间，识别率在92-93%上下。增大学习率到0.1，效果不太好，识别率在epoch达到65次左右才达到90%，150次迭代后识别率为91%左右。设置学习率为0.02，150次迭代后训练集识别率达到93.56%，验证集识别率达到92.87%。

信号前向传播：（预测）

隐含层第 j 个节点输入信号： $net_j = \sum_{i=1}^{N} w_{ij} u_i + \theta_j$ （u_i 输入）

………… 输出 …： $o_j = \varphi(net_j)$

输出层 … k …… 输入 …： $net_k = \sum_{j=1}^{L} (w_{jk} o_j + \theta_k) = \sum_{j=1}^{L} (w_{jk} \varphi(\sum_{i=1}^{N} w_{ij} u_i + \theta_j) + \theta_k)$

………… 输出 …： $y_k = \psi(net_k)$

误差反向传播：（训练）：

$$E = \frac{1}{2} \sum_{k=1}^{M} (d_k - y_k)^2 .$$

隐含 → 输出 · 权值： $\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{jk}} = -\eta \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{jk}}$

…… 偏置： $\Delta \theta_k = -\eta \frac{\partial E}{\partial \theta_k} = -\eta \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial \theta_k} = -\eta \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial \theta_k}$

输入 → 隐含 权值： $\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}}$

偏置： $\Delta \theta_j = -\eta \frac{\partial E}{\partial \theta_j} = -\eta \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial \theta_j}$

$\frac{\partial E}{\partial y_k} = -\sum_{p=1}^{P} \sum_{k=1}^{M} (d_k^{(p)} - y_k^{(p)})$ $\quad \frac{\partial net_k}{\partial w_{jk}} = o_j .$ $\quad \frac{\partial net_k}{\partial \theta_k} = 1 .$ $\quad \frac{\partial net_j}{\partial w_{ij}} = u_i .$ $\quad \frac{\partial net_j}{\partial \theta_j} =$

$\frac{\partial E}{\partial o_j} = -\sum_{p=1}^{P} \sum_{k=1}^{M} w_{jk} (d_k^{(p)} - y_k^{(p)}) \psi'(net_k) .$ $\quad \frac{\partial y_k}{\partial net_k} = \psi'(net_k)$ $\quad \frac{\partial o_j}{\partial net_j} = \varphi'(net_j)$

代码中： $\Delta w_{jk} = \eta \sum_{p=1}^{P} \sum_{k=1}^{M} (d_k^{(p)} - y_k^{(p)}) \psi'(net_k) o_j .$

$$\Delta w_{ij} = \eta \sum_{p=1}^{P} \sum_{k=1}^{M} w_{jk} (d_k^{(p)} - y_k^{(p)}) \psi'(net_k) \varphi'(net_j) u_i$$

$$\Delta \theta_k = \eta \sum_{p=1}^{P} \sum_{k=1}^{M} (d_k^{(p)} - y_k^{(p)}) \psi'(net_k) .$$

$$\Delta \theta_j = \eta \sum_{p=1}^{P} \sum_{k=1}^{M} w_{jk} (d_k^{(p)} - y_k^{(p)}) \psi'(net_k) \varphi'(net_j) .$$

sigmoid 函数 $f(\sigma) = \frac{1}{1+e^{-\sigma}}$ $\quad f'(\sigma) = f(\sigma)(1 - f(\sigma)) .$