深度学习与神经网络第一次课程项目

王逸群 19307110397

2022.3.18

目标：对输入的 16×16 手写数字灰度图进行识别，输出 $y \in \{-1, 1\}^{10}$；
数据集：存储于`digits.mat`；
激活函数：Tanh 函数；
损失函数：平方损失函数；
优化方法：随机梯度下降法。
模型评估：在测试集上的错误率为 47%，在验证集上的错误率如图 1。

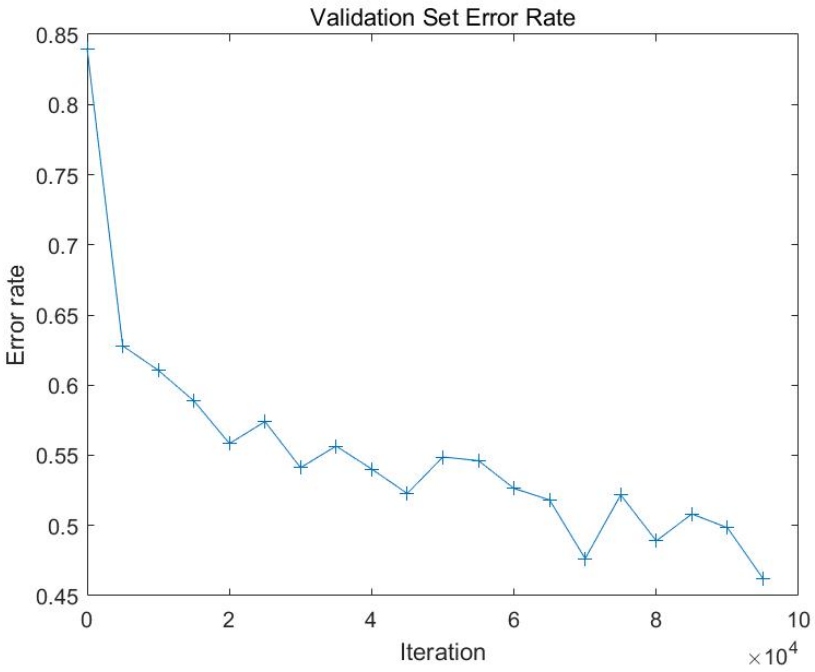

图 1: 原始模型在验证集上的错误率

1 网络结构

程序中，变量nHidden为一个向量，其维度表示网络隐藏层的个数，各个分量表示每个隐藏层的神经元个数。

基础模型中，nHidden = 10，将其修改得到的验证集和测试集错误率如图 2和表 1所示。

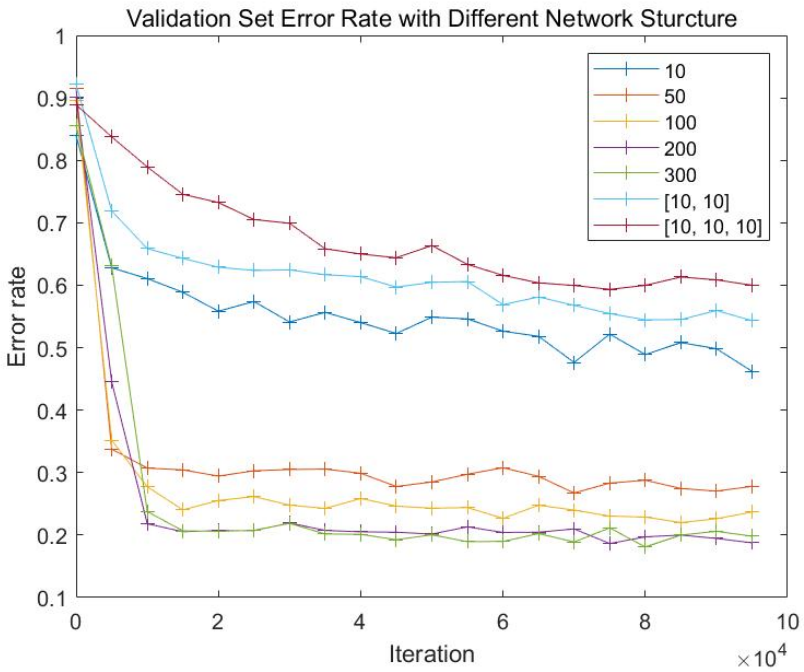

图 2: 不同网络结构的模型在验证集上的错误率

可以看到，随着单层网络神经元个数的增加，模型在测试集上的错误率逐渐减小，但是减小的程度趋缓。鉴于训练模型的时间成本随着网络神经元个数的增加而增加，神经元个数的选取是精度与成本的权衡。

另一方面，随着网络隐藏层个数的增加，模型在测试集上的错误率逐渐增加，可能的原因是出现了过拟合现象。

网络结构	测试集错误率
`nHidden = 10`	0.470
`nHidden = 50`	0.255
`nHidden = 100`	0.228
`nHidden = 200`	0.181
`nHidden = 300`	0.177
`nHidden = [10, 10]`	0.536
`nHidden = [10, 10, 10]`	0.562

表 1: 不同网络结构的模型在测试集上的错误率

2 学习率与 Momentum

在基础模型中，学习率始终为`alpha = 1e-3`，且没有引入 Momentum。本节尝试使用修改学习率常数、学习率衰减、引入 Momentum 的方式来改进模型，得到的验证集和测试集错误率如图 3和表 2所示。

初始学习率	学习率衰减	Momentum	测试集错误率
`alpha = 1e-3`	无	无	0.470
`alpha = 1e-2`	无	无	0.267
`alpha = 1e-4`	无	无	0.584
`alpha = 1e-3`	无	0.9	0.322
`alpha = 1e-2`	指数衰减	无	0.277
`alpha = 1e-2`	余弦衰减	无	0.259

表 2: 不同学习方式的模型在测试集上的错误率

可以看到，随着常数学习率的增加，验证集错误率的收敛速度变快，但是错误率变得越发不稳定。为了解决这个问题，采用指数衰减和余弦衰减两种学习率衰减方式，两者效果相近。

另一方面，使用 Momenteum 也能改进模型的训练效果。

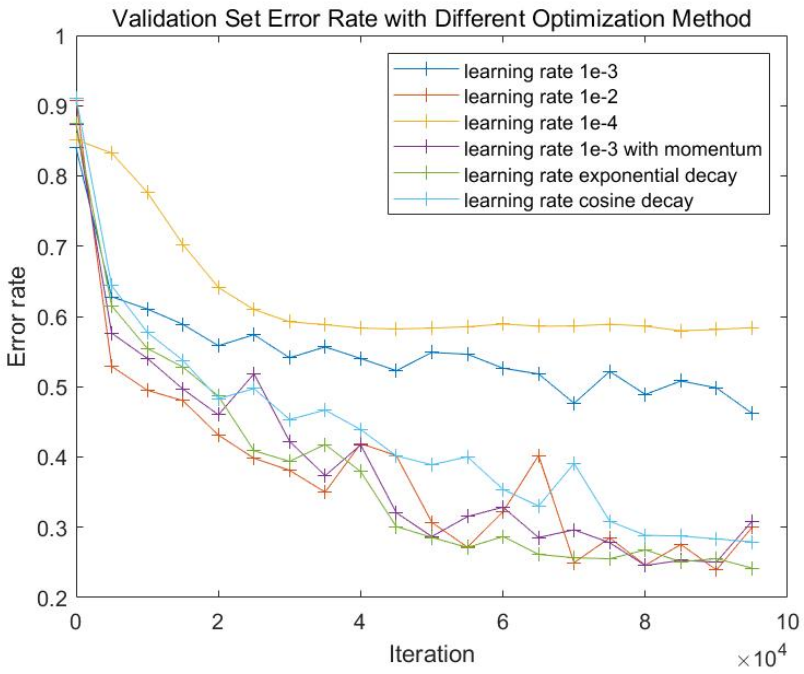

图 3: 采用不同优化方式的模型在验证集上的错误率

3 损失函数计算

基础模型中，在计算损失函数时，使用了矩阵计算，而非以下标作为循环变量的循环计算，并且避免进行不必要的条件判断。如反向传播算法：

```
% output layer
error = 2 * error;
gradOutput = gradOutput + Activation{end}' * error;
error = sech(netActivation{end}) .^ 2 .* ...
    (error * weightsOutput');
% hidden layers
for indexHidden = length(nHidden) - 1: -1: 1
    gradHidden{indexHidden} = gradHidden{indexHidden} + ...
        Activation{indexHidden}' * error;
    error = (error * weightsHidden{indexHidden}') .*  ...
        sech(netActivation{indexHidden}) .^ 2;
end
% input layer
gradInput = gradInput + X(indexInput,:)' * error;
```

其中，error表示误差项，初始值是模型输出值与真实值的差；

4 正则化

基础模型中，没有加入正则化。本节尝试调整正则化参数来改进模型，得到的验证集和测试集错误率如图 4和表 3所示。

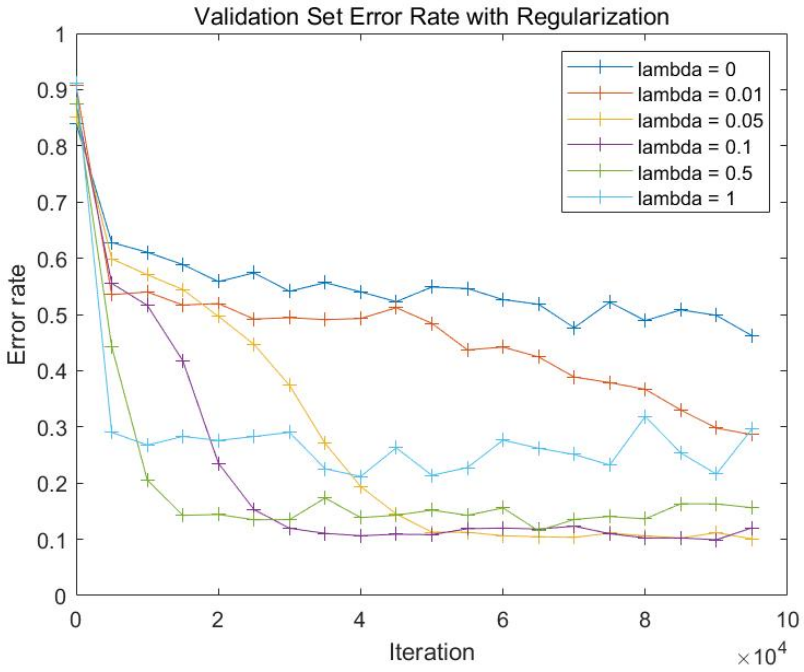

图 4: 不同正则化参数的模型在验证集上的错误率

可以看到，正则化参数为 0.05 时，正则化效果还不明显。之后，随着正则化参数的增加，验证集错误率的收敛速度变快，但是错误率也变大。

5 交叉熵损失函数

基础模型中，损失函数是平方损失函数。本节尝试加入 Softmax 函数、使用交叉熵损失函数来改进模型，得到的验证集和测试集错误率如图 5和表 4所示。

正则化参数	测试集错误率
0	0.470
0.01	0.286
0.05	0.104
0.1	0.100
0.5	0.164
1	0.235

表 3: 不同正则化参数的模型在测试集上的错误率

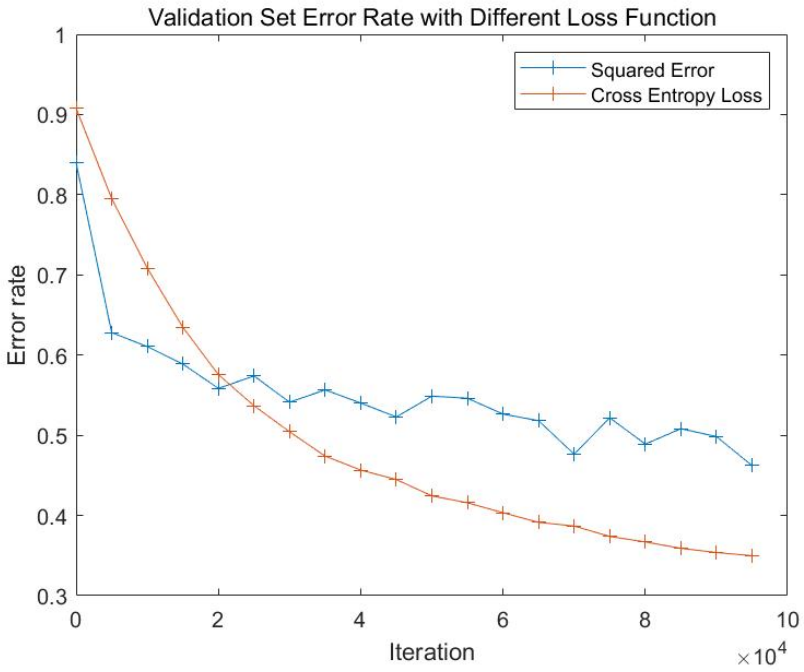

图 5: 不同损失函数的模型在验证集上的错误率

损失函数	测试集错误率
平方损失函数	0.470
交叉熵损失函数	0.353

表 4: 不同损失函数的模型在测试集上的错误率

可以看到，使用交叉熵损失函数会减小模型的错误率，增加错误率的稳定性，但也会减小收敛速度。

6 偏置

基础模型中，输入带有偏置项。本节使得每个隐藏层中的一个神经元成为偏置项，以此来改进模型，得到的验证集和测试集错误率如图**??**和表**??**所示。