人工神经网络 HW1 MNIST Digit Classification with MLP

冯卓尔
计 86 2017011998
fengzhuoer-thu@outlook.com

摘要：本文主要记录了 MNIST Digit Classification with MLP 作业中的一些实验，测试了 ReLU 与 Sigmoid 激活函数在本任务中的优劣，EuclideanLoss 与 SoftmaxCrossEntropyLoss 在相应网络中的性能，以及测试了各项参数对任务的准确率与 loss 中的影响。
关键字：多层感知机 MLP　数字图像识别　激活函数　损失函数　神经网络

一、网络结构说明

1.1 基本参数

通过测试 Google 的 neural network playground https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.49537&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false)我发现了一些常见的学习率参数 0.001，0.01，0.03，0.1，0.3，1，3，10 等。经过短期试验，我发现学习率 0.03 在本机上运行收敛速度较快，同时能够得到比较满意的准确度，因此我主要选用 0.03 为学习率。其余参数的设置通过实验以及同学交流，参与交流的同学为陈博涵、张哲昕。另外关于参数设置我还借鉴了该博客 https://www.cnblogs.com/guoyaohua/p/8542554.html

Weight_decay 为 0，batch_size 为 100，momentum 为 0.9，如果无特殊说明，linear 层的 std 为 0.001。

1.2 运行环境

本机运行 MacBook Air，内存为 4 GB 1600 MHz DDR3，处理器为 1.4 GHz Intel Core i5，机型性能较次。（可以发现本机运行时间较长，经过排查，应是本机 MacBook Air 机型较老算力不足所致）

1.3 无隐藏层全连接网络

学习率为 0.01，损失函数 Euclidean。

层	1
类型	Linear
输入节点数	784

输出节点数	10

1.4 一个隐藏层激活函数为 ReLU 网络

学习率为 0.03，损失函数Euclidean。

层	1	2	3	4
类型	Linear	ReLU	Linear	ReLU
输入节点数	784	256	256	10
输出节点数	256	256	10	10

1.5 一个隐藏层激活函数为 ReLU 网络损失函数为 SoftmaxCrossEntropy

学习率为 0.03，损失函数 SoftmaxCrossEntropy。

层	1	2	3	4
类型	Linear	ReLU	Linear	ReLU
输入节点数	784	256	256	10
输出节点数	256	256	10	10

1.6 一个隐藏层激活函数为 Sigmoid 网络

学习率为 0.03，损失函数 Euclidean。

层	1	2	3	4
类型	Linear	Sigmoid	Linear	Sigmoid
输入节点数	784	256	256	10
输出节点数	256	256	10	10

1.7 两个隐藏层激活函数为 ReLU 网络

学习率为 0.03，损失函数 Euclidean，linear 层的 std 为 0.01。

层	1	2	3	4	5	6
类型	Linear	ReLU	Linear	ReLU	Linear	ReLU
输入节点数	784	256	256	128	128	10
输出节点数	256	256	128	128	10	10

1.8 两个隐藏层激活函数为 Sigmoid 网络

学习率为 0.03，损失函数 Euclidean，linear 层的 std 为 0.01。

层	1	2	3	4	5	6
类型	Linear	Sigmoid	Linear	Sigmoid	Linear	Sigmoid
输入节点数	784	256	256	128	128	10
输出节点数	256	256	128	128	10	10

1.9 两个隐藏层激活函数为 ReLU 网络损失函数为 SoftmaxCrossEntropy

学习率为 0.08，损失函数 SoftmaxCrossEntropy，linear 层的 std 为 0.01。

层	1	2	3	4	5	6
类型	Linear	ReLU	Linear	ReLU	Linear	ReLU
输入节点数	784	256	256	128	128	10
输出节点数	256	256	128	128	10	10

二、模型运行结果

2.1 汇总实验结果

注：无特殊说明，本模块网络损失函数均使用 Euclidean（绘制限制，同时 ReLU 在本任务中表现更好，Euclidean 与 ReLU 的组合性能更好）

由于画图限制，在此图中所展示的曲线默认损失函数为 Euclidean 损失函数。关于两种函数的性能讨论详见后文，这里只给出了 ReLU 激活函数条件下两种损失函数的对比曲线。

网络名称	无隐藏层	一个隐藏层 ReLU	一个隐藏层 ReLU+softmax	一个隐藏层 Sigmoid	两个隐藏层 ReLU	两个隐藏层 ReLU+softmax	两个隐藏层 Sigmoid (higher linear std)	两个隐藏层 Sigmoid (normal linear std)
Training_acc	0.8532	0.9970	**1.0000**	0.9756	0.9980	**1.0000**	0.9946	0.9308

Training_loss	0.1911	0.0036	0.0004	0.0244	0.0011	**8.370e-06**	0.0058	0.0564
Testing_acc	0.8624	0.9825	0.9818	0.9665	**0.9836**	0.9834	0.9783	0.9274
Testing_loss	0.1930	0.0203	0.0780	0.0301	**0.0130**	0.1533	0.0178	0.0597
运行时长	**0:01:02.340**	0:10:14.125	**0:08:24.391**	0:09:23.604	0:11:23.919	0:10:33.457	0:09:22.075	0:11:12.226

2.2 网络 training_acc 图表

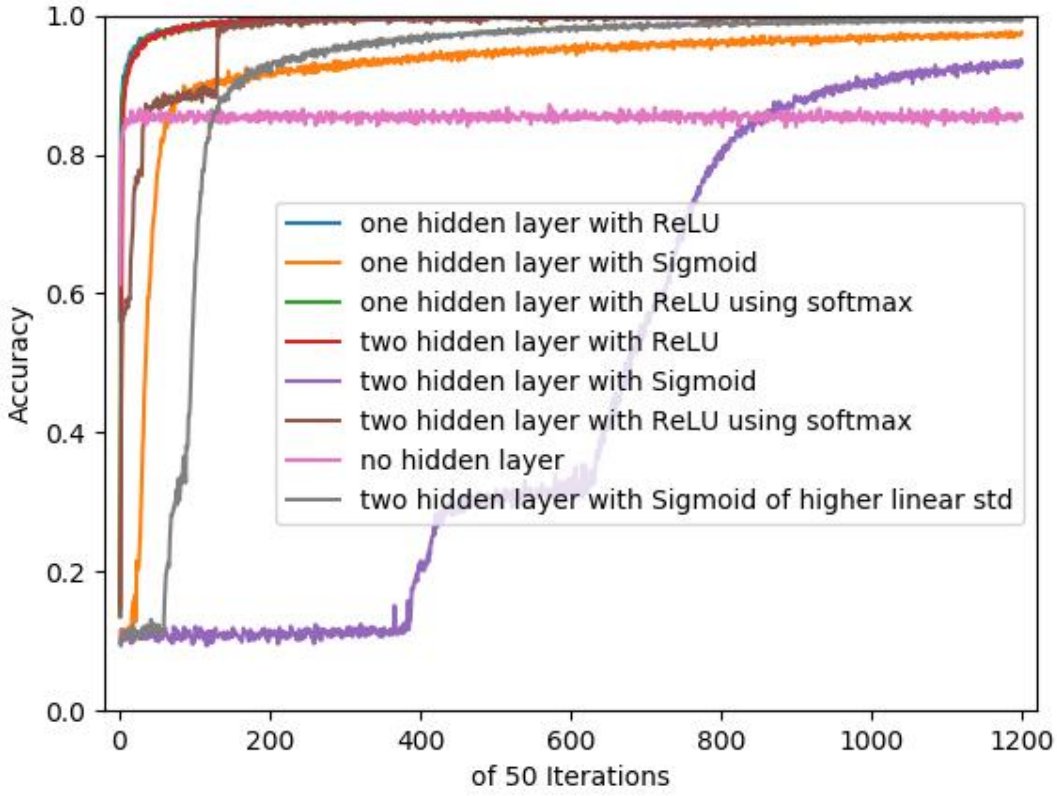

图 1　整合网络准确率

2.3 网络 training_loss 图表

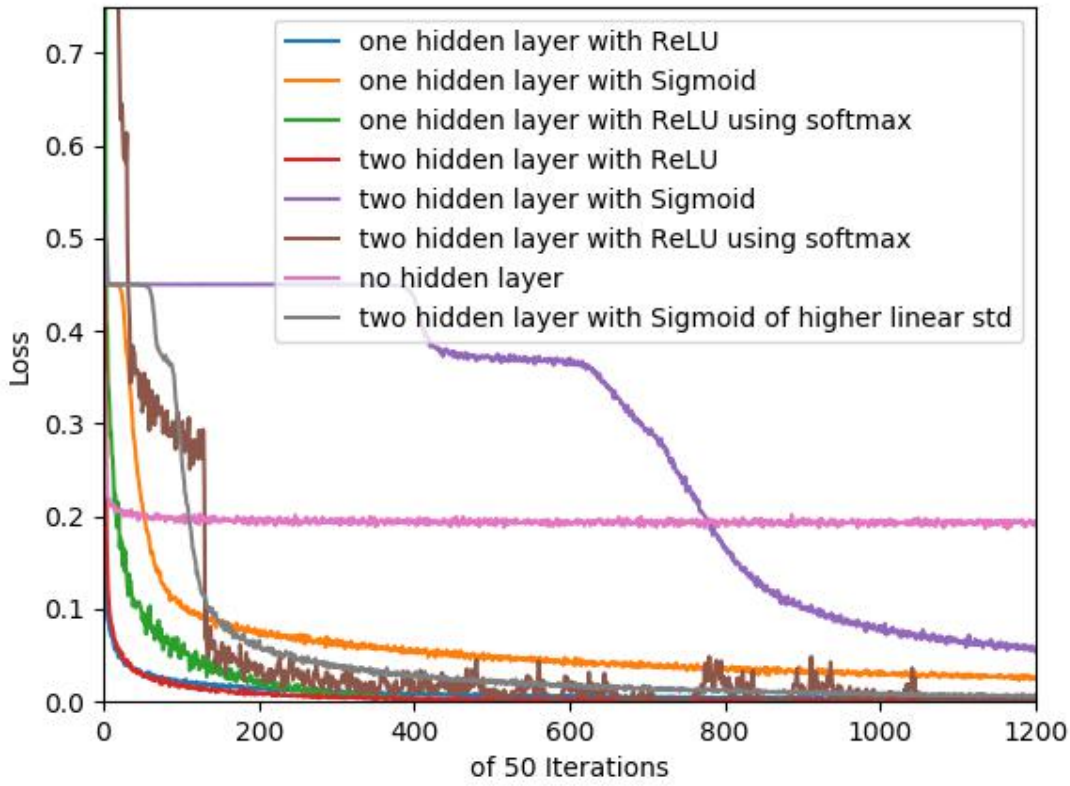

图 2　整合网络 loss

三、实验结论与讨论

3.1 Sigmoid 激活函数与 ReLU 激活函数在 MNIST Digit Classification 任务上的区别

对比曲线 one hidden layer with ReLU 与 one hidden layer with Sigmoid、two hidden layer with ReLU 与 two hidden layer with Sigmoid，与 Sigmoid 激活函数相比，我们发现使用 ReLU 激活函数的准确度高，运行时间短，loss 更小。

3.2 EuclideanLoss 与 SoftmaxCrossEntropyLoss 的对比

对比曲线 one hidden layer with ReLU 与 one hidden layer with ReLU using softmax、two hidden layer with ReLU 与 two hidden layer with ReLU using softmax，我们发现使用了 softmax 交叉熵 loss 的网络运行时间短，训练准确度高（甚至达到了 100%的训练准确度）。值得注意的是，虽然 softmax 交叉熵 loss 的训练准确度、训练 loss 均优于 Euclidean loss，但是在测试准确度和 loss 上 softmax 交叉熵的性能不及 Euclidean loss，这可能是因为出现了过拟合的现象。

另外，Euclidean 与 softmax 交叉熵针对不同的激活函数表现出不同的性能。在 ReLU 激活函数网络中，softmax 交叉熵能够提升准确率、降低 loss；但是在 Sigmoid 激活函数网络中，softmax 交叉熵将 loss 提升至一个较大值（1.5 左右）。虽然我不清楚这是什么原因，但是这个现象验证了 loss 与准确率并不是负相关，它们之间有一定

的独立性。另外，在给定的学习率条件下，softmax 交叉熵增加了 Sigmoid 网络的运行时间，略微降低了准确率，根据 3.5 的结论，准确率的升降或许会因学习率设定不同而有所不同。

网络	ReLU+Euclidean (learning rate = 0. 03)	ReLU+softmax (learning rate = 0. 03)	Sigmoid+Eucilidean (learning rate = 0. 03)	Sigmoid+softmax (learning rate = 0. 03)
Training_acc	0.9970	**1.0000**	**0.9756**	0.9608
Training_loss	0.0036	**0.0004**	**0.0244**	1.4989
Testing_acc	**0.9825**	0.9818	**0.9665**	0.9564
Testing_loss	**0.0203**	0.0780	**0.0301**	1.5044
运行时长	0:10:14.125	**0:08:24.391**	**0:09:23.604**	0:10:16.582

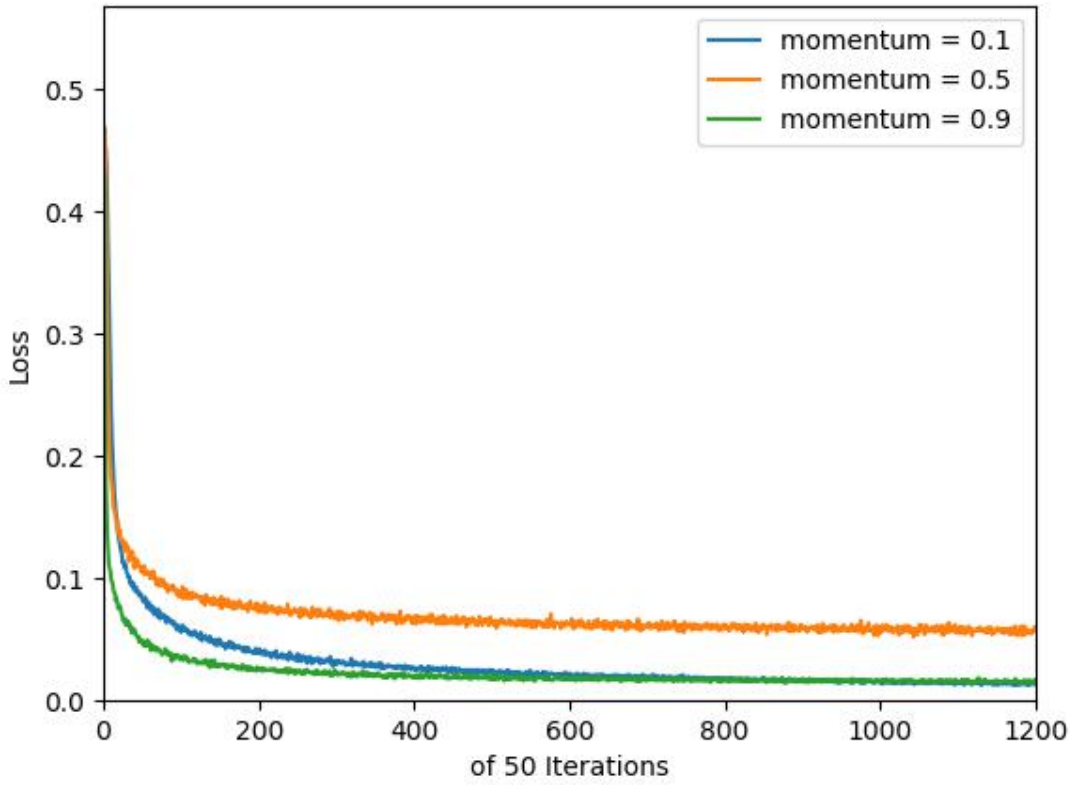

图 3　单个隐藏层不同 loss 函数 loss

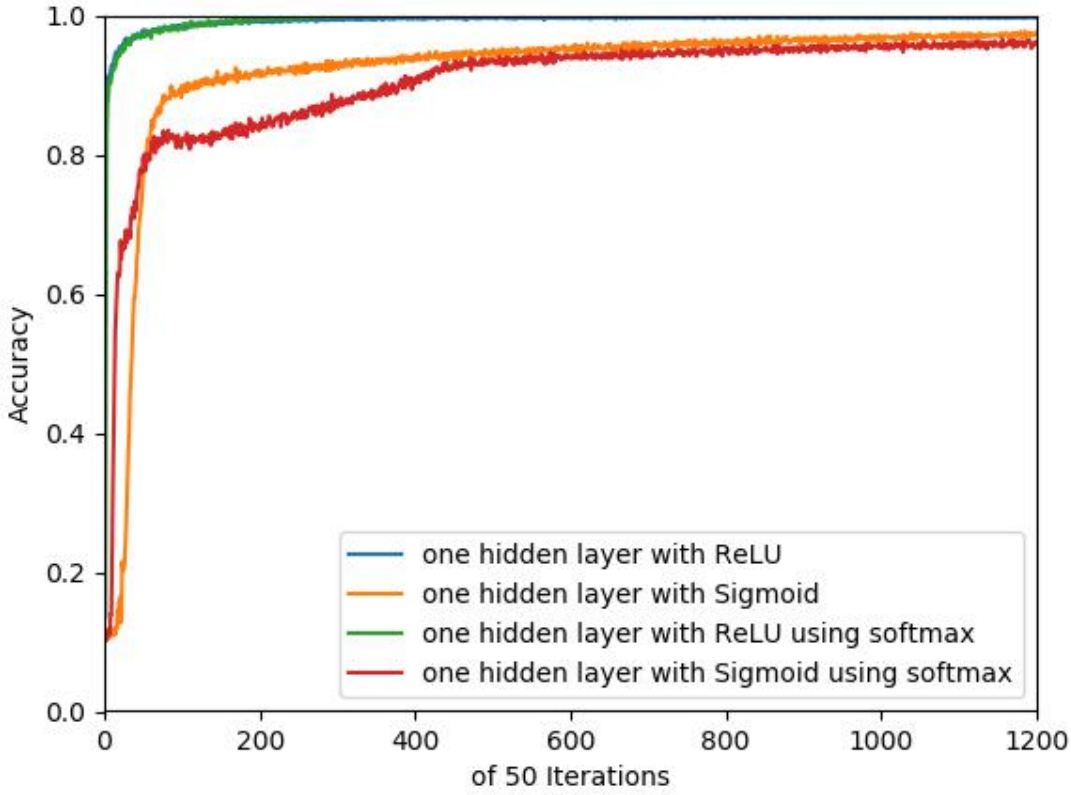

图 4　单个隐藏层不同 loss 函数准确率

另外，这两个 loss 函数的差距在两个隐藏层的网络中也呈现出单个隐藏层的特征。

网络	ReLU+Euclidean (learning rate = 0.03)	ReLU+softmax (learning rate = 0.08)	Sigmoid+Eucilidean (learning rate = 0.03)	Sigmoid+softmax (learning rate = 0.08)
Training_acc	0.9980	**1.0000**	**0.9946**	0.9854
Training_loss	0.0011	**8.370e-06**	**0.0058**	1.4753
Testing_acc	**0.9836**	0.9834	**0.9783**	0.9725
Testing_loss	**0.0130**	0.1533	**0.0178**	1.4879
运行时长	0:11:23.919	**0:10:33.457**	**0:09:22.075**	0:11:45.847

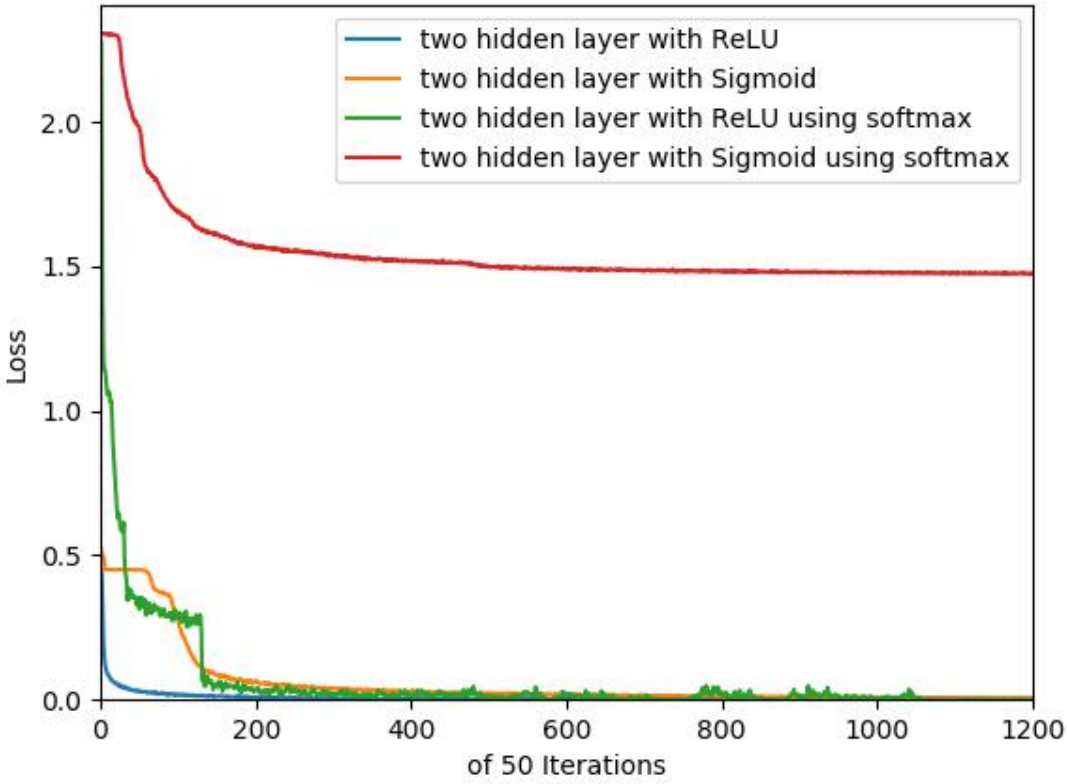

图 5　两个隐藏层不同 loss 函数 loss

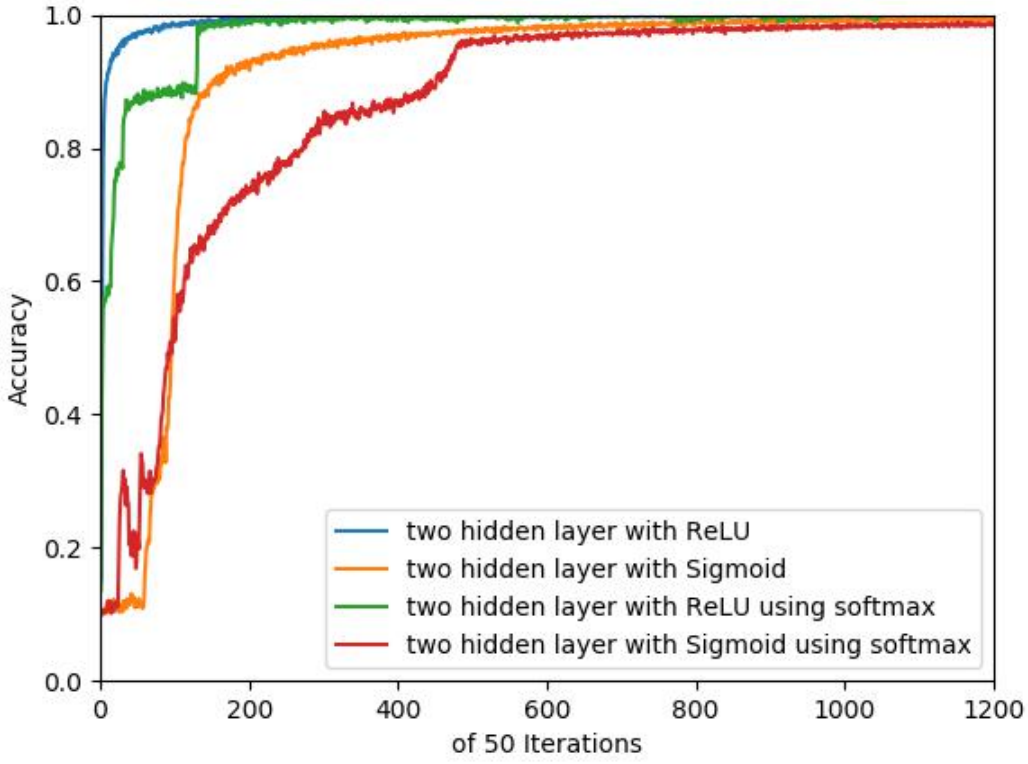

图 6　两个隐藏层不同 loss 函数准确率

3.3 隐藏层数量对 MNIST Digit Classification 任务的影响

通过对于一个隐藏层、两个隐藏层的 ReLU 激活函数的实验结果，结合基本的复杂度分析，两个隐藏层的计算量更大（多了数百个全连接节点），因而运行时间长。但是在其他性能上，由于 ReLU 激活函数本身已经很接近 100%准确度了，因而没有明显差距，但是 Sigmoid 激活函数的运行结果来看，两个隐藏层的网络除时间外性能均优于单层网络。

隐藏层是必须的，在图中可以看出，单独加全连接网络的 loss 和准确度比其他任何网络都差。

另外，优于增加隐藏层带了的计算量和未知量数目的大幅增加，必要的调整学习率与网络初始 std 是必要的，在 2.2 的图中我们能看到三条曲线 one hidden layer with Sigmoid、two hidden layer with Sigmoid、two hidden layer with Sigmoid，调整的原因是，如果不调整则到相同的训练 epoch 之后，多隐藏层的网络还没达到收敛，虽然表面上控制了变量，但是对比的基数不同，为了快速达到收敛，需要同时提高其学习率。他们的区别见下表

网络	one hidden layer with Sigmoid	two hidden layer with Sigmoid	two hidden layer with Sigmoid
Linear 层初始 std	0.001	0.001	0.01
学习率	0.03	0.03	0.03

3.4 隐藏层新增节点的数量设置

我尝试了两种节点设置。AB 两个网络的属性列于下表

层	1	2	3	4	5	6
类型	Linear	ReLU	Linear	ReLU	Linear	ReLU
A 输入节点数	784	**256**	**256**	**128**	**128**	10
A 输出节点数	**256**	**256**	**128**	**128**	10	10
B 输入节点数	784	**512**	**512**	**256**	**256**	10
B 输出节点数	**512**	**512**	**256**	**256**	10	10

符合预期的是，隐藏节点越多的网络的收敛速度更慢，由 3.3 得出的结论，需要更高的学习率，因而若想要增加节点并且达到同样的准确率，需要大幅增大学习率。结果如下

网络	A	B
Training_acc	**0.9980**	0.9010
Training_loss	**0.0011**	0.0497
Testing_acc	**0.9836**	0.8887
Testing_loss	**0.0130**	0.0605
运行时长	**0:11:23.919**	0:23:35.060

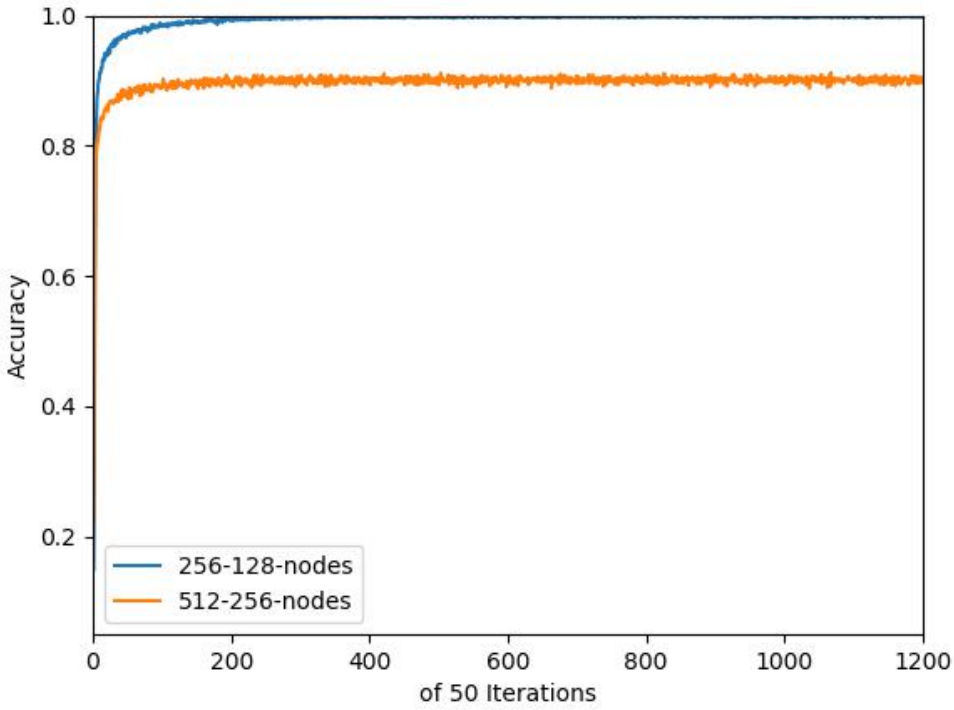

图 7　不同隐藏层节点个数网络准确率

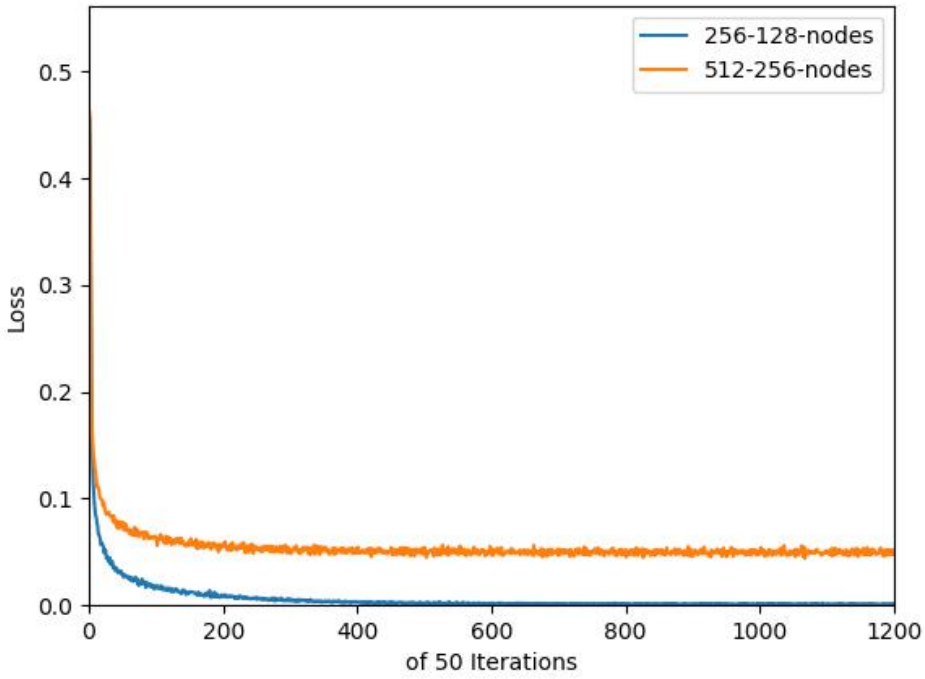

图 8　不同隐藏层节点个数网络 loss

3.5 学习率对 MNIST Digit Classification 任务的影响

针对 1.9 中（两层 ReLU+softmax 交叉熵损失函数）的网络，我做了三组实验，列于下表

学习率	0.03	0.05	0.08
Training_acc	0.4996	0.6056	**1.0000**
Training_loss	1.1536	1.1412	**8.370e-06**
Testing_acc	0.4967	0.5957	**0.9834**
Testing_loss	1.2115	1.2001	**0.1533**
运行时长	0:11:23.997	0:11:29.718	**0:10:33.457**

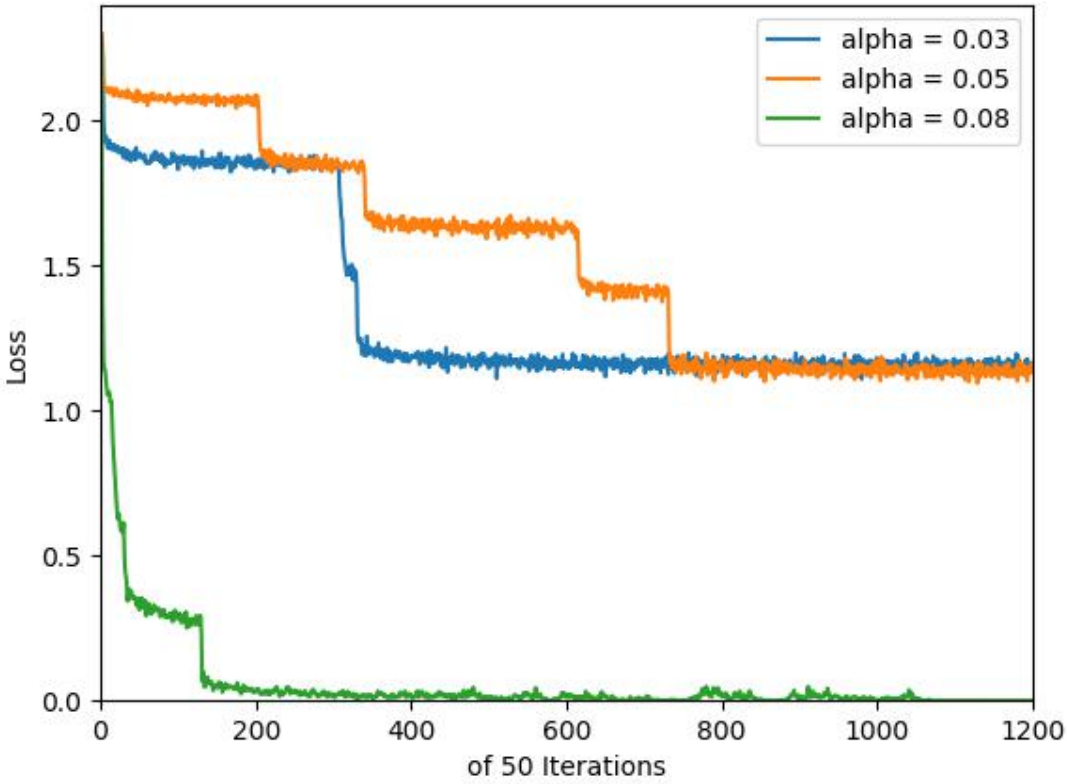

图 9　两层 ReLU+softmax 交叉熵损失函数网络不同学习率 loss

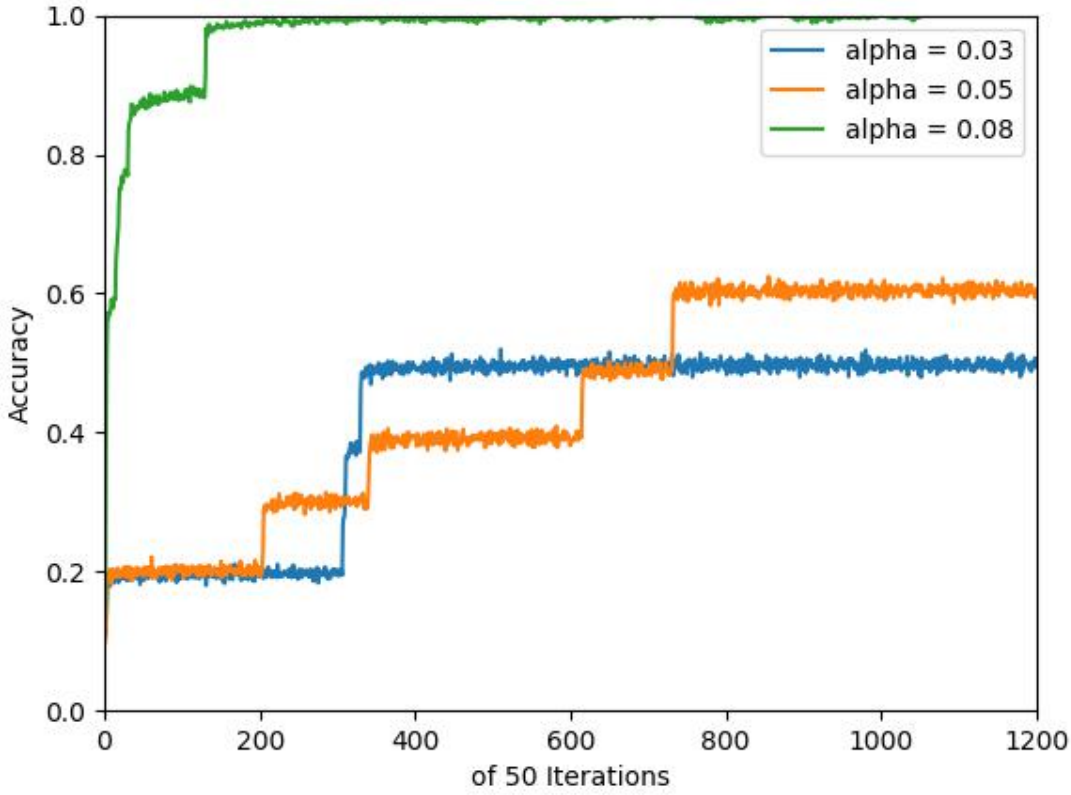

图 10　两层 ReLU+softmax 交叉熵损失函数网络不同学习率准确率

从结果可以看出，对于计算量相对较大的网络，如果学习率不足，可能导致梯度下降时陷入局部最小值，无法迈出足够的步长，导致有限的训练 epoch 内收敛在了很低的 acc 内。

但是，学习率过大容易导致过拟合现象。表现为训练高准确度，测试准确度比训练准确度低一定量。

3.6 weight_decay 对 MNIST Digit Classification 任务的影响

虽然有许多参考资料支持痕量的 weight_decay 能够提升性能，但是在实验中，我使用 weight_decay = 0.00005 与 0 在单层 Sigmoid 网络中的性能对比如下表

Weight_decay	0.00005	0
Training_acc	0.9624	**0.9756**
Training_loss	0.0373	**0.0244**
Testing_acc	0.9614	**0.9665**
Testing_loss	0.0372	**0.0301**

运行时长	0:09:43.084	**0:09:23.604**

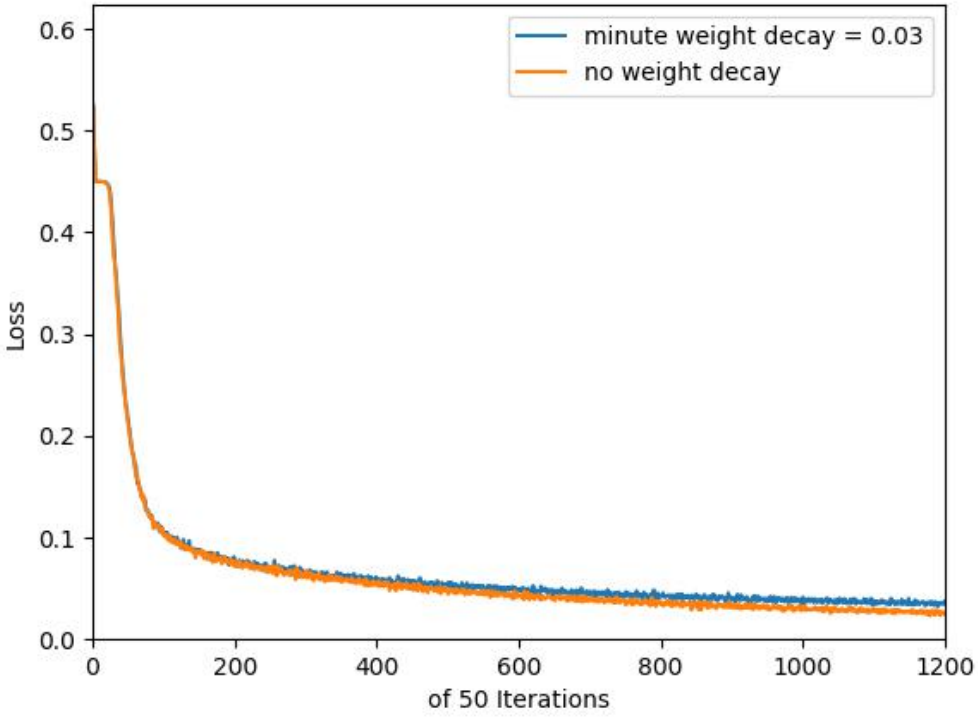

图 11　不同 weight_decay 单层 Sigmoid 网络 loss

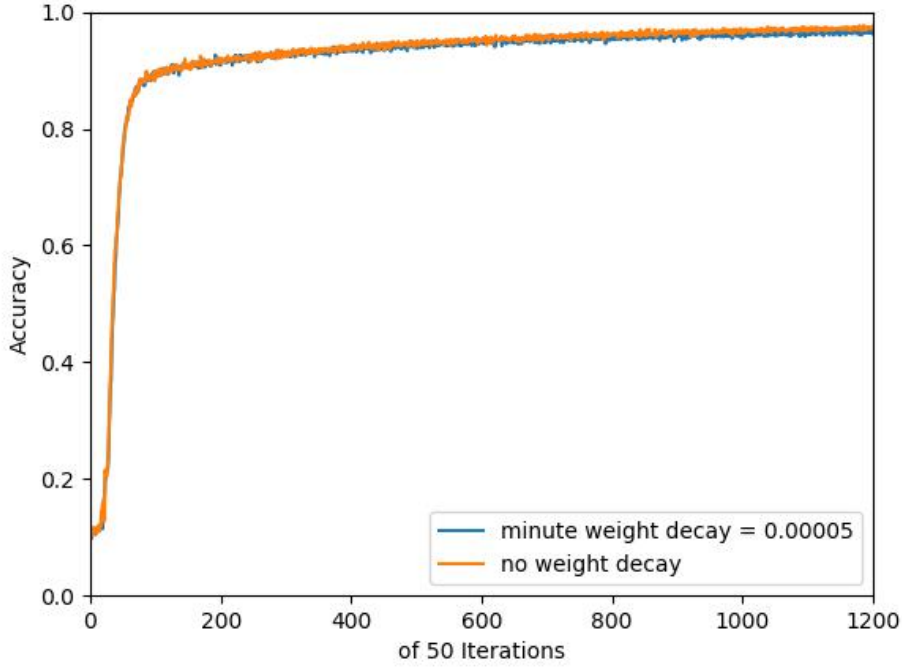

图 12　不同 weight_decay 单层 Sigmoid 网络准确率

3.7 batch_size、max_epoch 对 MNIST Digit Classification 任务的影响

max_epoch 主要影响的是训练迭代的次数，对于一个网络，如果在一定的 epoch 数量内能够达到收敛，那么此时 max_epoch 最高取比该 epoch 稍大的值，既能使之收敛，又能防止可能出现的过拟合现象、

batch_size 主要影响每一次迭代的“视野”。Batch_size 过小，单次迭代的稳定性下降，会受到偏差较大的样本的影响，在最终收敛阶段准确率仍然会有明显波动；batch_size 过大，单次迭代会忽略单个样本的特性，使输入样本模糊化，在 classification 任务中的体现为准确率下降。

3.8 momentum 对 MNIST Digit Classification 任务的影响

针对 1.4 中（单层 ReLU+Euclidean 损失函数）的网络，我做了三组实验，列于下表

momentum	0.1	0.5	0.9
Training_acc	0.9904	0.8986	**0.9970**
Training_loss	0.0143	0.0570	**0.0036**
Testing_acc	0.9797	0.8869	**0.9825**
Testing_loss	0.0232	0.0678	**0.0203**
运行时长	0:08:32.675	0:08:44.582	**0:10:14.125**

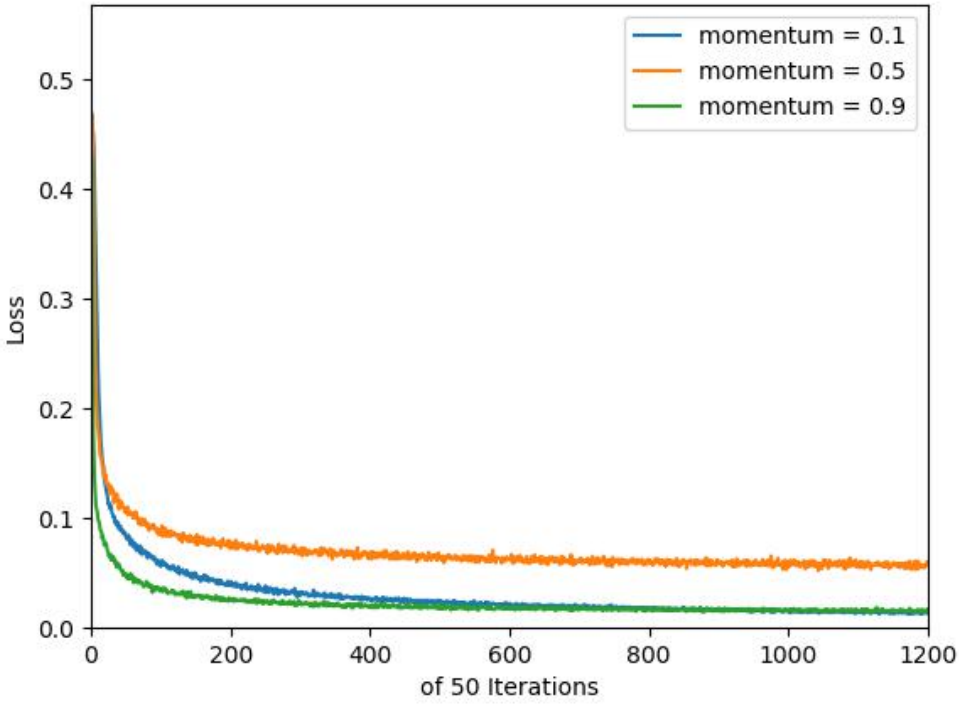

图 13　不同 momentum 单层 ReLU+Euclidean 损失函数网络 loss

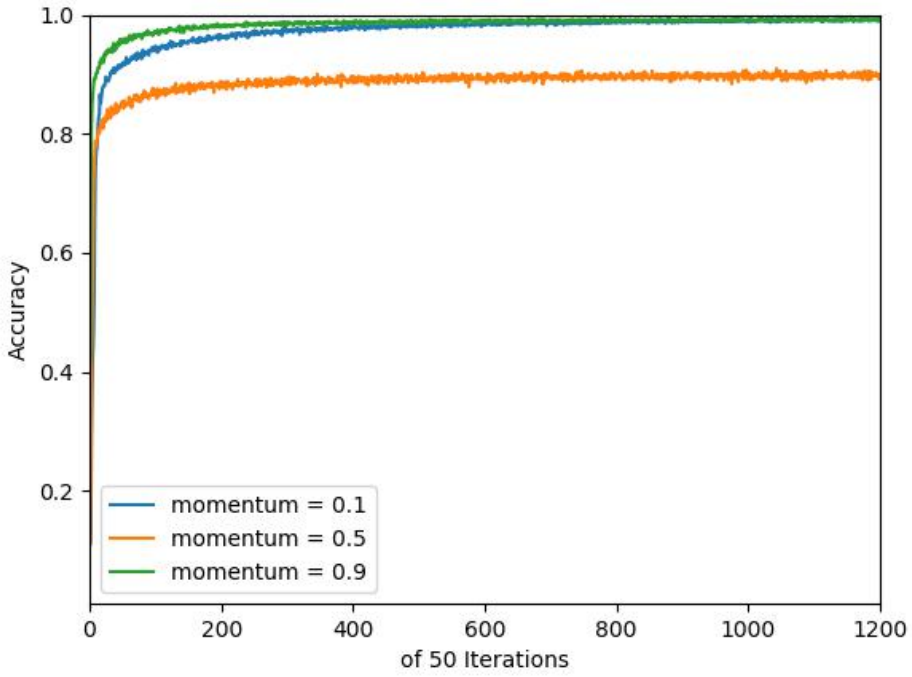

图 14　不同 momentum 单层 ReLU+Euclidean 损失函数网络准确率