人工神经网络
第一次作业
MNIST Digits Classification with CNN

计 55 张卡尔 2015011025

2017 年 10 月 25 日

实验内容

本实验将会使用卷积神经网络实现 MNIST 手写数字识别分类，对不同的网络结构以及不同的参数进行讨论，与 MLP 进行对比，探索优化方法及策略。

网络结构设计

本实验主要使用的网络结构如下：

- 输入层（$N * 28 * 28$）->
- 卷基层（$N * 4 * 28 * 28$，卷积核大小 $3 * 3$）->
- Relu 层
- 池化层($N * 4 * 14 * 14$)
- 卷基层（$N * 4 * 14 * 144$，卷积核大小 $3 * 3$）
- Relu 层
- 池化层($N * 4 * 7 * 7$)
- 全连接层($N * 196 * 10$)
- Softmax 交叉熵损失层

包含一个隐含层的全连接网络

在进行一层隐含层网络训练时，我分别对不同的隐含层神经元数量和激活函数对网络的影响进行了研究。

隐含层神经元数量

我使用同样的以 Relu 为激励函数的包含一层隐含层的全连接网络结构，分别在隐含层中使用 500 个神经元和 300 个神经元训练，其结果如下图，其中 300 个隐含神经元网络的最终准确率为 92.12%，500 个隐含神经元网络的最终准确率为 98.36%。(注：为使数据更有可观性，本文的所有图中的数据均为每 20iteration 输出一次。)

经测试发现，隐含层为 500 个神经元网络的收敛速度较快，精准度也更好，但在时间上，隐含层为 500 个神经元的网络训练时间约为 44 分钟，300 个神经元的网络训练时间为 24 分钟，可见，提高全连接神经网络隐含层神经元的数量可以提高收敛速度和准确度，但在训练时间上会付出更多的代价。

激活函数

使用相同参数、相同结构的 $784 * 500 * 10$ 的网络，分别使用 Relu 和 Sigmod 两种激活函数进行训练，对比结果如下。其中 Sigmoid 网络的在测试集上的准确率为 93.7%，Relu 网络则为 98.36%。

通过对比，可以发现在使用相同的参数和网络结构对同一任务进行训练时，在训练相同数量的 epoch 的情况下，Relu 比 Sigmoid 收敛速度更快且最终精确度更高，损失值更小。分析其原因，我认为是 Sigmoid 函数本身的特点导致的。Sigmoid 函数在中间部分的梯度较高，在两侧梯度很低，导致训练到一定程度时，梯度越来越小，从而训练速度十分缓慢。而 Relu 函数在 $[0, \infty)$ 上的梯度为常数，不会出现梯度消失的问题，因而收敛速度更快，且能够在更少数量的 epoch 上达到比较高水平的精准度。

在训练时间方面，Sigmoid 网络花费了 $51'53''$，Relu 网络花费了 $57'26''$，相差不大。

包含两个隐含层的全连接网络

与包含一个隐含层的全连接网络对比

使用相同参数的一层隐含层的 $785*500*10$ 的 Relu 网络和两层隐含层 $748*500*200*10$ 网络的训练结果如下，其中一层隐含层网络在测试集上的准确率为 98.36%, 两层隐含层的 Relu 网络的准确率为 98.52%。

经过对比，可以发现两层网络相比一层网络收敛速度低一点，但是最终准确率更高。究其原因，是因为多层网络在梯度传递时会越来越小，因此收敛速度比单层网络低但仍然速度很快。多了一层以后非线性度更好，因此最终准确率更高。

在训练时间上，一层 Relu 网络训练时间为 $57'26''$，两层 Relu 网络的训练时间为 $56'24''$，和预期不太相符，可能是因为我同时训练两个网络导致运算时间并不准确。

Relu 与 Sigmoid 的对比

使用相同参数、相同结构的 $784 * 500 * 200 * 10$ 的网络，分别使用 Relu 和 Sigmod 两种激活函数进行训练，发现在权重初始方差在 0.01 时 Relu 网络可以到达 98.52% 的准确率，而 Sigmoid 网络则很难进行有效训练，准确率在 10% 左右。可能是因为在多层网络 Sigmoid 网络下，梯度更容易消失，因此提高权重初始方位到 0.1，发现网络可以正常训练，最终准确率为 94.73%

可以看出两层的 Sigmoid 网络比 Relu 网络收敛更快，但在快速收敛以后训练速度及其缓慢，最终准确率也相对较低。分析原因，由于本实验存在两个变量，因此无法做到严格的对照试验。Sigmoid 网络收敛快有可能是高初始权重方差所致，这一点有待继续研究。另一方面，Relu 在后期训练速度上表现良好，可能是因为其梯度为常数，不会出现梯度消失的问题。

综上所述，通过两次对照试验，可以得出结论，在使用全连接网络解决 MNIST 手写数字识别问题上，Relu 激活函数因为更不容易出现梯度消失的现象，因而明显比 Sigmoid 更优。

优化过程及参数选择

Learning Rate

我尝试对一层 Relu 网络进行优化。分别使 learning rate 为 0.1 和 0.01 训练，得到结果如下图。其中 lr=0.1 的网络最终准确率为 98.36%，lr=0.01 的网络最终准确率为 97.31%。

可以看出，lr=1 的网络在收敛速度和最终准确度上都优于 lr=0.01 的网络，可能是 lr=0.01 的学习速率太慢了。

我们的目标是，在训练初期有较高的训练速度，而当训练接近收敛时，减小学习速率

和在高精准点周围的震荡幅度，提高精准性。因此，可以在训练初期使用较高的 learning rate，当训练接近收敛时，使用较低的 learning rate，提高训练的精准度。

我使用的具体方法是，将 learning rate 初始值设置为 0.1，在每个 epoch 训练完成之后，都计算当前网络对测试集的损失值，若损失值下降，则证明训练成功，learning rate 保持不变；若损失值增加，则说明当前 learning rate 过高，训练状态在目标周围震荡，因此将 learning rate 变为原来的一半，继续训练。为了避免 learning rate 连续多次更新而错过合适值，每次 learning rate 更新后，10 次 epoch 都不允许 learning rate 再更新。这样优化以后，可以保证在训练初期有较大的 learning，而训练后期 learning rate 有足够小。

然而不幸的是在我对这种优化算法进行测试的时候，其他参数没有调好，导致之后发现更好的参数的时候已经没有时间对这种算法进行测试了。

Momentum

Momentum 的定义是在每次计算出梯度的基础上再加一部分先前梯度值，使得当梯度连续向某一方向变化时，加快梯度变化的速度，从而提高收敛的速度。在接近目标值时，若出现反方向的变化，则说明出现了震荡，因此提高 momentum 会减弱梯度的震荡，更快地接近目标值。

本次实验中将 momentum 的值分别设为 0.1 和 0.9,对含有一个隐含层的 Relu 网络训练，如图所示。

其他参数

由于本次实验时间有限，之前几个参数的调整过程占用时间过长，其他参数的优化问题在本次实验中没有深入研究。batch_size 使用的是常规的 100，weight_decay 由于在其他参数尚未优化之前效果不明显，故暂取为 0。

实验总结

本次实验占用了我很多的时间，收获也是十分丰富的。我走了许多弯路，例如在一开始在寻找正常的高准确度上浪费了许多时间，由于有多个参数未调，因此有时候会同时修改了多个参数，导致找到了一个相对较优的参数就开始进行优化算法的实现。但其实后来才发现把 learning rate 调大以后甚至比现在做过优化的算法效果还

好。总而言之，通过这次实验我真实体会到了网络结构、激励函数、参数这些对网络效果的影响。只有真正理解了每一个数值的具体含义和其对网络的影响，才能进行合理的调参。