数据结果记录与分析

张鑫 1601110066

一．激活函数的影响

基础模型为无隐藏层，训练 5000 个 batch，batch_size = 100，初始化方式为零值初始化，损失函数为交叉熵，优化方法为梯度下降法，学习率为 0.01。由于最终输出必须在[0,1]，所以对 tanh()和 softsign()我们将其值域线性变换到[0,1]，即 f(x)=(x+1)/2.

得到测试集的表现如下：（取三次测试中间值）

激活函数	SOFTMAX	SIGMOID	TANH	SOFTSIGN
准确率	0.923	0.630	0.572	0.793

可以看出 SOFTMAX 表现是最好的

二．层数、单元数与初始化的影响

分别对一层隐藏层和两层隐藏层进行分析。基础模型为 5000 个 batch，batch_size 为 100，输出函数为 softmax 函数，隐藏层激活函数用 ReLU. 用交叉熵和梯度下降法，学习率为 0.01。对隐藏层节点个数和初始化方式的对比见下表：

ReLU 隐藏层单元数	零初始化	随机初始化
100	0.103	0.9751
300	0.098	0.9807

其中接近 0.1 的准确率相当于是随机猜中的概率，意味着毫无预测力。此外通过对初始化方式的比较可知，对 ReLU 而言随机初始化是很重要的，能够避免很多死节点。节点增加并没显著提高准确率。

一个自然的问题就是对于其他激活函数是否也有这种现象？我们把隐藏层激活函数换为 softmax 的话：

SOFTMAX 隐藏层单元数	零初始化	随机初始化
100	0.114	0.8762
300	0.101	0.837

对比之下，我们发现随机初始化确实很有必要。但跟奇怪的是无隐藏层的时候零初始化并没有什么印象，推测这主要是因为末端的梯度可以直接传递在权重矩阵和偏至上。

此外，我们可以发现两层的准确率比一层有显著提高。我们进一步尝试三层的 MLP，节点数采用 784 -> 300 -> 100 -> 10 的结构。用 ReLU 链接，随机初始化。很遗憾，这个网络下预测准确率仅为 0.098，对初始化随机方式进行了调整仍旧为这个结果。微调网络结构，改为 784 -> 100 -> 50 -> 10 的方式仍旧是这个结果，可见网络深度并非越深越好。

三．优化方法和学习率的影响

我们在前面表现最好的图上改变优化算法进行分析。选取一个 ReLU 隐藏层的 MLP，随机初始化。

	0.005	0.01	0.05
GradientDescent	0.980	0.980	0.098
Adagrad	0.967	0.975	0.098
Adam	0.098	0.098	0.098
Adadelta	0.761	0.8451	0.9215
Momentum(2 阶)	0.098	0.098	0.098

从上表可以看出，不同优化算法在个别问题上表现还是有差异的，学习率的影响也很显著，整体而言不应该过大，胆小了也不见得就好，见 Adadelta. 大致上可以认为 ReLU 配合梯度下降法是个不错的搭配。

四．Autoencoder 预训练

我们尝试在模型中加入 Autoencoder 进行预训练以使得参数初始状态处在较好的位置。计算图结构为 700 -> 300 ->10. 代码里在预训练和全体训练的时候都有打印第一个隐藏层的偏至，以示监视 Autoencoder 有在起作用。但是体现在最后的预测率上，有没有进行 Autoencoder 预训练并没有显著差别，均为 0.98 附近。这可能是因为我们的神经网络层数较低。