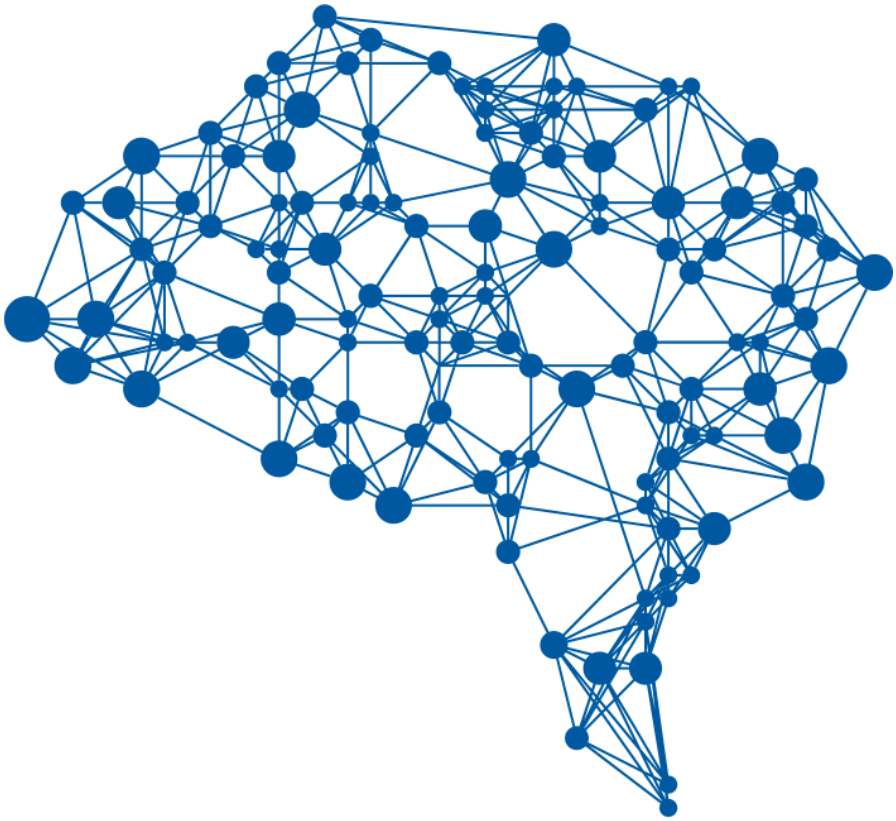
Deep Learning Homework 1



安捷 1601210097

2017 年 3 月 23 日

1 考虑跨层连接权值的多层 BP 算法推导

由于这一部分公式较多，且为了可以用图示表示清楚推导过程，我用 ipad 手写了这一部分的推导，本部分内容请看报告最后的附录。

2 基于 tensorflow 的多层感知机（MLP）算法实现

我使用 python 接口的 tensorflow 库实现了包含一个隐含层的 MLP 算法，现就算法实现的细节介绍如下：

2.1 参数设置

参数名称	参数取值	参数作用
LAYER_NUM	3	MLP 层数
HIDDEN_NEURON_NUM	500	隐含层神经元数目
TRAIN_SAMPLE_NUM	55000	训练数据个数
TEST_SAMPLE_NUM	10000	测试数据个数
NOLINEAR_FUNCTION	ReLU	非线性函数类型
OPTIMIZER	GradientDescent	优化函数类型
BATCH_NUM	128	batch 大小
TRAIN_CYCLE	100	训练次数

表 1: 算法参数表

2.2 分类准确率

经过调参，算法在上述参数表列出的参数下达到准确度：98.02%，重复测试的准确度均为 98% 且不随迭代次数增加而增加，同时，增加或减少隐含层节点数目不会提高准确度，可见上述参数设置没有过拟合与欠拟合的情况发生。

2.3 代码运行环境及测试平台信息

在没有 NVIDIA GPU 及 CUDA 支持的环境下代码依然可以运行，只是速度较慢

Python Version: 3.6.0
Tensorflow Version: tensorflow-gpu-1.0.1
CUDA Version: 8.0 OS: Arch Linux
Kernel: x86_64 Linux 4.10.4-1-ARCH
CPU: Intel Core i7-6700K @ 8x 4.2GHz
GPU: GeForce GTX 1060 6GB
RAM: 16003MiB

表 2: 代码运行环境及测试环境表

3 总结

通过这次作业，我学习了 tensorflow 的基本使用方法，在实现了 MLP 之后，我又实现了 CNN 并成功部署到了科研实验中，接下来我将尝试用 tensorflow 实现 FCN 用于图像分割；在完成作业的过程中，我同时利用上述测试环境中的 CPU 与 GPU 进行了训练，发现 GPU 训练的速度大约是 CPU 训练速度的两倍，之后的作业在显存足够的情况下我将主要使用 GPU 进行训练。

4 附录