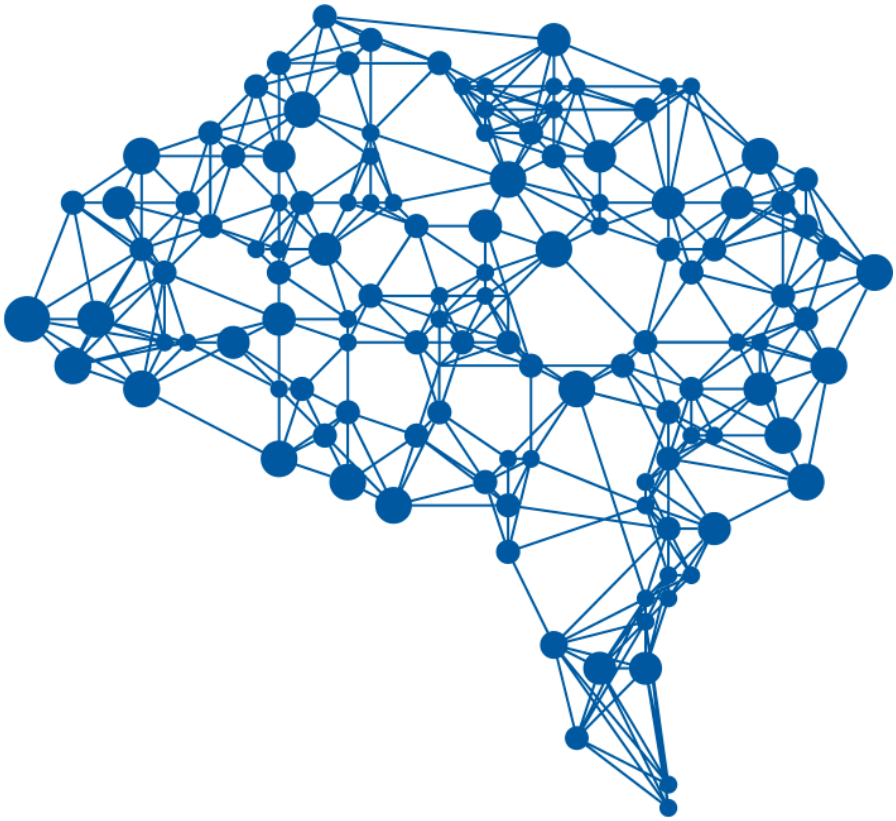
Deep Learning Homework 1



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

1. K层前馈网络BP算法推导(考虑跨层连接权值)

考虑如下K层前馈网络

$y_1 \quad y_2 \quad \cdots \quad y_e$ m个

$x_{K-1,1} \quad x_{K-1,2} \quad \cdots \quad x_{K-1,n_{K-1}}$ n_{K-1}个

$\vdots$ $\vdots$

$x_{2,1}, x_{2,2}, \cdots, x_{2,n_2}$ n_2个

$x_{1,1}, x_{1,2}, \cdots, x_{1,n_1}$ $n_1=n$个

其中 $x_{1,i}$ 表示网络的输入 x_i，为了推导方便写为 $x_{1,i}$，其中相邻层的每两个结点一定有边相连，不同层的结点也可以有边相连.

STEP 1 各层广义误差 δ 的推导(考虑一个输入点)

$$E = \frac{1}{2}\sum_{e=1}^{m}(t_e - y_e)^2$$

$$\delta_e^K = (t_e - y_e)\cdot y_e(1-y_e)$$

由三层BP算法的推导得

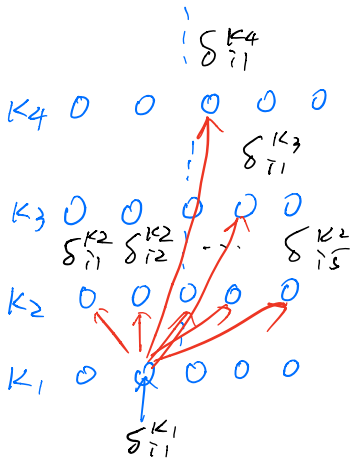
$$\delta_i^{K-1} = \sum_{e=1}^{m}\delta_e^K \cdot w_{K-1,e}^{K,i}$$

其中 δ_i^t 表示 t 层第 i 个的广义误差

$w_{a,i}^{b,j}$ 表示 a 层 i 个到 b 层 j 个的误差

则有 $\delta_i^t = \sum_{p=t+1}^{K} \cdot \sum_{q=1}^{n_p} \delta_q^p \cdot w_{t,i}^{p,q}$

如下图所示



则有 $\delta_{11}^{K1} = \sum_{K} \sum_{i} \delta_i^K$

STEP 2 对某一样本 u，$\frac{\partial E}{\partial w_{K1,i}^{K2,j}}$ 的推导

$$\frac{\partial E^u}{\partial w_{K1,i}^{K2,j}} = \frac{\partial E^u}{\partial x_j^{K2}} \cdot \frac{\partial x_j^{K2}}{\partial u_j^{K2}} \cdot \frac{\partial u_j^{K2}}{\partial x_i^{K1}}$$

$$= \delta_j^{K2} \cdot x_i^{K1}$$

则 $$\frac{\partial E}{\partial w_{K1,i}^{K2,j}} = \sum_{u=1}^{N} \delta_j^{K2,u} \cdot x_i^{K1,u}$$

STEP 3 综上，δ_i^K 的更新公式在考虑3跨层连接之后略有不同，其余与三层BP算法一致，可以使用三层BP算法学习，只需将第一步修改为：

对所有的 u，计算 $(x_i^k)_u$ 与 $(\delta_i^k)_u$，

其中 $\delta_i^k = \sum_{t \in K} \sum_{j=1}^{n_K} \delta_0^t \cdot w_{K,i}^{t,j}$

若这条边不存在，则对应的 $w=0$