

武汉大学
高级语言程序设计大作业报告

基于多层感知机网络的手写数字数据集分类
算法及其优化

院(系)名称: 计算机学院

专业名称: 计算机类

学生姓名: 张永康

学号: 2017301500335

指导教师: 谭成予 副教授

二〇一八年五月

摘 要

手写数字分类问题是机器学习与模式识别领域的基础问题之一，也是 OCR(光学字符识别) 的核心技术，具有重要的现实意义。本文针对 MNIST 手写数字数据集，通过 C++ 实现了基于多层感知机网络的手写数字数据集分类算法，并采用交叉验证、小批量随机梯度下降算法、扩充训练数据集的方法优化算法，最终达到了 98.042% 的分类准确率。

关键词: 多层感知机; 手写数字; 分类; 优化;

ABSTRACT

The classification of handwritten digits is one of the basic problems of machine learning and pattern recognition, which is also the key technology of optical character recognition(OCR). In this paper, according to the MNIST handwritten digits dataset, we implement a classifying algorithm based on multi-layer perceptron(MLP) by C++, and optimize it by cross-validation, mini-batch stochastic gradient descent algorithm and expanding the training dataset. The accuracy of our algorithm reaches 98.042%.

Key words: Multi-layer Perceptron; Handwritten Digits; Classification; Optimization;

目 录

| | |
|--------------------------------|----|
| 摘要 | I |
| ABSTRACT | II |
| 1 MNIST 手写数字数据集 | 1 |
| 2 MNIST 数据集的分类算法比较 | 2 |
| 3 基于多层感知机网络的分类算法 | 4 |
| 3.1 感知机 | 4 |
| 3.2 多层感知机网络 | 4 |
| 3.3 正向传播算法 | 5 |
| 3.3.1 输入数据的预处理 | 5 |
| 3.3.2 输入数据的正向传播 | 6 |
| 3.3.3 误差估计 | 6 |
| 3.4 反向传播算法 | 7 |
| 3.4.1 隐含层到输出层边权的权值更新 | 7 |
| 3.4.2 输出层偏置的权值更新 | 8 |
| 3.4.3 输入层到隐含层边权的权值更新 | 8 |
| 3.4.4 隐含层偏置的权值更新 | 9 |
| 4 分类算法的优化 | 10 |
| 4.1 交叉验证 | 10 |
| 4.2 小批量随机梯度下降算法 | 10 |
| 4.3 训练集扩充 | 11 |

| | |
|--------------------------------|-----------|
| 5 实验对比 | 12 |
| 6 总结 | 13 |
| 参考文献 | 14 |
| 附录 A 代码文件说明 | 15 |
| 附录 B 程序结构 | 16 |
| B.1 宏定义常量说明 | 16 |
| B.2 多层感知机网络结构体数据项说明 | 16 |
| B.3 多层感知机网络结构体函数接口说明 | 17 |
| B.4 程序运行效果 | 21 |

1 MNIST 手写数字数据集

MNIST(Mixed National Institute of Standards and Technology database) 是一个包含 70000 个 28×28 像素大小的手写阿拉伯数字灰度图片的大型数据集^[1]。MNIST 数据库被广泛用于测试机器学习算法的分类能力。

本文使用 Kaggle 在线大数据测试平台的 MNIST 测试系统^[2] 来检验分类算法的分类正确率，该系统将 MNIST 数据库中的 42000 个数字作为训练集，28000 个数字作为测试集。其中，每个图片被转换为 $28 \times 28 = 784$ 维的向量。训练集为 42000 行、784 列的矩阵，矩阵中第 i 行第 1 列代表第 i 张图片对应的数字，之后的 784 列分别代表第 i 张图片中每个像素点的灰度值。测试集训练集为 28000 行、784 列的矩阵，矩阵中第 i 行第 j 列代表第 i 张图片中像素点 j 的灰度值。

2 MNIST 数据集的分类算法比较

MNIST 数据集的分类算法有很多,如 K 近邻算法、支持向量机^[3]、集成学习方法、多层感知机网络、卷积神经网络等。目前分类准确率最高的是基于 DropConnect 的两层 CNN 的分类算法^[4],其准确率为 99.79%。

K 近邻算法 (K Nearest Neighbors,KNN) 是一种度量学习方法,它基于某种距离度量,在训练集中找到与测试对象最近的 K 个训练数据,并在这 K 个数据中选取出现频率最高的数据类别作为测试对象的所属类别,具有算法容易实现的特点。但其无训练过程,在对每个数据对象进行分类时需要遍历整个数据集,耗时过长。

支持向量机 (Support Vector Machine,SVM) 将数据映射到高维空间,使用核函数 (Kernel Function) 寻找合适的超平面划分数据集,与线性回归算法相比,SVM 可以划分原始空间中线性不可分的数据集,具有计算量小、不易过拟合等优点,但数学推导过程过于复杂。

集成学习方法 (如 Adaboost、随机森林) 是通过构建并结合多个分类器来完成训练、分类任务。集成学习通过一定的策略训练每个子分类器、结合每个子分类器的分类结果,常常能获得比单个子分类器更好的分类准确率,具有泛化能力强的优点,缺点是算法难以理解、实现。

多层感知机网络 (Multi Layer Perceptron,MLP) 是由若干层感知机神经元组成的全连接网络。其中每个神经元可以解决一个线性可分问题。MLP 通过多层神经元的共同作用,可以解决大多数的线性不可分问题,其结构简单,算法易于实现,缺点是训练速度略慢,无法提取图片中的空间关系,有时需要通过特征工程对输入数据进行特征提取。

卷积神经网络 (Convolutional Neural Network,CNN) 是由若干卷积层、下采样层组成的神经网络,通过卷积层的卷积核提取图片中的特征信息,具有学习图片中一定特征的能力,分类准确率高,缺点是结构复杂、反向传播算法难以实现、训练时间过长。

综合考虑算法实现难度、训练时间成本等因素,本文采用由两层感知机神经

元组成的多层感知机网络实现分类算法。

3 基于多层感知机网络的分类算法

3.1 感知机

感知机神经元 (Perceptron) 是多层感知机网络的基本组成单位, 由输入向量 X 、权重向量 W 、偏置 b 、激励函数 σ 和输出值组成。输入向量经权重向量加权求和后输入到感知机中, 然后经激励函数得到输出值。

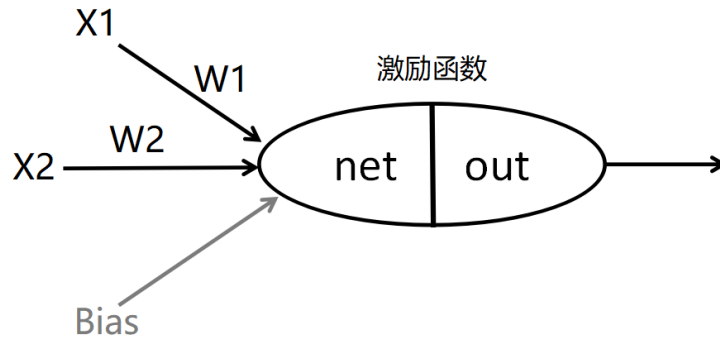


图 3.1 感知机神经元结构

感知机的结构如图 3.1所示。其中, 激励函数一般为非线性函数, 将输入值映射到一个较小区间中, 以提高感知机对非线性函数的拟合能力。激励函数有很多, 如 *Sigmoid*、*tanh*、*ReLU* 等。

本文采用最常见的 *Sigmoid* 函数作为感知机的激励函数。*Sigmoid* 函数及其导函数可表示为:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (3.1)$$

$$Sigmoid'(x) = Sigmoid(x)(1 - Sigmoid(x)) \quad (3.2)$$

3.2 多层感知机网络

单个感知机已被证明不能解决线性不可分问题 (如异或问题), 但多层感知机网络可以解决大多数此类问题。本文采用由两层感知机神经元组成的多层感知机

网络，该网络包含输入层、隐含层与输出层。

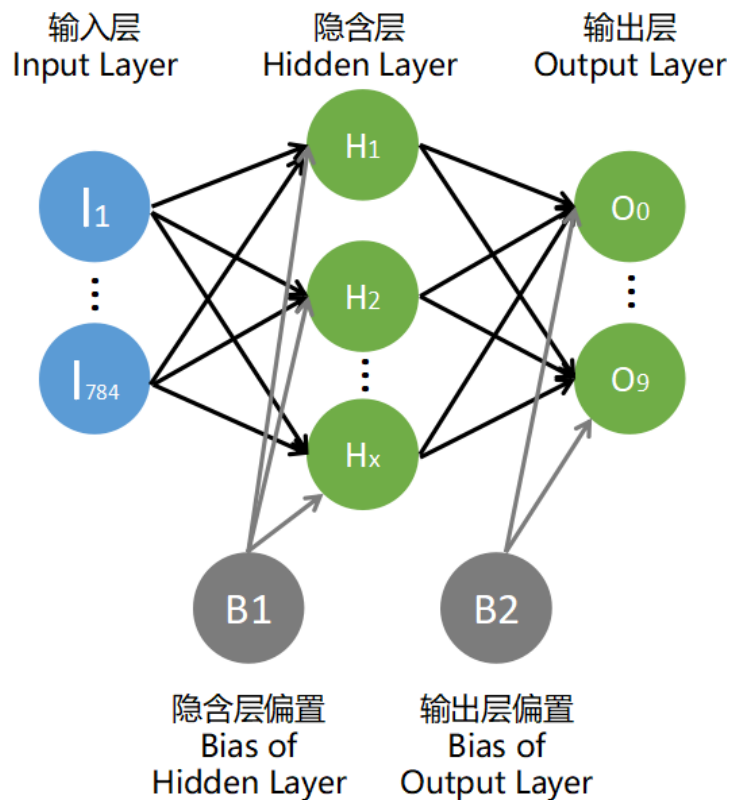


图 3.2 多层感知机网络结构

该网络结构如图3.2所示。其中输入层由 784 个结点构成，分别代表经归一化处理后的输入数据的每个像素点的灰度值，隐含层由 n_h 个感知机神经元组成，输出层由 10 个感知机神经元组成，它们的输出分别代表输入图片为数字 0 到 9 的概率。

3.3 正向传播算法

3.3.1 输入数据的预处理

为了提高训练的收敛速率、最终的分类准确率，需要对输入数据进行归一化 (Normalization) 处理，使得输入数据由较大的区间 $[0, 255]$ 映射到区间 $[0, 1]$ 上。本文采用 minmax 归一化方法。

设输入数据的 784 维中灰度值最小值为 min ，最大值为 max ，则该数据的第

i 维灰度值归一化为:

$$input_i \leftarrow \frac{input_i - min}{max - min} \quad (3.3)$$

3.3.2 输入数据的正向传播

输入数据通过隐含层、输出层正向传播后得到输出结果。

对于第 i 个隐含层神经元 h_i ，它接受来自输入层的 784 个输入值 $input_j$ ，对它们加权求和并加上偏置 b_h 得到 net_{h_i} ：

$$net_{h_i} = \sum_{j=1}^{784} w_j input_j + b_h \quad (3.4)$$

然后经激励函数处理后得到隐含层神经元的输出值 out_{h_i} ：

$$out_{h_i} = \sigma(net_{h_i}) \quad (3.5)$$

对于第 i 个输出层神经元 h_i ，它接受来自隐含层的 n_h 个输入值 out_{h_j} ，对它们加权求和并加上偏置 b_o 得到 net_{o_i} ：

$$net_{o_i} = \sum_{j=1}^{n_h} w_j out_{h_j} + b_h \quad (3.6)$$

然后经激励函数处理后得到输出层神经元的输出值 out_{o_i} ：

$$out_{o_i} = \sigma(net_{o_i}) \quad (3.7)$$

3.3.3 误差估计

在训练过程中，输入数据经正向传播得到输出数据后，需要通过误差估计，将其与期望得到的输出数据作比对。

误差估计的方法有很多，常见的有均方误差 (Mean Squared Error,MSE)、交叉熵误差 (Cross Entropy Error,CEE)。本文采用 MSE 作为误差估计方法。

对于输出层的第 i 个神经元，其输出 out_{o_i} 与期望输出 $output_i$ 之间的 MSE 为：

$$E_i = \frac{1}{2}(out_{o_i} - output_i)^2 \quad (3.8)$$

则总的 MSE 可表示为:

$$E = \sum_{i=0}^9 E_i \quad (3.9)$$

3.4 反向传播算法

多层感知机网络中包含很多边权与偏置权重, 因此需要高效的算法来调整这些权重, 使得网络的 MSE 尽可能小, 这一过程称之为训练 (Train)。

训练算法有很多, 如反向传播算法 (Error Backpropagation Algorithm, BP)、神经进化算法 (Neuro Evolution) 等, 本文采用最经典的反向传播算法。

BP 算法的本质是对于每个权重, 求出总 MSE 对该权重的偏导数, 并根据偏导数将该权重沿使 MSE 减小的方向调整。

两层结构的多层感知机网络的 BP 算法具体如下^[5]。

3.4.1 隐含层到输出层边权的权值更新

设隐含层神经元 h_i 到输出层神经元 o_j 的边权为 $w_{i,j}$, 则总的 MSE 对其的偏导数为:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial out_{o_j}} \frac{\partial out_{o_j}}{\partial net_{o_j}} \frac{\partial net_{o_j}}{\partial w_{i,j}} \quad (3.10)$$

其中,

$$\frac{\partial E}{\partial out_{o_j}} = out_{o_j} - output_j \quad (3.11)$$

$$\frac{\partial out_{o_j}}{\partial net_{o_j}} = \sigma'(net_{o_j}) = out_{o_j}(1 - out_{o_j}) \quad (3.12)$$

$$\frac{\partial net_{o_j}}{\partial w_{i,j}} = out_{h_i} \quad (3.13)$$

将式3.11、3.12、3.13代入式3.10:

$$\frac{\partial E}{\partial w_{i,j}} = (out_{o_j} - output_j) out_{o_j}(1 - out_{o_j}) out_{h_i} \quad (3.14)$$

定义隐含层到输出层边权的学习率为 $\eta_{h,o}$ ，则 $w_{i,j}$ 的更新公式为：

$$w_{i,j} \leftarrow w_{i,j} - \eta_{h,o} \frac{\partial E}{\partial w_{i,j}} \quad (3.15)$$

3.4.2 输出层偏置的权值更新

设输出层偏置为 b_o ，则总的 MSE 对其的偏导数为：

$$\frac{\partial E}{\partial b_o} = \frac{\partial E}{\partial out_{o_j}} \frac{\partial out_{o_j}}{\partial net_{o_j}} \frac{\partial net_{o_j}}{\partial b_o} \quad (3.16)$$

其中，

$$\frac{\partial E}{\partial b_o} = 1 \quad (3.17)$$

将式3.11、3.12、3.17代入式3.16：

$$\frac{\partial E}{\partial b_o} = (out_{o_j} - output_j) out_{o_j} (1 - out_{o_j}) \quad (3.18)$$

定义输出层偏置权值的学习率为 η_{b_o} ，则 b_o 的更新公式为：

$$b_o \leftarrow b_o - \eta_{b_o} \frac{\partial E}{\partial b_o} \quad (3.19)$$

3.4.3 输入层到隐含层边权的权值更新

设输入层神经元 i_i 到隐含层神经元 h_j 的边权为 $w_{i,j}$ ，则总的 MSE 对其的偏导数为：

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial out_{h_j}} \frac{\partial out_{h_j}}{\partial net_{h_j}} \frac{\partial net_{h_j}}{\partial w_{i,j}} \quad (3.20)$$

其中，

$$\begin{aligned} \frac{\partial E}{\partial out_{h_j}} &= \sum_{k=0}^9 \frac{\partial E}{\partial out_{o_k}} \frac{\partial out_{o_k}}{\partial net_{o_k}} \frac{\partial net_{o_k}}{\partial out_{h_j}} \\ &= \sum_{k=0}^9 (out_{o_k} - output_k) out_{o_k} (1 - out_{o_k}) w_{h_j, o_k} \end{aligned} \quad (3.21)$$

$$\frac{\partial out_{h_j}}{\partial net_{h_j}} = out_{h_j}(1 - out_{h_j}) \quad (3.22)$$

$$\frac{net_{h_j}}{w_{i,j}} = input_i \quad (3.23)$$

定义输入层到隐含层边权权值的学习率为 $\eta_{i,h}$ ，则 $w_{i,j}$ 的更新公式为：

$$w_{i,j} \leftarrow w_{i,j} - \eta_{i,h} \frac{\partial E}{\partial w_{i,j}} \quad (3.24)$$

3.4.4 隐含层偏置的权值更新

设隐含层偏置为 b_h ，则总的 MSE 对其的偏导数为：

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial out_{h_j}} \frac{\partial out_{h_j}}{\partial net_{h_j}} \frac{\partial net_{h_j}}{\partial b_h} \quad (3.25)$$

其中，

$$\frac{\partial net_{h_j}}{\partial b_h} = 1 \quad (3.26)$$

定义隐含层偏置权值的学习率为 η_{b_h} ，则 b_h 的更新公式为：

$$b_h \leftarrow b_h - \eta_{b_h} \frac{\partial E}{\partial b_h} \quad (3.27)$$

4 分类算法的优化

4.1 交叉验证

神经网络的训练过程迭代次数过多，容易导致过拟合。为了避免过拟合的产生，本文采用交叉验证 (Cross Validation) 的方法，即选取训练数据中的 2000 组数据作为验证集，其余数据作为训练集，使用训练集训练神经网络，每隔一段时间使用验证集模拟测试数据，输入神经网络，并估计训练误差。

4.2 小批量随机梯度下降算法

小批量随机梯度下降算法 (Mini-batch Stochastic Gradient Descent, Mini-batch SGD) 是梯度下降算法的一种，结合了批量梯度下降算法和随机梯度下降算法的优点。

批量梯度下降算法 (Batch Gradient Descent, BGD) 每次更新权值时需要将全部训练数据放入神经网络，通过反向传播计算出每个权值对于所有数据的梯度之和。

随机梯度下降算法 (Stochastic Gradient Descent, SGD) 每次更新权值时只随机抽取一个训练数据放入神经网络，通过反向传播计算出每个权值对于该数据的梯度。

一般来讲，BGD 往往能更稳定地朝损失值减小的方向更新权值，但每次迭代时计算量过大，训练速度慢；SGD 每次迭代时计算量小，训练速度快，但训练过程中损失值波动幅度大，极不稳定。

Mini-batch SGD 每次从所有训练数据中有序地抽取 mini-batch 个训练样本作为一个训练批次放入神经网络，通过反向传播计算出每个权值对于所有训练样本的梯度之和，具有训练速度快、损失值下降相对稳定的特点。

4.3 训练集扩充

由于本次实验中采用的神经网络模型参数多，训练数据集过少时，神经网络容易过拟合。因此本文采用扩展训练数据集的方式提高最终分类的准确率。扩展训练数据集的方法有很多，如对数字图片添加噪声、仿射变换 (Affine Distortion)、弹性变换 (Elastic Distortion)^[6] 等。本文采用弹性变换的方法。数字 0 的图片经弹性变换后产生的新图片如图4.1所示。

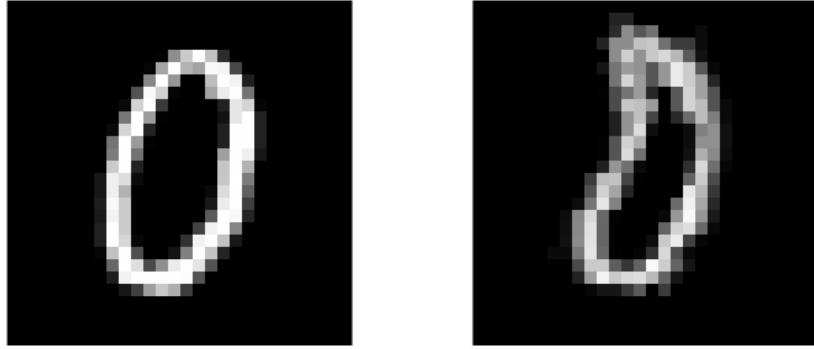


图 4.1 数字 0 的图片经弹性变换后产生的新图片

弹性变换的具体算法如下：

1. 随机生成矩阵 $x[28,28], y[28,28]$ ，矩阵中每个元素的值均在 $[-1,1]$ 之间
2. 使用大小为 n ，标准差为 σ 的高斯核对 x 、 y 进行滤波处理
3. 新图片中每个像素点的像素值按以下公式计算：

$$I'(i, j) = \frac{I(\lfloor x_{i,j} \rfloor, \lfloor y_{i,j} \rfloor) + I(\lfloor x_{i,j} \rfloor, \lceil y_{i,j} \rceil) + I(\lceil x_{i,j} \rceil, \lfloor y_{i,j} \rfloor) + I(\lceil x_{i,j} \rceil, \lceil y_{i,j} \rceil)}{4} \quad (4.1)$$

5 实验对比

本文的分类算法在 Windows 10、i7-8550U、8G 内存的环境下进行测试实验，实验结果如表5.1所示。

表 5.1 不同分类算法在 MNIST 数据集上的准确率

| 算法 | 准 确 率 (%) | 备 注 | 来 源 |
|-----------------------------|--------------|---|-----|
| 2-Layer MLP | 93.085 | $\eta=0.01, n_h=30, 4000$ 次迭代 | 本文 |
| 2-Layer MLP | 96.542 | $\eta=0.01, n_h=300, \text{batch-size}=100, 50$ epoch | 本文 |
| 2-Layer MLP | 96.957 | $\eta=0.01, n_h=300, \text{batch-size}=100, 25$ epoch, 扩充数据至 84000 组 | 本文 |
| 2-Layer MLP | 97.485 | $\eta=0.01, n_h=300, \text{batch-size}=100, 25$ epoch, 扩充数据至 126000 组 | 本文 |
| 2-Layer MLP | 98.042 | $\eta=0.01, n_h=300, \text{batch-size}=100, 35$ epoch, 扩充数据至 126000 组 | 本文 |
| LeNet-5 | 99.2 | | [7] |
| Capsule Network | 99.75 | | [8] |
| 2-Layer CNN, DropConnect | 99.79 | | [4] |

6 总结

本文使用 C++ 实现了基于多层感知机网络的手写数字数据集分类算法，并使用交叉验证、Mini-batch SGD、扩充训练数据集的方式，最终在 Kaggle 平台上达到了 98.042% 的分类准确率，这一结果距离目前最高水平仍有较大差距，原因如下：

1. 本文采用的多层感知机网络与目前主流的 CNN 相比，无法自动地学习图片中的空间特征，其准确率受图片中数字的缩放、旋转、位移等操作的影响。
2. 受计算机计算能力的限制，没有对训练数据做进一步的扩充。
3. 神经网络的训练过程是一个在非凸函数的最优化问题，梯度下降算法易陷入局部最优点。
4. 传统的梯度下降算法对参数依赖大，不能准确、高效地找到最优点，可以采用随机对角 LM 算法 (Stochastic Diagonal Levenberg-Marquardt) 替代。
5. 损失函数为均方误差函数，可以使用带 Softmax 激励函数的输出层、交叉熵误差函数替代。

参考文献

- [1] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits, 1998.
- [2] kaggle. Digit recognizer, 3/18 2018.
- [3] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [4] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.
- [5] 周志华. 机器学习. 清华大学出版社, 2016.
- [6] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition, 2003. Proceedings*, International Conference on Document Analysis and Recognition, 2003. Proceedings, page 958, 2003.
- [7] Yann Lécun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [8] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.

附录 A 代码文件说明

大作业由以下四部分代码组成

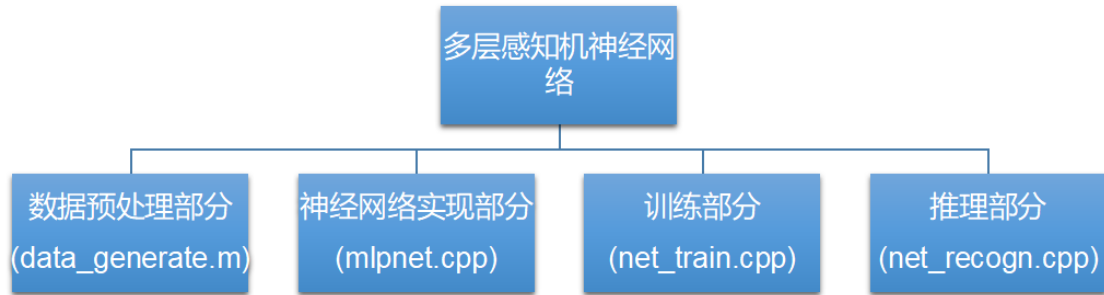


图 A.1 大作业代码组成图示

另外,大作业中还包含经弹性变换扩充至原数据量三倍的训练数据 (train_ext2.txt)、Kaggle 平台上的 MNIST 测试集数据 (test.txt)、本网络对测试数据集的分类结果 (label.csv)、已经训练好的神经网络参数文件 (net_data_ver6.txt)

附录 B 程序结构

B.1 宏定义常量说明

```
#define INPUT_MAXNUM 800 // 输入层神经元个数上限
#define HIDDEN_MAXNUM 320 // 隐含层神经元个数上限
#define OUTPUT_MAXNUM 10 // 输出层神经元个数上限

#define SAMPLE_MAXNUM 130000 // 训练样例数量
#define CHECK_MAXNUM 5000 // 测试样例数量
```

B.2 多层感知机网络结构体数据项说明

多层感知机网络在代码中被封装为如下的结构体：

```
struct NeuralNetwork
```

结构体数据项：

1. input_num,hidden_num,output_num: 输入层、隐含层、输出层神经元个数
2. net_error: 当前神经网络输出与预期输出之间的均方差误差
3. RATE_INPUT2HIDDEN,RATE_HIDDEN2OUTPUT,RATE_BIAS_HIDDEN, RATE_BIAS_OUTPUT: 网络参数, 分别为输入层到隐含层学习率, 隐含层到输出层学习率, 隐含层偏置学习率, 输出层偏置学习率
4. input[]: 输入层神经元的值
5. hidden_net[]: 隐含层神经元 Net 值
6. hidden_out[]: 隐含层神经元 Out 值, Out=Sigmoid(Net)
7. output_net[]: 输出层神经元 Net 值
8. output_out[]: 输出层神经元 Out 值
9. output_expectation[]: 期望输出值
10. sumg_w_input2hidden[][]: 输入层神经元到隐含层神经元的每条边的权重梯度

之和

- 11. sumg_w_hidden2output[]: 隐含层神经元到输出层神经元的每条边的权重梯度之和
- 12. w_input2hidden[]: 输入层神经元到隐含层神经元的边的权重
- 13. w_hidden2output[]: 隐含层神经元到输出层神经元的边的权重
- 14. bias_hidden: 隐含层偏置值
- 15. bias_output: 输出层偏置值
- 16. sumg_bias_hidden: 隐含层偏置值梯度之和
- 17. sumg_bias_output: 输出层偏置值梯度之和

B.3 多层感知机网络结构体函数接口说明

```
void network_init()
```

函数功能: 对神经网络权值 (w_input2hidden[], w_hidden2output[], bias_hidden, bias_output) 进行初始化

入口参数: 无

出口参数: 无

函数返回值: 无

```
void save_net(string file_dir)
```

函数功能: 保存已训练好的神经网络的参数

输出文件格式:

input_num, hidden_num, output_num

net_error, RATE_INPUT2HIDDEN, RATE_HIDDEN2OUTPUT,

RATE_BIAS_HIDDEN, RATE_BIAS_OUTPUT

w_input2hidden

w_hidden2output

bias_hidden, bias_output

入口参数: file_dir(输出文件目录及文件名)

出口参数: 无

函数返回值: 无

```
void load_net(string file_dir)
```

函数功能: 读入一个训练好的神经网络的各参数

输入文件格式:

input_num,hidden_num,output_num

net_error,RATE_INPUT2HIDDEN,RATE_HIDDEN2OUTPUT,

RATE_BIAS_HIDDEN,RATE_BIAS_OUTPUT

w_input2hidden

w_hidden2output

bias_hidden,bias_output

入口参数: file_dir(输入文件目录及文件名)

出口参数: 无

函数返回值: 无

```
void forward_propagation()
```

函数功能: 从输入数据正向传播得到神经网络的预测输出

入口参数: input[] (输入数据)

出口参数: output_out[] (神经网络的预测输出)

函数返回值: 无

```
void backward_propagation()
```

函数功能: 根据神经网络的预测输出和训练数据的真实输出之间的误差,

反向传播得到神经网络各参数的梯度, 并将每个参数的梯度累加到

该参数在当前训练批次下的梯度之和 sumg 中。

入口参数: output_out[](神经网络的预测输出),output_expectation[](训练数据的真实输出)

出口参数:

sumg_w_input2hidden[][]: 输入层神经元到隐含层神经元的每条边的权重梯度之和

sumg_w_hidden2output[][]: 隐含层神经元到输出层神经元的每条边的权重梯度之和

sumg_bias_hidden: 隐含层偏置值梯度之和

sumg_bias_output: 输出层偏置值梯度之和

函数返回值: 无

```
void update_weight()
```

函数功能: 根据当前训练批次下每个参数的梯度总和 sumg, 更新每个参数。

入口参数:

sumg_w_input2hidden[][](输入层神经元到隐含层神经元的每条边的权重梯度之和)

sumg_w_hidden2output[][](隐含层神经元到输出层神经元的每条边的权重梯度之和)

sumg_bias_hidden(隐含层偏置值梯度之和)

sumg_bias_output(输出层偏置值梯度之和)

出口参数:

w_input2hidden[][](输入层神经元到隐含层神经元的边的权重)

w_hidden2output[][](隐含层神经元到输出层神经元的边的权重)

bias_hidden(隐含层偏置值)

bias_output(输出层偏置值)

函数返回值: 无

```
void get_error(int check_num,double check_input[][],  
double check_output[][])
```


函数功能：输入一批验证数据，使用均方误差估计神经网络对于每个训练数据的预测输出与真实输出之间的误差，并输出所有验证数据的误差之和。

入口参数：

check_num(验证数据个数)

check_input[][](验证数据的输入)

check_output[][](验证数据的真实输出)

出口参数：

net_error(当前神经网络输出与预期输出之间的均方差误差)

函数返回值：无

```
void train(int sample_num, double sample_input[][],  
double sample_output[][])
```

函数功能：读入一批训练数据，使用这些训练数据更新一次整个网络的权值。

入口参数：

sample_num(训练数据的个数)

sample_input[][](训练数据的输入)

sample_output[][](训练数据的真实输出)

出口参数：无

函数返回值：无

```
void recognize(double data_input[], double data_output[])
```

函数功能：读入一个测试数据的输入，输出神经网络的预测结果

入口参数：

data_input[][](测试数据的输入)

出口参数：

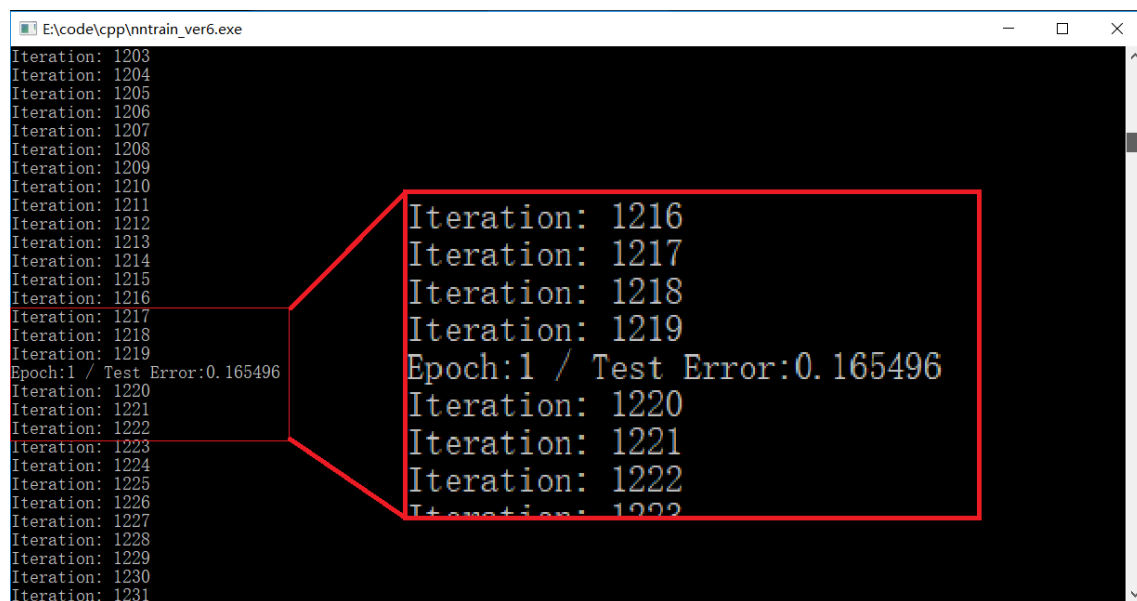
data_output[][](测试数据的预测输出)

函数返回值：无

B.4 程序运行效果

多层感知机网络的训练过程如图B.1所示，Iteration x 表明当前已对网络的每个权值更新过 x 次。

每训练完成一个 epoch 后，程序会输出 Epoch x / Test Error: y，表明当前已经训练完成 x 个 epoch，目前使用验证数据集估计得到的网络的均方误差为 y。



```
E:\code\cpp\nntrain_ver6.exe
Iteration: 1203
Iteration: 1204
Iteration: 1205
Iteration: 1206
Iteration: 1207
Iteration: 1208
Iteration: 1209
Iteration: 1210
Iteration: 1211
Iteration: 1212
Iteration: 1213
Iteration: 1214
Iteration: 1215
Iteration: 1216
Iteration: 1217
Iteration: 1218
Iteration: 1219
Epoch:1 / Test Error:0.165496
Iteration: 1220
Iteration: 1221
Iteration: 1222
Iteration: 1223
Iteration: 1224
Iteration: 1225
Iteration: 1226
Iteration: 1227
Iteration: 1228
Iteration: 1229
Iteration: 1230
Iteration: 1231
```

图 B.1 多层感知机网络训练程序运行效果