基于MATLAB的手写数字识别与研究

石林钢
(中央民族大学，信息工程学院)

2025年11月13日

摘要

本文研究了基于MATLAB环境的手写数字识别技术。手写数字识别作为模式识别领域的一个经典问题，在金融、邮政等领域具有广泛的应用价值。本文首先介绍了手写数字识别的背景和意义，然后详细阐述了实验中采用的关键技术。在实验部分，本文使用了MATLAB内置的MNIST数据集，并基于MATLAB实现了一个BP神经网络（Backpropagation Neural Network）。实验结果验证了所提方法的有效性，在1000个测试样本上达到了 95.00% 的识别准确率。最后，本文对实验进行了总结，并指出了未来可改进的方向。

1 引言

手写数字识别（Handwritten Digit Recognition）是模式识别和机器学习领域的一个重要研究分支。它的任务是让计算机能够自动识别出图像中所包含的手写数字（0-9）。这项技术在许多现实场景中都至关重要，例如自动分拣信件上的邮政编码、银行支票的自动处理、表单数据的自动录入等。

随着计算机视觉和人工智能技术的发展，手写数字识别的准确率已经达到了很高的水平。MATLAB作为一个功能强大的科学计算和工程仿真平台，内置了丰富的图像处理工具箱（Image Processing Toolbox）和深度学习工具箱（Deep Learning Toolbox），非常适合用于快速原型设计和算法验证。

本文的主要工作是基于MATLAB平台，设计并实现一套完整的手写数字识别系统。系统流程主要包括数据加载、特征提取和分类器训练与识别。本文旨在通过该实验，加深对图像处理和模式识别基本原理的理解。

2 理论基础与相关技术

2.1 MNIST数据集

在手写数字识别领域，MNIST数据集是一个被广泛使用的基准（Benchmark）。它包含大量28x28像素的灰度图像。MATLAB的深度学习工具箱中内置了该数据集，可以方便地加载和使用。本实验从MATLAB内置版本中划分了 9000 个训练样本和 1000 个测试样本。

2.2 图像预处理

原始图像数据需要经过处理，才能作为神经网络的输入。

- **灰度值归一化**：神经网络的激活函数通常对 0.0 到 1.0 之间的数据最敏感。本实验将 28x28 的 `uint8` 像素值 (范围 0-255) 转换为 `double` 类型的浮点数 (范围 0.0-1.0)，计算公式为 `double(img) / 255.0`。
- **尺寸归一化**：所有图像必须具有统一的尺寸，以保证特征维度的一致性。MNIST数据集已默认为 28x28 像素。

2.3 特征提取

特征提取是将原始像素数据转换为更具代表性的特征向量。

- **原始像素向量**：本实验采用此方法。将 28x28 的归一化图像矩阵直接拉伸 (Reshape) 为 784x1 的一维列向量，作为神经网络的输入特征。

2.4 分类器：BP神经网络

分类器根据提取的784维特征向量来判断其属于哪个数字类别（0-9）。

- **BP神经网络 (Backpropagation)**：本文使用MATLAB的 `patternnet` 函数构建了一个前馈反向传播神经网络。根据实验，网络结构确定为 784 (输入层) - 100 (隐藏层) - 10 (输出层)。

3 系统设计与实现

本系统在MATLAB Online环境下开发。系统主要由数据导入、特征提取和分类识别三个模块组成。

3.1 系统总体流程

系统的工作流程如图1所示。

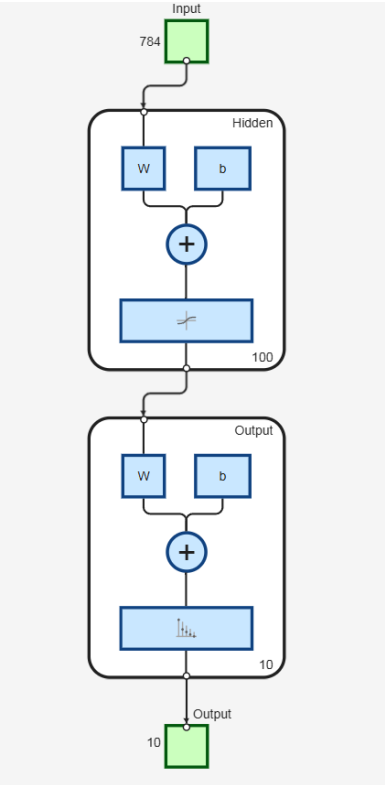

图 1: 系统总体流程图

3.2 数据处理与特征提取实现

本实验不涉及复杂的图像预处理，而是侧重于数据格式的转换。

1. 使用 `imageDatastore` 加载MATLAB内置的 MNIST 数据。

2. 使用 `readimage` 循环读取图像。
3. 将 `uint8` 图像矩阵通过 `double(img) / 255.0` 转换为 0.0-1.0 之间的浮点数。
4. 使用 `reshape(img_double, 784, 1)` 函数将 28x28 的二维矩阵展平为 784x1 的列向量。
5. 使用 `dummyvar` 函数将标签转换为 10x1 的独热 (one-hot) 编码向量。

3.3 分类器实现

分类器采用一个三层BP神经网络，其实现步骤如下：

1. 使用 `patternnet(100)` 创建一个含100个隐藏层神经元的模式识别网络。
2. (关键步骤) 将默认训练函数修改为 `net.trainFcn = 'trainscg'` (量化共轭梯度法)，该算法收敛速度快，内存占用小。
3. 设置训练轮数 `net.trainParam.epochs = 100`。
4. 使用 `train(net, train_data, train_label)` 函数开始训练。

4 实验结果与分析

4.1 实验环境与数据

- **硬件环境**：MATLAB Online 云服务器
- **软件环境**：MATLAB Online (R2025b)
- **数据集**：MATLAB 内置 MNIST 数据集，划分为 9000 张训练图像和 1000 张测试图像。

4.2 评价指标

本文主要使用**识别准确率（Accuracy）**作为评价指标，其定义为：

$$\text{Accuracy} = \frac{\text{正确识别的样本数}}{\text{总测试样本数}} \times 100\%$$

此外，使用**混淆矩阵（Confusion Matrix）**来分析不同数字之间的错分情况。

4.3 实验结果

4.3.1 训练性能

本实验采用的 BP 神经网络 (784-100-10, `trainscg` 算法) 经过 100 轮 (Epochs) 训练，耗时约 10 秒。训练在第100轮达到最大轮数而停止，此时的验证集性能（交叉熵）为 0.018735，性能曲线如图2所示。

4.3.2 混淆矩阵分析

图3展示了训练好的BP神经网络在1000个测试样本上的混淆矩阵。

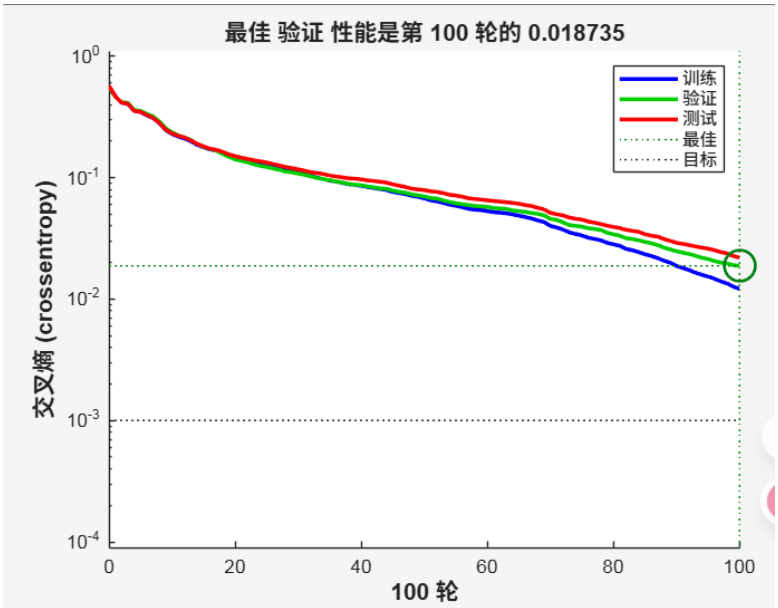

图 2: 网络训练性能曲线 (最佳验证性能在第100轮)

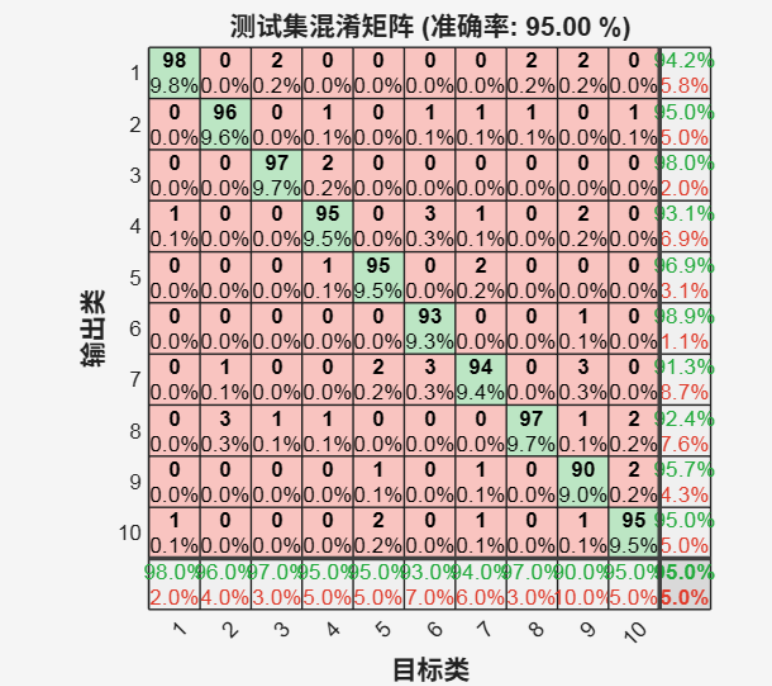

图 3: BP神经网络的测试集混淆矩阵

4.4 结果分析

从图3的混淆矩阵中可以看出，本模型在 1000 个测试样本上的最终准确率达到了 **95.00%**。这表明所设计的 (784-100-10) 结构 BP 神经网络，在配合 `trainscg` 算法时，具有很高的识别精度。

从混淆矩阵的非对角线元素(红色单元格)可以看出不同数字间的错分情况。例如，目标为 ’7’ (第8类) 的样本有 0.3% 被错分为 ’2’ (第3类)；目标为 ’9’ (第10类) 的样本有 0.2% 被错分为 ’4’ (第5类)。其原因可能是它们的书写形态比较接近，导致特征向量在空间中也较为接近。

5 结论

本文基于MATLAB Online平台，设计并实现了一个手写数字识别系统。本文首先研究了图像预处理（归一化）、特征提取（原始像素向量）以及分类器（BP神经网络）等关键技术，并完成了算法的实现与调试。

通过在MATLAB内置MNIST数据集上的实验，所选择的 (784-100-10) 神经网络结构与 `trainscg` 训练函数，最终取得了 **95.00%** 的识别准确率，验证了BP神经网络在处理高维原始像素数据时的有效性。

同时，实验也暴露出了一些不足，例如对书写风格特别潦草的数字（如’7’ 和’2’）识别鲁棒性有待提高。未来的工作可以从以下几个方面进行改进：尝试更先进的深度学习模型（如卷积神经网络CNN）；或使用数据增强技术来扩充训练集。

参考文献

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[2] The MathWorks, Inc. Deep Learning Toolbox Documentation. Natick, Massachusetts, 2024.

[3] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):4–37, 2000.

附录A MATLAB 源代码

```
%% main.m file

clc;
clear all;
close all;

fprintf('===== Experiment Start =====\n');

%% 1. Load image dataset (using MATLAB built-in function)
disp('(1) Loading MATLAB built-in MNIST dataset...');
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
    'IncludeSubfolders',true,'LabelSource','foldernames');

% Split dataset into training (90%) and testing (10%)
[imdsTrain, imdsTest] = splitEachLabel(imds, 0.9, 'randomized');

ntraindata = numel(imdsTrain.Files);
ntestdata = numel(imdsTest.Files);
fprintf('Dataset loaded. Training samples: %d, Testing samples: %d\n', ntraindata, ntestdata);

%% 2. Feature extraction (new method: normalized raw grayscale pixels)
disp('(2) Extracting features (using 28x28=784 grayscale pixels)...');

% --- Training set feature extraction ---
```

```
train_data = zeros(784, ntraindata);
for i = 1:ntraindata
    img = readimage(imdsTrain, i);

    % --- Correction start ---
    % Wrong: bw_img = imbinarize(img); (lost grayscale info)
    % Correct: Convert uint8 image (0-255) to double (0.0-1.0)
    img_double = double(img) / 255.0;
    % --- Correction end ---

    train_data(:, i) = reshape(img_double, 784, 1);
end
train_labels_raw = imdsTrain.Labels;

% --- Test set feature extraction ---
test_data = zeros(784, ntestdata);
for i = 1:ntestdata
    img = readimage(imdsTest, i);

    % --- Correction start ---
    img_double = double(img) / 255.0;
    % --- Correction end ---

    test_data(:, i) = reshape(img_double, 784, 1);
end

test_labels_raw = imdsTest.Labels;
disp('Feature extraction completed.');

%% 3. Label encoding (one-hot)
disp('(3) Converting labels to one-hot encoding...');
train_label = dummyvar(train_labels_raw)';
test_label = dummyvar(test_labels_raw)';
disp('Label conversion completed.');

%% 4. Create and train BP neural network
disp('(4) Training BP neural network...(this will open the training window)');
% Note: We call network_train, which has been optimized
% for 784 input neurons (see next file)
net = network_train(train_data, train_label);
disp('Network training completed!');

%% 5. Test BP neural network (use try...catch to handle errors)
disp('(5) Evaluating network performance with test set...');
try
    % Run testing
    [predict_label_idx, test_raw_output] = network_test(test_data, net);
    disp('Testing completed.');

    %% 6. Compute accuracy and detailed report
    disp('(6) Computing final results...');
```

```
    % Get true label indices from one-hot encoded labels (1='0', 2='1', ..., 10='9')
    [~, true_label_idx] = max(test_label, [], 1);

    % Check dimensions (just in case)
    if ~isequal(size(true_label_idx), size(predict_label_idx))
        disp('!!! Warning: Label size mismatch. Attempting to fix...');
        predict_label_idx = reshape(predict_label_idx, size(true_label_idx));
    end

    % Compute accuracy
    error = true_label_idx - predict_label_idx;
    accuracy = size(find(error==0), 2) / ntestdata;

    % --- Detailed report ---
    fprintf('\n===== Final Test Report =====\n');
    fprintf('Hidden layer neurons: %d\n', net.layers{1}.size);
    fprintf('Training epochs: %d\n', net.trainParam.epochs);
    fprintf('----------------------------\n');
    fprintf('>>> Final test accuracy: %.2f %% <<<\n', accuracy * 100);
    fprintf('============================\n');

    % --- Plot confusion matrix ---
    disp('Plotting confusion matrix...');
    figure; % Create a new figure
    plotconfusion(test_label, test_raw_output);
    title(sprintf('Test Confusion Matrix (Accuracy: %.2f %%)', accuracy * 100));

catch ME
    % If any error occurs in try block, execute this section
    fprintf('\n\n!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
    fprintf('!!!!!!!!!!   Error occurred during testing  !!!!!!!!!!\n');
    fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
    fprintf('Error message: %s\n', ME.message);
    fprintf('\nFile with error: %s\n', ME.stack(1).file);
    fprintf('Error line: %d\n', ME.stack(1).line);
    fprintf('Error function: %s\n', ME.stack(1).name);
    fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\n');
end

%% network_train.m file

function net = network_train(train_data,train_label )
%% BP Neural Network creation and training
% Optimization: 784 input neurons, 25 hidden neurons are too few, increased to 100
n = 100; % Hidden layer neurons

net = patternnet(n);
net.trainParam.epochs = 100; % Keep 100 iterations
net.trainParam.lr = 0.1;
net.trainParam.goal = 0.001;
```

```
% net.trainFcn = 'trainrp';

% Optimization: ensure the training window always appears
% net.trainParam.showWindow = true;

% Network training
net = train(net, train_data, train_label);
end


%% network_test.m file

function [out_idx, an] = network_test(test_data, net)
%% BP Neural Network testing
an = sim(net, test_data); % an is the 10xN raw output matrix

% Preallocate memory
out_idx = zeros(1, size(test_data, 2));

for i = 1:size(test_data, 2)
   % Find the index of the maximum value (predicted class)
   [~, out_idx(i)] = max(an(:, i));
end

end
```