项目报告

机器学习课程项目：中文手写数字识别

2020 年 12 月

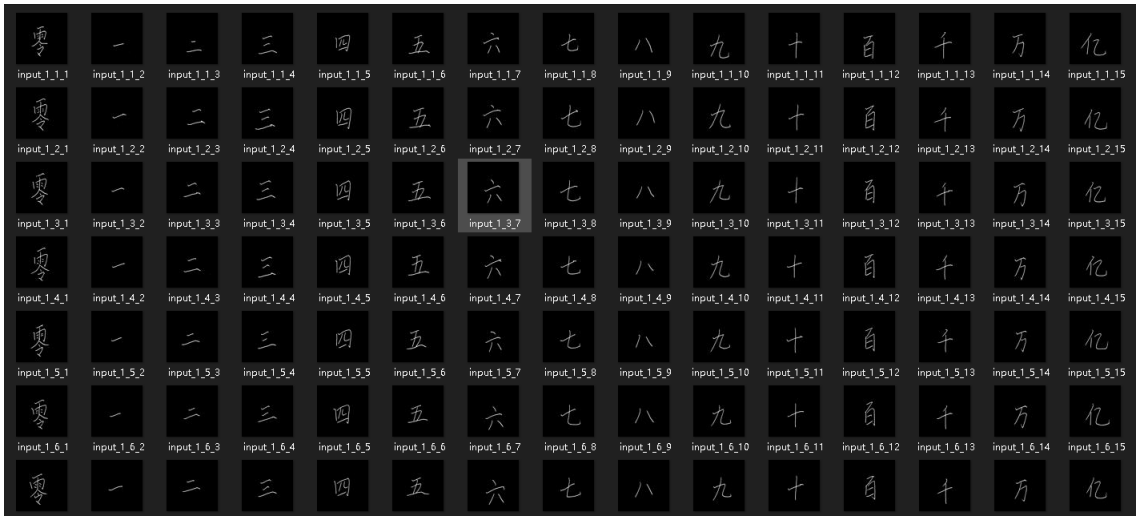
摘 要

此次实验，我们小组选择用神经网络模型用于中文手写数字的识别。课上老师有讲过用神经网络来做 MNIST 手写阿拉伯数字识别的实验，本项目在此基础上完成。本次实验利用 Matlab 构建了一个单隐层的神经网络，并且通过 10k 个训练样本以及 BP 算法对该网络进行了训练，最后使用训练出的网络对测试样本集中的样本进行的测试。虽然多次运行结果显示最终的实验结果不唯一，但识别的正确率基本在 0.9~1 之间，正确率较高，说明了 BP 算法的有效性，初步完成机器学习领域的一个 Toy Project。

关键词：Chinese-MNIST 特征提取 神经网络 BP 算法

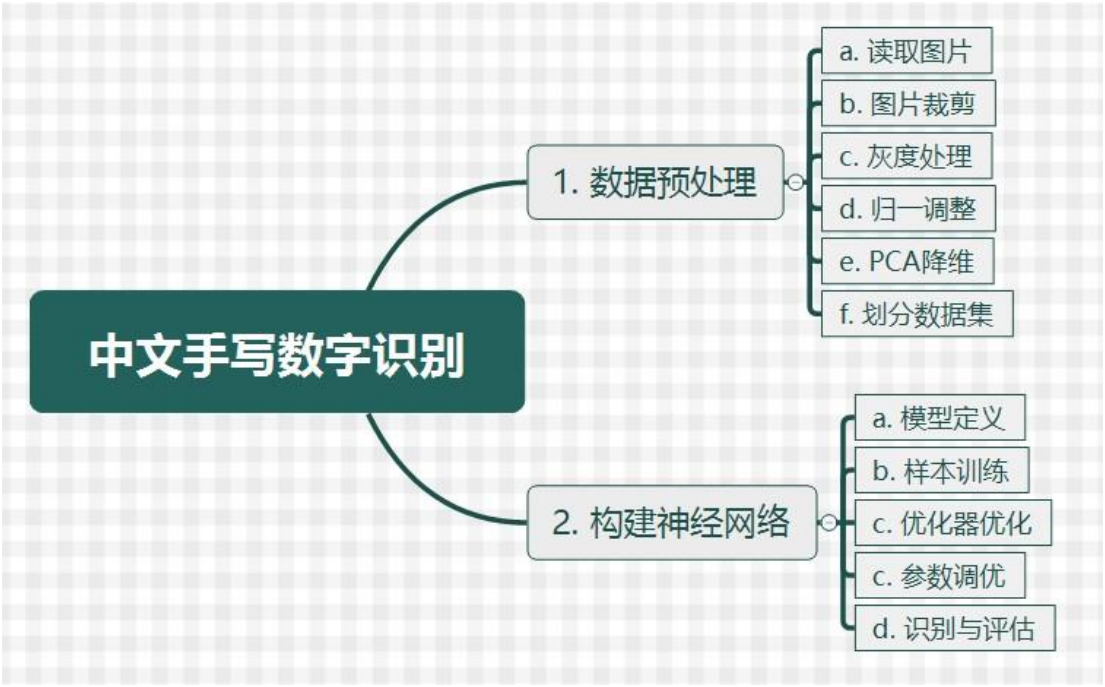
1 介绍

1.1 主要问题



原始数据集包含 15000 张 jpg 图像的文件夹，尺寸为 64 x64，和索引文件 chinese_mnist.csv。隐含信息为 label 与图片名称的关系。通过现有数据集进行数据处理与模型构建，最终测得识别的准确率来评估模型性能好坏。

1.2 解决方案



本文首先对待识别数据集的预处理进行了介绍，包括图像分割裁剪、归一化、二值化、PCA 降维等处理方法；其次，尝试了数字字符特征向量的提取；最后采用了 BP 神经网络算法,并以 MATLAB 作为工具实现了对中文手写数字数据集的识别，不断手动调整训练参数，最终形成模型。从实验结果来看，本方法具有较高的识别率，并有一定的的抗噪性能。

2 方法

2.1 数据预处理

2.1.0 预处理的必要性

在获取原始数字图像过程中，由于光照、背景纹理、镜头分辨率、拍摄角度等原因，难免会造成图像失真并带有噪声。由于这些噪声的影响，如果对获取得到的数字图像进行直接处理的话通常不能得到满意的结果，因此在获取原始数字图像后，需要对图像进行预处理。预处理阶段在该系统中是一个很重要的阶段。预处理效果的好坏会直接影响到整个系统的性能。

由于原始数字图像数据量大，冗余信息较多，一般不进行直接识别，而是进行提取有效特征数据、压缩数据，然后再进行识别。换句话说特征提取是为了去除图像信息中对分类没有帮助的部分，将图像信息集中到几个有代表性的特征上。

相关代码：

```
%% 准备工作，导入并预处理 data
disp('——>数据导入 ing...');
path_data = 'D:\MEDESKTOP\data2\handwritingPictures\';

files = dir(fullfile(path_data,'*.jpg'));
m = length(files);
n = 400;

y = zeros(m, 1);
X = zeros(m, n);

for i = 1 : m
    Img = imread(strcat(path_data,files(i).name));

    Img = imageCrop(Img);
    Img = imbinarize(Img,0.1);
    Img = imcomplement(Img);
    Img = imageResize(Img);
    Img = reshape(Img,20,20);
```

```
    kkk = strsplit(files(i).name, {'_', '.'});
    id = str2double(cell2mat(kkk(4)));
    y(i, :) = id;
    X(i, :) = (Img(:))';
end

for i = 1 : m
    minn = min(X(i, :));
    meann = mean(X(i,:));
    maxx = max(X(i, :));
    X(i, :) = (X(i, :) - meann) / (maxx - minn);
end

R = randperm(m);
num_train = 12000;

X_train = X(R(1:num_train), :);
y_train = y(R(1:num_train), :);

R(1:num_train) = [];

X_test = X(R, :);
y_test = y(R, :);
```

2.1.1 预处理的方法

对于字符识别的预处理过程一般包括：二值化、数字分割裁剪、归一化处理。经过预处理后的图片不仅能够有效滤除噪声，并且能够将不同的大小字符进行归一化到一个固定大小，对大量数据进行压缩处理。本实验尝试了图像裁剪、图像归一化处理、PCA 降维等方法。

在进行了灰度化处理之后，图像中的每个象素只有一个值，那就是象素的灰度值。它的大小决定了象素的亮暗程度。为了更加便利的开展下面的图像处理操作，还需要对已经得到的灰度图像做一个二值化处理。图像的二值化就是把图像中的象素根据一定的标准分化成两种颜色。

在数字图像处理中，二值化占有非常重要的地位。一方面类似于字符、指纹、工程图等图像本身就是二值的。另一方面，在某些情况下即使图像本身是有灰度的，也将其转换成二值图像再处理。这样在图像处理系统中，可以减少图像信息并提高处理速度。

对于手写数字识别来说，归一化是一个很重要的预处理因素。由于人们在书写、设备使用焦距不同等原因可以导致获取后的数字图像字符的大小不一。对于大小不一的字符图像，处理起来很不方便，因此在通常情况下，在对字符进行预处理的时候，我们必须将单个字符进行归一化处理。归一化处理一般的是将单字符图像进行归一化到固定的大小的数字图像，进过归一化处理后，一般的能够将大的字符图像信息进行缩小到固定大小的数字图像，这样在提高识别率的同时也能提高处理的效率。

归一化包括位置归一化，大小归一化及笔画粗细归一化。这里主要进行位置和大小的归一化，而笔画粗细的归一化可以看做是数字的细化。

相关代码（部分）：

```
function [pcaA V] = fastPCA( A, k )
% 输入：
% A --- 样本矩阵，每行为一个样本
% k --- 降维至 k 维
% 输出：
% pcaA --- 降维后的 k 维样本特征向量组成的矩阵，每行一个样本，列数 k 为降维后的样本特征维数
% V --- 主成分向量

[r c] = size(A);

% 样本均值
meanVec = mean(A);

% 计算协方差矩阵的转置 covMatT
Z = (A-repmat(meanVec, r, 1));
covMatT = Z * Z';

% 计算 covMatT 的前 k 个本征值和本征向量
[V D] = eigs(covMatT, k);

% 得到协方差矩阵 (covMatT)' 的本征向量
V = Z' * V;

% 本征向量归一化为单位本征向量
for i=1:k
    V(:,i)=V(:,i)/norm(V(:,i));
end

% 线性变换（投影）降维至 k 维
pcaA = Z * V;

% 保存变换矩阵 V 和变换原点 meanVec
```

2.1.2 特征提取

特征提取的目标是找到某种变换，将 n 维或 n×n 维的模式类别空间转换到维数更小的特征空间，并同时保留识别所需要的大部分信息。通过特征提取，模式分类可以在维数低得多的空间上进行，从而降低了计算的复杂度。而且，对给定的训练样本进行特征提取可以获得更精确的分类函数的描述，以构造更可靠的分类规则。

虽然，在一定意义上特征提取和特征选择都是要达到降维的目的，只是所实现的途径不同，特征提取是通过某种变换的方法组合原始高维特征，获得一组低

维的新特征，而特征选择是根据专家的经验知识或根据某种评价准则来挑选出那些对分类最优影响力的特征，并生成新的特征。有时这两者并不是截然分开的。例如可以先将原始特征空间映射到维数较低的空间，在这个空间中再进行选择以进一步降低维数。当然也可以先经过选择，去掉那些明显没有分类信息的特征，再进行映射，以降低维数。

相关代码（部分）：

```
function lett = imageResize(bw2)
% 提取特征，进行图片缩放

bw_2020=imresize(bw2,[200,200]);

for cnt=1:20
   for cnt2=1:20

data_temp=bw_2020(((cnt*10-9):(cnt*10)),((cnt2*10-9):(cnt
2*10)));
      atemp=sum(data_temp);
      lett((cnt-1)*20+cnt2)=sum(atemp);
   end
end

lett=lett';

end
```

2.1.3 数据集的划分

秉持着尽量选取较少的数据集作为训练集，来训练出性能较好的模型的原则，随机选择 10000 张图片作为训练集，5000 张图片作为测试集，比例为 2：1。

2.2 神经网络的构建

2.2.0 为什么选择神经网络模型

本质上讲，BP 算法就是以网络误差平方为目标函数、采用梯度下降法来计算目标函数的最小值。包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。

误差反传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，使误差

沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数(权值和阈值)，训练即告停止。

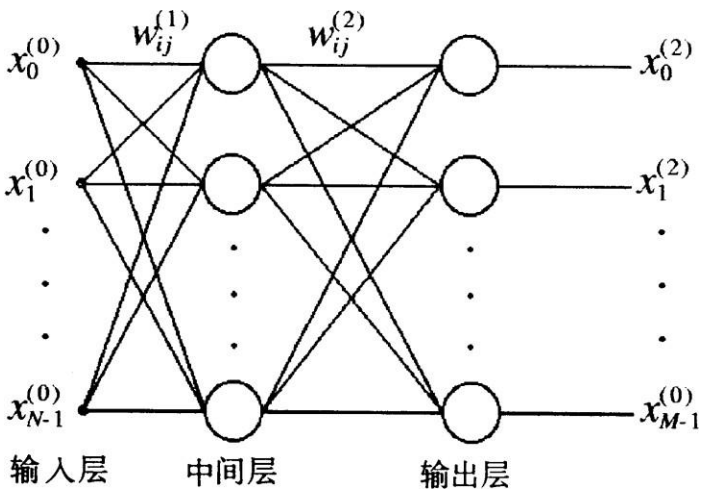
本项目中，首先进行随机初始化，构建单层神经网络，然后手动调整迭代次数和迭代次数等参数进行梯度下降，求出前向传播、反向传播的梯度和代价，加以正则化。

2.2.1 模型定义

对于输入层的节点数，一般与输入的特征向量的个数相同，由于本文提取的字符的特征向量有 400 个，因此采用的神经网络的输入层的节点数也为 400。

中间隐层 1000 个神经元。一般的选择隐含层的层数要从网络精度和训练时间上考虑，对于比较简单的映射关系，在网络精度达到要求的情况下，应该尽量选择较少的隐含层层数，对于较复杂的映射关系，可以通过增加隐含层层数，保证映射关系的正确实现。其实两个隐含层就能解决任何形式的分类的问题，事实上任何一个连续的函数，都可以用三层 BP 神经网络映射来逼近。因此本文选择三层 BP 神经网络（输入层、1 个隐含层、输出层）。

对于输出层的节点数，由于数字识别有 15 类，2 的 4 次方就可以覆盖 15 类，因此本文采用的神经网络的输出层的节点数为 4。



（图片参考）

相关代码（部分）：

```
function W = weightInit(L_in, L_out)
% 将权重随机初始化为小值
r = rand(L_out, 1 + L_in);
x = 0.12;
W = r * x * 2 - x;

end

function W = weightInitDebug(fan_out, fan_in)
W = zeros(fan_out, 1 + fan_in);
W = reshape(sin(1:numel(W)), size(W)) / 10;
```

```
end


function p = dataPredict(Theta1, Theta2, X)
m = size(X, 1);
num_labels = size(Theta2, 1);

% 前向传播
p = zeros(size(X, 1), 1);
h1 = Sigmoid([ones(m, 1) X] * Theta1');
h2 = Sigmoid([ones(m, 1) h1] * Theta2');

% 取预测值
[~, p] = max(h2, [], 2);

end
```

2.2.2 样本训练

相关代码（部分）：

```
function [J, grad] = costCompute(nn_params, input_layer_size,
hidden_layer_size, num_labels, X, y, lambda)

Theta1 = reshape(nn_params(1:hidden_layer_size *
(input_layer_size + 1)), hidden_layer_size,
(input_layer_size + 1));
Theta2 = reshape(nn_params((1 + (hidden_layer_size *
(input_layer_size + 1))):end), num_labels,
(hidden_layer_size + 1));

%  m 为训练样本个数
m = size(X, 1);

% 两个权值矩阵的梯度，代价函数的初始化
J = 0;
Theta1_grad = zeros(size(Theta1));
Theta2_grad = zeros(size(Theta2));

% 初始化独热编码
yNum = zeros(m, num_labels);

% 转换独热编码
for i = 1:m
```

```
  yNum(i, y(i)) = 1;
end

% 开始训练
a1 = X;

z2 = (Theta1 * [ones(m, 1) a1]')';
a2 = Sigmoid(z2);
z3 = (Theta2 * [ones(m, 1) a2]')';
a3 = Sigmoid(z3);

for i = 1:m
  J = J + (-yNum(i, :) * log(a3(i, :))' - (1.- yNum(i, :)) *
log(1.- a3(i, :))');

end

J = J / m;

% 正则化防止过拟合
t1 = Theta1(:, 2:end);
t2 = Theta2(:, 2:end);

% 正则化项
regularization = lambda / (2 * m) * (sum(sum(t1 .^ 2)) +
sum(sum(t2 .^ 2)));
J = J + regularization;

% 反向传播
d3 = a3-yNum;
d2 = (d3 * Theta2(:, 2:end)) .* SigmoidGradient(z2);
a2_with_a0 = [ones(m, 1) a2];
D2 = d3' * a2_with_a0;

Theta2_grad = D2 / m;

%正则化项
regularization = lambda / m * [zeros(size(Theta2, 1), 1)
Theta2(:, 2:end)];
Theta2_grad = Theta2_grad + regularization;
a1_with_a0 = [ones(m, 1) a1];
D1 = d2' * a1_with_a0;
Theta1_grad = D1 / m;
regularization = lambda / m * [zeros(size(Theta1, 1), 1)
Theta1(:, 2:end)];
Theta1_grad = Theta1_grad + regularization;

grad = [Theta1_grad(:) ; Theta2_grad(:)];
```

```
end


function numgrad = gradientCompute(J, theta)
% 返回：梯度的数值估计
% from: Andrew Ng's exercises

e = 1e-4;
numgrad = zeros(size(theta));
perturb = zeros(size(theta));

for p = 1:numel(theta)
    % 扰动向量
    perturb(p) = e;
    loss1 = J(theta - perturb);
    loss2 = J(theta + perturb);

    % 数值梯度
    numgrad(p) = (loss2 - loss1) / (2*e);
    perturb(p) = 0;
end

end

function gradientCheck(lambda)
% 设置一个神经网络进行梯度检测
% from: Andrew Ng's exercises
disp('计算[数值梯度][解析梯度]');

if ~exist('lambda', 'var') || isempty(lambda)
    lambda = 0;
end

input_layer_size = 3;
hidden_layer_size = 5;
num_labels = 3;
m = 5;

Theta1 = weightInitDebug(hidden_layer_size,
input_layer_size);
Theta2 = weightInitDebug(num_labels, hidden_layer_size);
X  = weightInitDebug(m, input_layer_size - 1);
y = 1 + mod(1:m, num_labels)';

nn_params = [Theta1(:) ; Theta2(:)];

costFunc = @(p) costCompute(p, input_layer_size,
hidden_layer_size, ...
```

```
                                num_labels, X, y, lambda);

[~, grad] = costFunc(nn_params);
numgrad = gradientCompute(costFunc, nn_params);

disp([numgrad grad]);

disp('check1--两栏数值是否非常接近');

disp('计算[Relative Difference]');

diff = norm(numgrad-grad)/norm(numgrad+grad);
fprintf('%g\n\n', diff);

disp('check2--数值是否小于 1e-9');

end
```

2.2.3 优化与调优

放入优化器、得到拟合率，参考了老师实验课和 Andrew Ng 的实验代码中封装好的优化器函数——fmincg 函数。

BP 神经网络中极小值比较多，所以很容易陷入局部极小值，这就要求对初始权值和阀值有要求，要使得初始权值和阀值随机性足够好，可以多次随机来实现。

相关代码（部分）：

```
function [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4,
P5)

...

end
```

3 实验性能

作为多分类问题，本项目的性能指标比较简单，参考学长建议，直接使用模型准确率作为模型评估指标。

Lambda	迭代次数	数据集	准确率
1.0	100	Test	0.8848
1.0	200	Test	0.9388
1.0	400	Test	0.9586
1.0	800	Test	0.9620
1.0	1600	Test	0.9672
1.5	100	Test	0.8966
1.5	200	Test	0.9340
1.5	400	Test	0.9622
1.5	800	Test	0.9600
1.0	1600	Test	0.9618

由程序运行结果可知，本次实验中对测试样本集的识别正确率达到了 90%+，验证该模型的有效性。但在实际的运行中发现，识别结果在每次训练中都不尽相同，即识别的结果具有不确定性，最严重时甚至可能出现代价函数无法稳定的情况，这些都与程序中大量使用随机函数有关。

而随着迭代次数增加，会过拟合，这也是一门学问，逐步求精；运行时间问题也需要考虑，虽然数据量很小，毕竟作为一个作业项目，跟实际工业界的应用还有很大差距。

4 讨论(结论)

本文对手写体中文数字识别的基本原理及方法作了介绍，并用 MATLAB 工具实现了大体地识别。在实际生活中，手写体数字识别应用比较广泛，例如银行票据认证、成绩自动录入等等。

本文从建立一个入门级别的中文手写数字识别模型的目的出发，对识别的重要几个环节：数据预处理、图片二值化、数据集分割、优化器优化、参数调优等进行了研究与实现。

一个维度的 BP 神经网络就相当于 CNN 中的全连接了，无非是多几个全连接，CNN 是二维，二维如果搞成全连接就导致运算量巨大，所以有了权重共享，大大减少了运算量，CNN 卷积的思想应该也源于 BP 神经网络。

至于二维卷积问题，其实图像处理里边早都有了各种微分和积分的卷积模板这样做过的，或许是受这些思想的启发，卷积的形式可以这样搞。

通过实验测试，本文设计的系统取得了较好的识别效果并具有良好的抗噪能力。但是还需要从预处理和特征选择方面做更深入研究，有待进一步提高系统的识别率。

在课题研究过程中，由于时间等条件等因素的限制，对一些问题的研究深度不够，在以下几个方面还需要做进一步研究工作:

（1）为了获得更好的识别效率，从特征向量入手，应想办法提取更精确的特征向量，比如投影特征、环凸凹特征等。

（2）如何简化图像预处理，以及针对某些步骤寻找更简单有效的方法，完善某些不甚成熟的地方，从而更好的提高识别率。

（3）采用 CNN 卷积神经网络/SVM 支持向量机等方法来搭建模型训练。

（4）不用 onehot 独热编码，因为带来了增加了大量的维数的副作用。

通过合作完成机器学习的项目作业，在实践中巩固了课上所学，很有意义。机器学习是当下计算机科学领域一大热门方向，本学期可以选修这么一门贴合前沿的专业课，非常幸运。借助本项目，大致了解了研究的流程与基本方法，为今后的深入研究打下基础。

以上就是此次项目作业的流程与心得，不得不说，有汗水也有满满的收获。感谢金老师每次课的倾心传授，尤其是晚上的实验课，老师辛苦了。也感谢各位助教学长，不厌其烦地解答了作为小白的许多问题，给了项目很多有用的意见。没有老师、学长双力相助，很难有有成果的产出，再次感恩！

5 代码运行请看项目根目录下的 README 文件