《人工智能与模式识别》

实 验 报 告 书

班级： 计科1801

学号： 181604124

姓名： 汤礽禾

指导教师： 翟婷婷

2020-2021学年 第 二 学期

**实验名称： 集成学习算法的实现和应用**

**实验时间： 2021 年 5 月 12 日 第 11 周 星期 三**

一、实验目的

1. 熟悉BP神经网络的原理，并能够灵活运用来解决给定的手写数字分类问题。

2. 熟悉两种经典的集成学习算法的思想：Adaboost和Bagging，并能以BP神经网络作为弱学习算法编程实现这两个集成学习算法。

3. 能够实际运用Adaboost和Bagging解决手写数字分类问题，理解集成算法相比于单个弱学习算法的优势所在。

二、实验预习（预备知识的问题及回答）

1. 简述BP神经网络如何接受带权值的训练样本集来训练神经网络的参数。(1分)

**解答：**

**BP算法基于“梯度下降”的迭代算法，在迭代的每一轮中，按照误差的负梯度方向对网络参数进行调整：**

****

**u是网络的某个参数，是学习步长，Et是当前的误差。**

2. 一个已经训练好的BP神经网络模型对待识别的实例***x***的输出为向量，则网络应该将***x***预测为哪一类？ (1分)

**解答：**

**将*x*预测为向量y中值最大的一个yi对应的类。**

3. 设Adaboost算法总共训练了M个弱学习器，且以BP神经网络作为弱学习算法。在预测一个实例***x***的类标时，若第个训练好的弱学习器对***x***的预测为向量，且该弱学习器的决策权重为，则Adaboost算法应该将***x***预测为哪一类？请用数学公式表示。 (1分)

**解答：**

4. 当Bagging算法以BP神经网络作为弱学习算法时，如果第个训练好的弱学习器将***x***预测为类，则Bagging算法应该将***x***预测为哪一类？ (1分)

**解答：**

**Bagging算法采用多数投票原则。有最多弱分类器预测的类为x的最终所属的类。**

三、实验内容（包含实验所用命令或相关程序源代码）

mnist手写数字图像数据集，其中包含了0到9的手写数字的灰度图像，每个数字都包含大量的不同风格书写的该数字的图像。手写数字图像识别的任务是：给定任意一张手写数字图像，能够精确识别出该图像所对应的数字是什么。显然，手写数字图像识别问题是个多分类问题。本实验中用到的mnist\_exp数据集是mnist数据集的一个子集。

1. 利用mnist\_exp数据集训练出一个BP神经网络模型，用来对手写数字图像进行分类，并评估训练出的模型的测试准确率。

1. 要求使用**图像的像素矩阵**作为每一张图像的特征，来构建训练数据集和测试数据集。要求每个数字文件夹中的第i张图片，如果满足i对10取余<7，则将其作为训练图片，否则将其作为测试图片。注意也要构建训练数据集和测试数据集的标记向量。

提示：使用imageDatastore函数一次性读取所有图片的路径名。

**Matlab代码：**（2分）

%%

clear;clc;close all;

%%

testSet = [];

trainSet = [];

trainY = [];

testY = [];

DataPath = E:\扬大\模式识别\实验\实验4 5.12\trh\mnist\_exp\';

DataDir = dir(DataPath);

for i = 1:length(DataDir)

if(isequal(DataDir(i).name,'.')||...

isequal(DataDir(i).name,'..')||...

~DataDir(i).isdir)

continue;

end

faceImgDir = dir([DataPath, DataDir(i).name, '/\*.jpg']);

imds = imageDatastore([DataPath, DataDir(i).name],'FileExtensions',{'.jpg'});

y = str2num(imds.Files{i}(size(imds.Files{i},2)-10)); % label

for j = 1:length(imds.Files)

url = imds.Files(j);

url = url{1};

temp = imread(url);

if(mod(j, 10) < 7)

trainSet = [trainSet; im2double(reshape(temp, 1, [])), str2double(DataDir(i).name)];

trainY = [trainY;y];

else

testSet = [testSet; im2double(reshape(temp, 1, [])), str2double(DataDir(i).name)];

testY = [testY;y];

end

end

end

%% ´存储数据

% allData = {trainSet,trainY,testSet,testY};

% save('allData','allData');

%%

% % 加载数据

% load('allData.mat');

1. 编写只包含单个隐层的BP神经网络的训练算法，将该训练算法写成一个函数：

function [network] = BP\_train(epoch, neuros, eta, data, label)，其中各参数含义如下：

epoch：遍历整个训练数据集的次数； neuros：隐层神经元的个数

eta：学习步长; data是训练神经网络使用的训练数据集，label是训练数据集的标签向量

Network：是一个结构体，将训练得到的所有网络参数存储在该结构体中，作为返回值

**Matlab代码：**（4分）

function [network] = BP\_train(epoch, neuros, eta, data, label)

[n,m] = size(data) ;

w1 = rand(m,neuros);

theta1 = rand(1,neuros);

w2 = rand(neuros,10);

theta2 = rand(1,10);

for e = 1:epoch

randindex = randperm(n);

randData = data(randindex,:);

randY = label(randindex);

allCost = 0;

for i = 1:n

sampleData = randData(i,:);

sampleLabel = zeros(1,10);

sampleLabel(randY(i)+1) = 1;

A = sampleData \* w1;

Y1 = sigmoid(A- theta1);

B = Y1 \* w2;

Y2 = sigmoid(B - theta2);

cost = (Y2-sampleLabel)\*(Y2-sampleLabel)'/2;

allCost = allCost + cost;

%

tz = sigmoid(theta2) .\* (1 - sigmoid(theta2));

gradient\_theta2 = (-1)\*tz .\* (Y2-sampleLabel);

tz2 = sigmoid(B) .\* (1 - sigmoid(B));

gradient\_w2 = Y1'\*tz2.\*(Y2-sampleLabel);

tz3 = sigmoid(theta1) .\* (1 - sigmoid(theta1));

gradient\_theta1 = (-1)\*tz3 .\* (tz2.\*(Y2-sampleLabel)\* w2' );

tz4 = sigmoid(A) .\* (1 - sigmoid(A));

gradient\_w1 = sampleData'\*(tz4 .\* (tz2.\*(Y2-sampleLabel)\* w2' ));

w1 = w1 - eta \* gradient\_w1;

theta1 = theta1- eta \* gradient\_theta1;

w2 = w2 - eta \* gradient\_w2;

theta2 = theta2- eta \* gradient\_theta2;

end

end

network = {w1,theta1,w2,theta2};

1. 编写使用训练好的神经网络进行预测的算法，将该算法写成一个函数：

function [pred\_label, pred\_y] = BP\_predict(network, X)，其中各参数含义如下：

Network: 训练好的神经网络 X：是测试数据集（不带标签）

pred\_label: 是一个向量，其中，第i个元素表示对第i个测试实例预测得到的标签

pred\_y：是一个矩阵，第i 行表示网络对第i个测试实例的输出向量

**Matlab代码：**（2分）

function [pred\_label, pred\_y] = BP\_predict(network, X)

% 读参数

w1 = network{1};

theta1 = network{2};

w2 = network{3};

theta2 = network{4};

%

A = X \* w1;

Y1 = sigmoid(A - theta1);

B = Y1 \* w2;

pred\_y = sigmoid(B - theta2);

[~,pred\_label] = max(pred\_y,[],2);

pred\_label = pred\_label - 1;

1. 调用上述函数，在(1)得到的训练数据集上训练一个BP神经网络，并计算该网络在测试数据集上的测试准确率。要求：

①先对训练数据运用PCA进行降维，保存转换矩阵Q，然后在降维后的数据上训练神经网络；在测试时，也要对数据进行相同的PCA变换，即利用转换矩阵Q，对测试数据进行降维，然后再进行预测。

②函数BP\_train函数的参数设置为：epoch=500 , neuros=40 , eta = 0.1。

**Matlab代码和运行截图：**（3分）

% 读取数据

load('allData.mat');

trainData = allData{1};

k = 100;

[coeff,score] = pca(trainData);

trainData=trainData\*coeff(:,1:k);

trainY = allData{2};

testData = allData{3};

testData=testData\*coeff(:,1:k);

testY = allData{4};

%参数设置

epoch = 500;

neuros = 40;

eta = 0.1;

% BP\_train

[network] = BP\_train(epoch, neuros, eta, trainData, trainY);

[pred\_label, pred\_y] = BP\_predict(network, testData);

acc = sum(pred\_label == testY) / size(testY,1);

fprintf("准确率:3.3f\n",acc);



1. 在手写数字识别任务中，4和9是比较难区分的。利用mnist\_4\_vs\_9数据集，训练一个Adaboost分类器，用来对4和9的图像进行分类，并评估训练出的集成模型的测试准确率。
2. 按照与第1题相同的方式构建训练数据集和测试数据集，以及相应的标签向量。

**Matlab代码：**（1分）

function [trainData,trainY,testData,testY] = readData()

trainData = [];

trainY = [];

testData = [];

testY = [];

for numClass=0:9

if(numClass~=4 & numClass~=9)

continue

end

imds = imageDatastore(fullfile('mnist\_4\_vs\_9',num2str(numClass)),'FileExtensions','.jpg');

files = imds.Files;

for k=1:size(files,1)

len = size(files{k},2);

id = str2num(files{k}(len-8:len-4));

x = imread(files{k});

x = im2double(x);

x = reshape(x,1,28\*28);

y = str2num(files{k}(len-10));

if(mod(id,10)<7)

trainData = [trainData;x];

trainY = [trainY;y];

else

testData = [testData;x];

testY = [testY;y];

end

end

end

1. 编写Adaboost算法，要求弱学习算法采用BP神经网络算法，将该集成分类器的训练算法写成一个函数：function [myensemble] = my\_adaboost(data,label,M)，其中，

data: 训练数据集, label:训练数据集的标签向量, M为预定义的弱分类器数目

myensemble：保存了所学到的M个神经网络模型，提示可以用cell数组存放每个模型

**Matlab代码：**（5分）

%%

addpath("BP");

function [myensemble] = my\_adaboost(data,label,M)

epoch = 200;

neuros = 20;

eta = 0.5;

varData = data ;

myensemble = [];

t = 1;

while t <= M

[network] = BP\_train(epoch, neuros, eta, varData, label);

[pred\_label, ~] = BP\_predict(network, varData);

acc = sum(pred\_label==label) / size(label,1);

err = 1-acc;

fprintf("第%d个学习器再训练集的分类准确率为%0.3f,",t,acc);

a = 1/2 \*log((1-err)/err);

multiTimes = ones(size(data,1),1);

for i = 1:size(data,1)

if(pred\_label(i)==label(i))

multiTimes(i) = exp(-a);

else

multiTimes(i) = exp(a);

end

end

Z = sum(multiTimes);

multiTimes = multiTimes / Z;

multiTimes = multiTimes+1;

varData = varData.\*multiTimes;

myensemble = [myensemble;network,a];

fprintf("¸该学习器的训练误差:%0.3f\n",err);

t = t+1;

end

end

1. 编写使用训练好的Adaboost分类器进行预测的算法，将该算法写成一个函数：

function [pred\_label] = adaboost\_predict(ensemble,X)，其中各参数含义如下：

ensemble: 训练好的集成模型 X：是测试数据集（不带标签）

pred\_label: 是一个向量，其中，第i个元素表示对第i个测试实例预测得到的标签

**Matlab代码：**（2分）

%%

function [pred\_label] = adaboost\_predict(ensemble,X)

all\_pred\_y = 0;

for i = 1:size(ensemble,1)

subNetwork = {ensemble{i,1},ensemble{i,2},ensemble{i,3},ensemble{i,4}};

[~, sub\_pred\_y] = BP\_predict(subNetwork, X);

all\_pred\_y = all\_pred\_y + ensemble{i,5} \* sub\_pred\_y;

end

[~, pred\_label] =max(all\_pred\_y,[],2);

pred\_label = pred\_label -1;

(4) 调用上述函数，在(1)得到的训练数据集上训练一个Adaboost分类器，并计算其在测试数据集上的测试准确率。要求：

①先对训练数据运用PCA进行降维后再训练网络；在测试时，也要对测试数据进行相同的PCA变换，再进行预测。

②函数my\_adaboost的参数设置为：M = 20。

③训练神经函数时，BP\_train函数中epoch=200 , neuros=20 , eta = 0.5。

④在训练过程中，使用disp函数输出每个弱分类器的训练误差。

**Matlab代码和运行截图：**（2分

close all; clc; clear;

%% 加载数据

load('Data');

%%

M = 20;

data=trainData;

label=trainY;

k = 100;

[coeff,score] = pca(trainData);

trainData=trainData\*coeff(:,1:k);

testData=testData\*coeff(:,1:k);

%% adaboos

[myensemble] =my\_adaboost(trainData,trainY,M);

%% predict

[pred\_label] = predict(myensemble,testData);

rs = sum(pred\_label==testY) / size(testY,1);

fprintf("测试准确率:%3.3f\n",rs);