《人工智能与模式识别》

实 验 报 告 书

班级： 计科 2002

学号： 202801108

姓名： 李蔚

指导教师： 翟婷婷

2022-2023学年 第 二 学期

**实验名称： 集成分类器的应用**

**实验时间： 2023 年 6 月 2 日 第 16 周 星期 五**

一、实验目的

1. 理解两种经典的集成学习的思想：Adaboost和Bagging，并以BP神经网络作为弱分类器编程实现两个集成分类器。

2. 实际运用Adaboost和Bagging解决手写数字分类问题，对比集成分类器和单个弱分类器的性能，理解集成的优势。

二、实验预习（预备知识的问题及回答）

1. 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。 (0.5分)

**解答：**Adaboost算法预测实例x的类别为：

argmax(c) { Σ( \* ) }

其中，argmax(c)表示在c个类别中选择具有最大值的索引，Σ表示求和运算，表示第k个弱学习器的决策权重，表示第k个弱学习器对x的预测结果中第c个类别的预测值。

通过计算所有弱学习器的预测结果的加权和，并选择具有最大加权和的类别，即可得到Adaboost算法对实例x的类别预测结果。

2. 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？ (0.5分)

**解答：**Bagging的预测结果通常是通过多个弱学习器的投票来决定的。每个弱学习器投票给一个类别，最终预测结果是得票数最多的类别，即的众数。

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。

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。

1. 按照与实验7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）

% 读取所有图片的路径名

imageDir = 'D:\mnist\_4\_vs\_9\mnist\_4\_vs\_9';

imageData = imageDatastore(imageDir, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');

% 初始化训练数据集和测试数据集

trainData = [];

trainLabels = [];

testData = [];

testLabels = [];

% 遍历所有图片

for i = 1:length(imageData.Files)

% 读取图片

img = readimage(imageData, i);

% 将图像像素矩阵转换为 double 类型

img = im2double(img);

% 将图像像素矩阵拉伸为行向量

imgVector = reshape(img, 1, []);

% 获取图像对应的标签

label = imageData.Labels(i);

% 判断图片是作为训练图片还是测试图片

if mod(i, 10) < 7

% 将图片添加到训练数据集

trainData = [trainData; imgVector];

trainLabels = [trainLabels; label];

else

% 将图片添加到测试数据集

testData = [testData; imgVector];

testLabels = [testLabels; label];

end

end

% 将标签向量转换为独热编码形式

trainLabels = dummyvar(double(trainLabels));

testLabels = dummyvar(double(testLabels));

% 保存训练和测试数据以及标签

save('Data8\_1.mat', 'trainData', 'testData', 'trainLabels', 'testLabels');

clear; % 清除工作区变量

clc; % 清空命令行

load('Data8\_1.mat');% 加载数据

% Sigmoid函数

function y = sigmoid(x)

y = 1 ./ (1 + exp(-x));

end

% Sigmoid函数导数

function y = sigmoid\_derivative(x)

y = sigmoid(x) .\* (1 - sigmoid(x));

end

% BP神经网络训练函数

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

input\_size = size(data, 2); % 输入层大小

output\_size = 2; % 输出层大小

% 初始化网络参数

W1 = rand(neuros, input\_size);

b1 = rand(neuros, 1);

W2 = rand(output\_size, neuros);

b2 = rand(output\_size, 1);

% 开始训练

for e = 1:epoch

for i = 1:size(data, 1)

% 前向传播

x = data(i, :)';

y = label(i,:)';

z1 = W1 \* x + b1;

a1 = sigmoid(z1);

z2 = W2 \* a1 + b2;

a2 = sigmoid(z2);

% 反向传播

delta2 = (a2 - y) .\* sigmoid\_derivative(z2);

delta1 = (W2' \* delta2) .\* sigmoid\_derivative(z1);

% 更新权重和偏置

W2 = W2 - eta \* delta2 \* a1' \* D(i);

b2 = b2 - eta \* delta2 \* D(i);

W1 = W1 - eta \* delta1 \* x';

b1 = b1 - eta \* delta1;

end

end

% 存储网络参数

network.W1 = W1;

network.b1 = b1;

network.W2 = W2;

network.b2 = b2;

end

% BP神经网络预测函数

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

W1 = network.W1;

b1 = network.b1;

W2 = network.W2;

b2 = network.b2;

pred\_label = zeros(size(X, 1), 1);

pred\_y = zeros(size(X, 1), size(W2, 1));

for i = 1:size(X, 1)

x = X(i, :)';

z1 = W1 \* x + b1;

a1 = sigmoid(z1);

z2 = W2 \* a1 + b2;

a2 = softmax(z2);

[~, idx] = max(a2);

pred\_label(i) = (idx-1.5)\*2;

pred\_y(i, :) = a2';

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] = adaboost\_train(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：**（8分）

% Adaboost算法训练函数

function [myensemble] = adaboost\_train(data, label, M)

N = size(data, 1); % 训练数据集大小

D = ones(N, 1) / N; % 初始化样本权重为均匀分布

myensemble = cell(M, 1); % 用于保存所学到的M个神经网络模型

for m = 1:M

% 使用当前样本权重训练一个BP神经网络

weak\_learner = BP\_train(200, 20, 0.5, data, label, D);

% 计算当前弱学习器的训练误差

[pred\_label, ~] = BP\_predict(weak\_learner, data);

trueLabels = (vec2ind(label')'-1.5)\*2;

err = sum(trueLabels ~= pred\_label) / length(pred\_label);

% 计算当前弱学习器的权重

alpha = 0.5 \* log((1 - err) / err);

% 更新样本权重

for i = 1:M

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

D(i) = D(i) \* exp(-alpha);

else

D(i) = D(i) \* exp(alpha);

end

end

D = D / sum(D);

% 保存当前弱学习器及其权重到集成模型中

myensemble{m} = struct('model', weak\_learner, 'weight', alpha);

% 输出当前弱学习器的训练误差

disp(['弱学习器 ', num2str(m), ' 的训练误差: ', num2str(err\*100),'%']);

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：**（3分）

% Adaboost算法预测函数

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

M = length(ensemble); % 弱分类器数目

N = size(X, 1); % 测试数据集大小

pred\_label = zeros(N, 1); % 预测标签向量

for m = 1:M

weak\_learner = ensemble{m}.model; % 获取当前弱学习器

% 使用当前弱学习器进行预测

[pred, ~] = BP\_predict(weak\_learner, X);

% 根据权重加权累加预测结果

pred\_label = pred\_label + ensemble{m}.weight \* pred;

end

% 将预测结果转换为最终标签

pred\_label = sign(pred\_label);

end

(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adaboost\_train的参数设置为：M = 20。

③训练神经网络弱学习器时，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）

% 调用PCA进行降维

[coeff] = pca(trainData);

Q = coeff(:, 1:40);

trainData\_pca = trainData \* Q;

testData\_pca = testData \* Q;

% Adaboost算法训练

M=20;

myensemble = adaboost\_train(trainData\_pca, trainLabels, M);

% 在测试集上进行预测

pred\_label = adaboost\_predict(myensemble, testData\_pca);

% 计算测试准确率

trueLabels = (vec2ind(testLabels')'-1.5)\*2;

accuracy = sum(pred\_label == trueLabels) / length(trueLabels);

disp(['Adaboost算法测试准确率: ', num2str(accuracy\*100),'%']);

