**OpenCV2.0中使用随机森林进行分类**

**一 随机森林原理**

随机森林是使用决策树作为基分类器实现的，随机森林及决策树的原理请参考清华大学出版社周志华的《机器学习》或其它教材。

**二 基于OpenCV实现**

**2.1函数说明**

在OpenCV2中，随机森林由CvDTree类实现。使用方法很简单：

1、 定义CvDTree对象

2、 训练

3、 测试

在训练时，比较重要的一个参数是CvDTreeParams，在OpenCV2的文档中，解释如下：

CvDTreeParams( int max\_depth, int min\_sample\_count, float regression\_accuracy, bool use\_surrogates, int max\_categories, int cv\_folds, bool use\_1se\_rule,bool truncate\_pruned\_tree, const float\* priors );

Max\_depth:树的最大深度，当树的当前节点深度小于最大深度时，算法会尝试分裂节点，实际树的深度可能会小于该深度，因为其它停止条件达到时。

Min\_sample\_count:最小样本数量，如果一个节点中的样本数量少于该值，则该节点将不会再继续分裂。

Regression\_accuracy:回归树的停止条件

Use\_surrogates:若为真，将执行surrogate分裂，这种分裂允许缺失数据。

Max\_categories:最大类别

Cv\_folds:若该值大于为，执行K折交叉验证，将样本集分为K份，其中K-1份作为训练集，而另外一份作为验证数据集，用验证集来验证所得分类器或者回归的错误码率。

Use\_1se\_rule:若为真，执行修剪

Truncate\_pruned\_tree：（不太理解这一参数）

Priois:预测的权重

**2.2代码**

#include "stdafx.h"

#include <opencv2\opencv.hpp>

using namespace std;

using namespace cv;

int \_tmain(int argc, \_TCHAR\* argv[])

{

//准备训练数据

float fdata0[10][2] = { { 1.0, 1.0 }, { 1.0, 2.0 }, { 1.9, 1.5 }, { 6.0, 3.0 }, { 4.0, 2.0 },

{ 2.0, 0.0 }, { 3.0, 0.0 }, { 4.0, 0.0 }, { 4.0, 1.0 }, { 5.2, 2.5 } };

float fdata1[10][2] = { { 2.5, 1.5 }, { 2.7, 2.0 }, { 2.2, 2.1 }, { 2.5, 4.0 }, { 3.0, 4.0 },

{ 4.0, 3.5 }, { 5.0, 4.0 }, { 4.5, 4.5 }, { 3.5, 3.1 }, { 5.5, 5.0 } };

Mat traindata1(10, 2, CV\_32FC1, fdata0);

Mat traindata2(10, 2, CV\_32FC1, fdata1);

Mat trainMat;

trainMat.push\_back(traindata1);

trainMat.push\_back(traindata2);

float responsedata[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

Mat responseMat(20, 1, CV\_32F, responsedata);

//定义并训练随机森林

CvDTree \*tree=new CvDTree;

float priors[] = { 1, 1 };

CvDTreeParams params(8, 1, 0, true, 15, 0, true, true, priors);

tree->train(trainMat, CV\_ROW\_SAMPLE, responseMat, Mat(), Mat(), Mat(), Mat(), params);

//测试数据

while (1)

{

int choice;

cout << "请选择：1)测试 0)退出：";

cin >> choice;

if (choice)

{

float x[2];

cout << "请输入第一个特征";

cin >> x[0];

cout << "请输入第二个特征";

cin >> x[1];

Mat sample(1, 2, CV\_32FC1, x);

float r = tree->predict(sample)->value;

cout << "(" << x[0] << "," << x[1] << ")输入类别" << cvRound(r) << endl<< endl;

}

else

{

break;

}

}

return 0;

}

 