OpenCV 之 HaarTraining 算法剖析

作者：周明才
Email：mingcai.zhou@ia.ac.cn

由于作者水平有限，文中还会有不妥甚至错误之处，欢迎大家批评指正，同时希望能与一起交流探讨！QQ:4356176
2008.10.08 第二版

1.引言

通过前段时间阅读 OpenCV 的 HaarTraining 代码，基本掌握了 HaarTraining 算法。现将其中的算法作一梳理，同时对 HaarTraining 的使用方法做一简要说明。

HaarTraining 算法总体上以 Friedman, J. H 等人的“Additive Logistic Regression: a Statistical View of Boosting”为出发点，实现了其中 2 类分类问题的 4 种 Boost 算法：Discrete AdaBoost, Real AdaBoost, LogitBoost 和 Gentle AdaBoost。同时实现了文中第 8 节 Additive Logistic Trees 和第 9 节 Weight Trimming.

Friedman, J. H 等人只描述了如何训练一个强分类器，对于训练级联的强分类器（Cascade of Classifiers），OpenCV 采用的是 Paul Viola 等人的“Robust Real-Time Face Detection”中所述方法。

HaarTraining 采用的是 OpenCV 扩展的 Haar 特征，具体描述可参考 Rainer Lienhart 等人的“An Extended Set of Haar-like Features for Rapid Object Detection”。

2.总体框架

要训练一个 Haar 分类器，总体上包括 3 步：1)准备正负样本；2)用 CreateSamples 程序建正样本集；3)用 HaarTraining 程序训练，得到最终的分类器模型（xml 文件）。如图 2.1 所示。

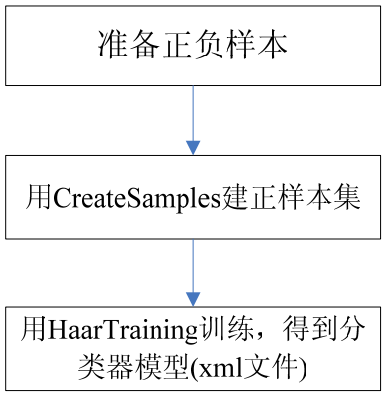


图 2.1 训练 Haar 分类器总器框架

3. 样本准备

HaarTraining 需要使用正样本和负样本进行训练。下面分别进行描述。

3.1 正样本

对于正样本，通常的做法是先把所有正样本裁切好，并对尺寸做规整（即缩放至指定大小），如图 3.1 所示。

图 3.1 正样本集示意图

由于 HaarTraining 训练时输入的正样本是 vec 文件，所以需要使用 OpenCV 自带的 CreateSamples 程序将准备好的正样本转换为 vec 文件。转换的步骤如下：

1) 制作一个正样本描述文件，用于描述正样本文件名（包括绝对路径或相对路径），正样本数目以及各正样本在图片中的位置和大小。典型的正样本描述文件如下：

   face_100/face00001.bmp 1 0 0 20 20

   face_100/face00002.bmp 1 0 0 20 20

   face_100/face00003.bmp 1 0 0 20 20

   …

   可采用 Dos 命令结合 EditPlus 软件生成样本描述文件。具体方法是在 Dos 下的恰当目录敲入 dir face_100 /b > samples.dat，则会生成一个 samples.dat，里面包含所有正样本文件名列表，但没有相对路径名和正样本位置信息。在 samples.dat 文件各行行首增加“face _100/”的方法是使用 EditPlus，先选中所有行，然后按 Tab 键为每行增加一个制表位，然后将制表位全部替换为“face _100/”即可。通过将“bmp”替换为“bmp 1 0 0 20 20”即可在每行添加“1 0 0 20 20”。

2) 运行 CreateSamples 程序。如果直接在 VC 环境下运行，可以在 Project\Settings\Debug 属性页的 Program arguments 栏设置运行参数。下面是一个运行参数示例：

   -info F:\FaceDetect\samples.dat -vec F:\FaceDetect\samples.vec -num 200 -w 20 -h 20

   表示有 200 个样本，样本宽 20，高 20，正样本描述文件为 samples.dat，结果输出到 samples.vec。

3) 运行完了会生成一个*.vec 的文件。该文件包含正样本数目，宽高以及所有样本图

像数据。

3.2 负样本

负样本图像可以是不含有正样本模式的任何图像，比如一些风景照等。训练时，OpenCV 需要一个负样本描述文件，该文件只需包含所有负样本的文件名及绝对（或相对）路径名。以下是一个负样本描述文件内容示例：

nonface_200/00001.bmp
nonface_200/00002.bmp
nonface_200/00003.bmp
…
负样本描述文件的生成方法可参照正样本描述文件生成方法。

负样本图像的大小只要不小于正样本就可以，在使用负样本时，OpenCV 自动从负样本图像中抠出一块和正样本同样大小的区域作为负样本，具体可查看函数 icvGetNextFromBackgroundData（）。具体抠图过程为：

1) 确定抠图区域的左上角坐标(Point.x, Point.y)
2) 确定一个最小缩放比例，使得原负样本图像缩放后恰好包含选中负样本区域
3) 对原负样本图象按计算好的缩放比例进行缩放
4) 在缩放后的图像上抠出负样本，如图 3.2 左半部分的虚线框所示。

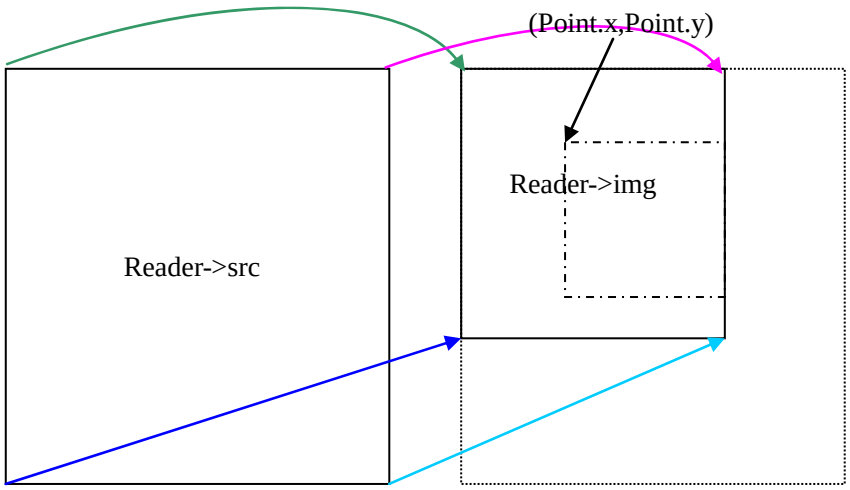


图 3.2 抠图示意图

4. 训练

准备好正样本集（即 samples.vec 文件），负样本集及其描述文件后，就可以看是训练了。从下面的代码中可以看出训练时命令行参数列表。

```
printf( "Usage: %s\n    -data <dir_name>\n"
        "    -vec <vec_file_name>\n"
        "    -bg <background_file_name>\n"
        "    [-npos <number_of_positive_samples = %d>]\n"
        "    [-nneg <number_of_negative_samples = %d>]\n"
        "    [-nstages <number_of_stages = %d>]\n"
        "    [-nsplits <number_of_splits = %d>]\n"
        "    [-mem <memory_in_MB = %d>]\n"
        "    [-sym (default)] [-nonsym]\n"
        "    [-minhitrate <min_hit_rate = %f>]\n"
        "    [-maxfalsealarm <max_false_alarm_rate = %f>]\n"
        "    [-weighttrimming <weight_trimming = %f>]\n"
        "    [-eqw]\n"
        "    [-mode <BASIC (default) | CORE | ALL>]\n"
        "    [-w <sample_width = %d>]\n"
        "    [-h <sample_height = %d>]\n"
        "    [-bt <DAB | RAB | LB | GAB (default)>]\n"
        "    [-err <misclass (default) | gini | entropy>]\n"
        "    [-maxtreesplits <max_number_of_splits_in_tree_cascade = %d>]\n"
        "    [-minpos <min_number_of_positive_samples_per_cluster = %d>]\n",
        argv[0], npos, nneg, nstages, nsplits, mem,
        minhitrate, maxfalsealarm, weightfraction, width, height,
        maxtreesplits, minpos );
```

如果在 VC 中进行调试，可采用第 3 章介绍的方法设置命令行参数，我在调试时设置的参数如下：

-data F:\FaceDetect\trainout -vec F:\FaceDetect\samples.vec -bg F:\FaceDetect\negatives.dat -nstages 3 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 100 -nneg 200 -w 20 -h 20 -mem 512 -eqw 1 -mode ALL -bt GAB -minpos 50

为调试方便，正负样本数目用的非常少，正样本 100 个，负样本 200 个。通过查看 cvhaartraining.h 文件的 cvCreateCascadeClassifier 函数参数说明可进一步了解上述参数的含义：

```
/*
 * cvCreateCascadeClassifier
 *
 * Create cascade classifier
 * dirname                - directory name in which cascade classifier will be created.
 *     It must exist and contain subdirectories 0, 1, 2, ... (nstages-1).
 * vecfilename            - name of .vec file with object's images
```

```
* bgfilename             - name of background description file
* npos                   - number of positive samples used in training of each stage
* nneg                   - number of negative samples used in training of each stage
* nstages                - number of stages
* numprecalculated - number of features being precalculated. Each precalculated feature
*     requires (number_of_samples*(sizeof( float ) + sizeof( short ))) bytes of memory
* numsplits              - number of binary splits in each weak classifier
*     1 - stumps, 2 and more - trees.
* minhitrate             - desired min hit rate of each stage
* maxfalsealarm          - desired max false alarm of each stage
* weightfraction         - weight trimming parameter
* mode                   - 0 - BASIC = Viola
*                            1 - CORE    = All upright
*                            2 - ALL     = All features
* symmetric              - if not 0 vertical symmetry is assumed
* equalweights           - if not 0 initial weights of all samples will be equal
* winwidth               - sample width
* winheight              - sample height
* boosttype              - type of applied boosting algorithm
*     0 - Discrete AdaBoost
*     1 - Real AdaBoost
*     2 - LogitBoost
*     3 - Gentle AdaBoost
* stumperror             - type of used error if Discrete AdaBoost algorithm is applied
*     0 - misclassification error
*     1 - gini error
*     2 - entropy error
*/
```

4.1 训练的总体流程

图 4.1 是 HaarTraining 训练的一个简单流程。这个流程之所以是简单流程，因为只有当用户是要建一个简单的级联分类器时才是这么一个流程。对于建树形强分类器，其流程比这个要复杂，暂时没有去梳理它的流程。

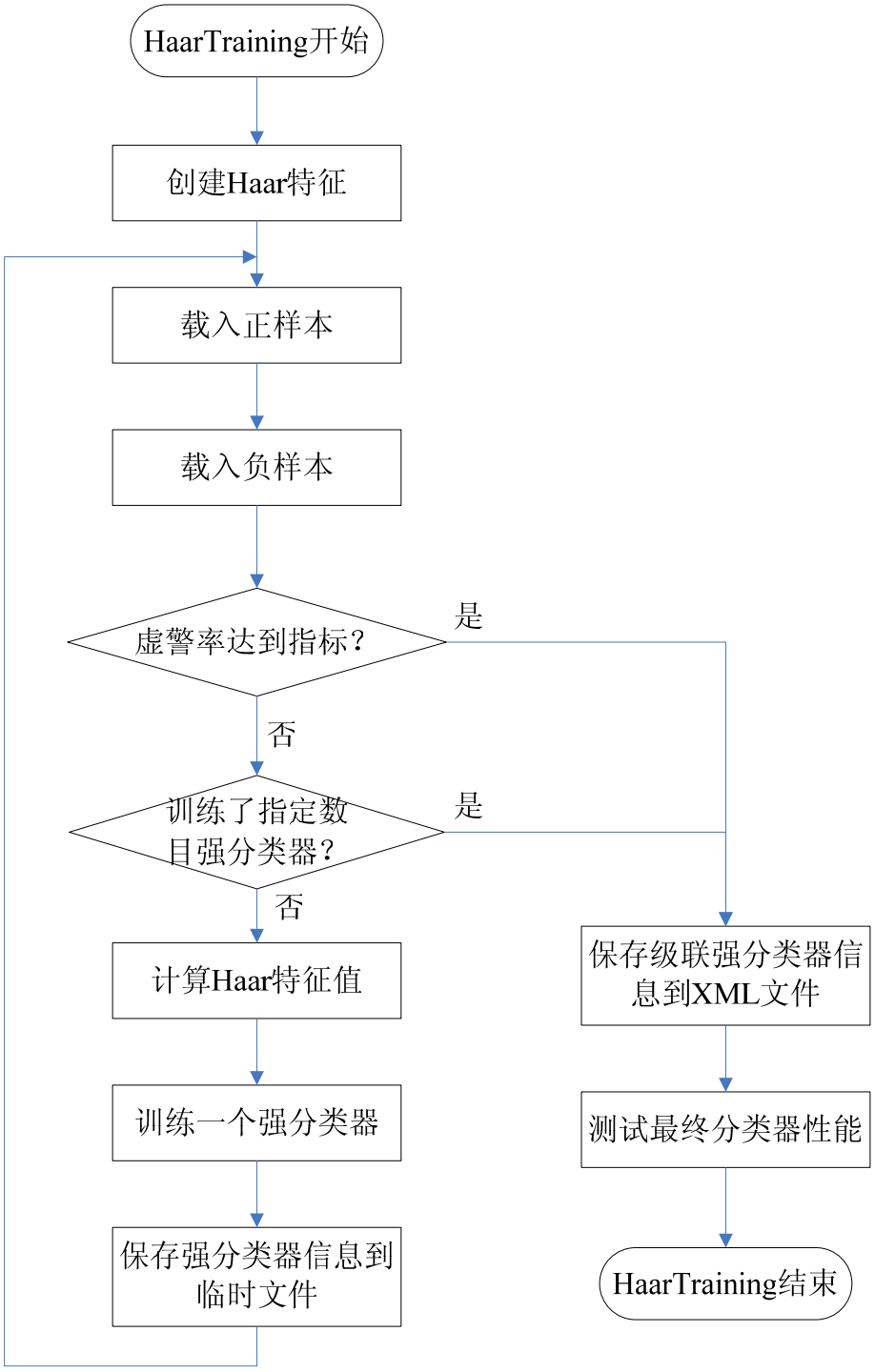
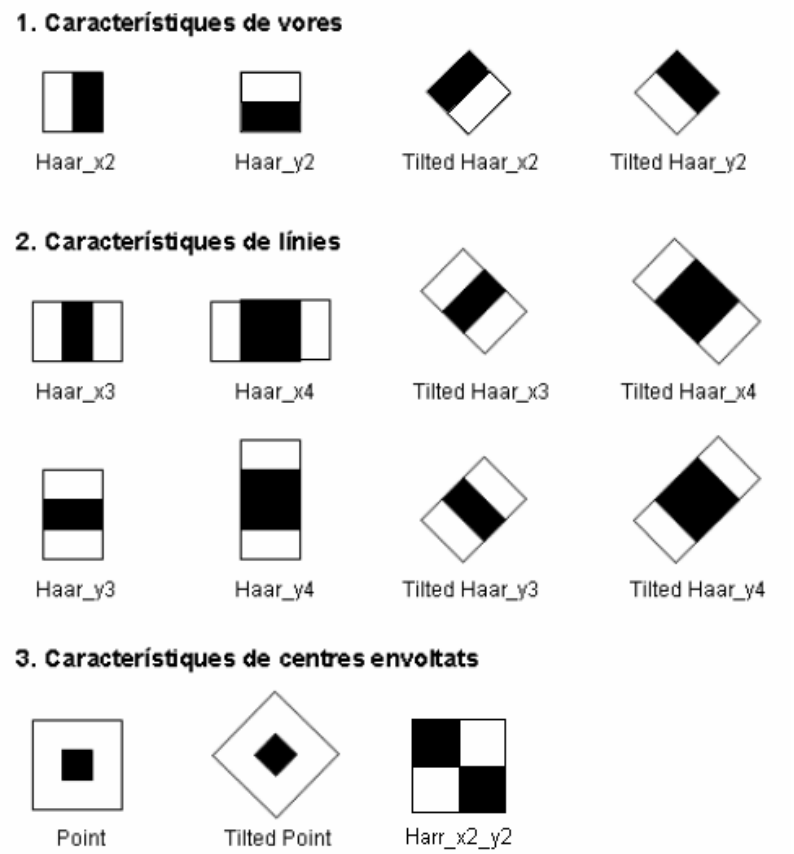


图 4.1 HaarTraining 训练流程

4.2 创建 Haar 特征

函数 icvCreateIntHaarFeatures(winsize, mode, symmetric)负责创建所有可能的 Haar 特征。Mode 决定使用基本的 5 种特征还是所有 upright 特征抑或所有特征。Symmetric 为 1 时表示只创建 Haar 特征的中心在左半部分的所有特征，为 0 时创建所有特征。当训练人脸图像时，由于人脸的左右对称性可以设置 Symmetric 为 1，以加速训练。

在创建特征时，OpenCV 对每种特征进行了文字描述，具体如图 4.2 所示。



在该函数中，有一段代码感觉有点问题：

```
if ( (x+dx*2 <= winsize.height) && (y+dy <= winsize.width) ) {
    if (dx*2*dy < s0) continue;
    if (!symmetric || (y+y+dy <= winsize.width)) {
        haarFeature = cvHaarFeature( "haar_y2",
            y, x,       dy, dx*2, -1,
            y, x+dx, dy, dx,      +2 );
        CV_WRITE_SEQ_ELEM( haarFeature, writer );
    }
}
```

从上面代码可看出，haar_y2 特征与 haar_x2 关于对角线对称，当 winsize 宽高不等时，haar_y2 不能全部遍历到???。当然，对于宽高相等的情况不会有问题。

另外，对于旋转 Haar 特征的旋转矩形区域定义也有点疑惑，尤其是结合后面的积分图像计算和用积分图像求任意旋转矩形区域像素和时，总觉得不是很完美。

4.3 载入正样本

```
int icvGetHaarTrainingDataFromVec( CvHaarTrainingData* data, int first, int count,
                                   CvIntHaarClassifier* cascade,
                                   const char* filename,
                                   int* consumed )
```

函数 icvGetHaarTrainingDataFromVec()负责从正样本集*.vec 文件中载入 count 个正样本。在程序第一次运行到此（即训练第一个分类器之前）时，只要正样本集中有 count 个样本，就一定能取出 count 个正样本。在以后运行到此时，有可能取不到 count 个样本，因为必须是用前面的级联强分类器分类为正样本（即分类正确的样本）的样本才会被取出作为下一个强分类器训练样本，具体可参考 icvGetHaarTrainingData 和 icvEvalTreeCascadeClassifierFilter 函数。

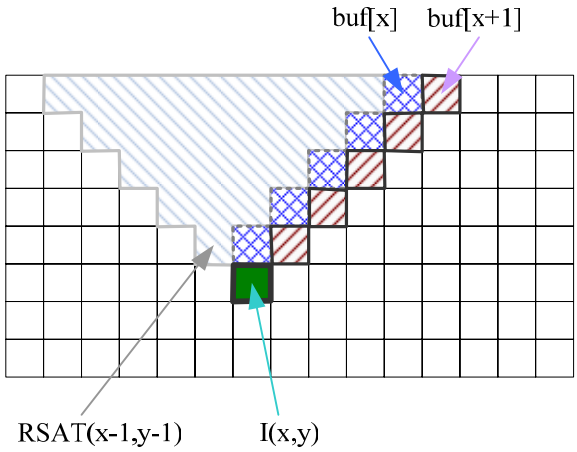
传递返回值的 Consumed 参数表示为取 count 个正样本，查询过的正样本总数。

此外，函数内还通过调用 icvGetAuxImages 计算积分图像。

在此需要特别说明的是旋转 Haar 特征所用积分图像 RSAT 的计算方法。OpenCV 代码中的方法与其文章中的方法有些差异，效率比文章中方法更高。在计算 RSAT 时，代码中采用的是如下公式：

$$RSAT(x,y) = RSAT(x-1,y-1) + I(x,y) + buf[x] + buf[x+1]$$

其中 $buf[x]$ 是对角线像素灰度值之和，如下图所示：



此外，计算归一化因子时需要注意，OpenCV 没有用样本所有像素去计算标准差，而是去除了边界一圈像素（normrect = cvRect(1, 1, img->cols - 2, img->rows - 2)），并且归一化因子是标准差和归一化区域面积的乘积((*normfactor) =sqrt(valsqsum / area - (valsum / are)^2) * area)，这样就去除了在检测时目标与样本大小不同的影响。

4.4 载入负样本

int icvGetHaarTrainingDataFromBG(CvHaarTrainingData* data, int first, int count, CvIntHaarClassifier* cascade, double* acceptance_ratio)

函数 icvGetHaarTrainingDataFromBG ()负责从负样本集中载入 count 个负样本。在程序第一次运行到此（即训练第一个分类器之前）时，只要负样本集中有 count 个样本，就一定能取出 count 个负样本。在以后运行到此时，有可能取不到 count 个样本，因为必须是用前面的级联强分类器分类为正样本的样本（即分类错误的样本）才会被取出作为下一个强分类

器训练样本，具体可参考 icvGetHaarTrainingDataFromBG 和 icvEvalTreeCascadeClassifierFilter 函数。

传递返回值的 acceptance_ratio 参数记录的是实际取出的负样本数与查询过的负样本数之比（acceptance_ratio = ((double) count) / consumed_count），也就是虚警率，用于判断已训练的级联分类器是否达到指标，若达到指标，则停止训练过程。

此外，函数内还通过调用 icvGetAuxImages 计算积分图像。

注意函数 icvGetHaarTrainingDataFromBG 中一个主要的 For 循环：

```
for( i = first; i < first + count; i++ ) //共读取 count 个负样本,当读取不到
{                                        //这么多负样本时将出现死循环！
```

对上面代码中的注释有必要进一步说明一下：只有当之前的强分类器对负样本集内的样本全部分类正确时才会出现死循环。因为只要有一个样本会被错分为正样本，那么通过 count 次扫描整个负样本集就能得到 count 个负样本，当然这 count 个负样本实际上就是一个负样本的 count 个拷贝。为避免这些情况的发生，负样本集中的样本数需要足够多。

在负样本图像大小与正样本大小完全一致时，假设最终的分类器虚警率要求是 falsealarm，参加训练的负样本要求是 count 个，则需要的负样本总数可计算如下：

TotalCount = count / falsealarm

以 Rainer Lienhart 的文章中的一些参数为例，falsealarm=0.5^20=9.6e-07, count=3000，则 TotalCount=3000/(0.5^20)= 3,145,728,000=31 亿。

当正负样本顺利载入，屏幕上会出现类似下面的输出界面：

```
*** 1 cluster ***
POS: 100 100 1.000000
NEG: 200 0.379507
```

含义：

POS(正样本): 取出的正样本数目　　查询过的正样本数目 两者之比

NEG(负样本): 取出的负样本数目　　查询过的负样本数目 两者之比

上面的输出由下面两行代码得到：

```
{
        printf( "POS: %d %d %f\n", poscount, consumed, ((double) poscount)/consumed );
        printf( "NEG: %d %g\n", negcount, false_alarm );
        // false_alarm    = ((double) negcount) / consumed_negcount;
}
```

4.5 计算 Haar 特征值

```
void icvPrecalculate( CvHaarTrainingData* data, CvIntHaarFeatures* haarFeatures,
                      int numprecalculated )
```

函数 icvPrecalculate ()负责计算所有取出的正负样本的前 numprecalculated 个 Haar 特征

值（由 icvGetTrainingDataCallback 实现），并且对每种特征，将所有样本标号按其特征值升序排序（由 cvGetSortedIndices 实现，每种特征分别排序）。

Numprecalculated 的计算公式如下：

```
numprecalculated = (int) ( ((size_t) mem) * ((size_t) 1048576) /
    ( ((size_t) (npos + nneg)) * (sizeof( float ) + sizeof( short )) ) );
```

其中 mem 是内存大小，以 M 为单位，1048576=1024*1024，表示 1M 字节。sizeof(float) 为保存一个特征值需占用的字节数，sizeof(short)表示对特征值排序后保存一个排序序号需占用的字节数。

4.6 训练一个强分类器

```
CvIntHaarClassifier* icvCreateCARTStageClassifier( CvHaarTrainingData* data,
                                                   CvMat* sampleIdx,
                                                   CvIntHaarFeatures* haarFeatures,
                                                   float minhitrate,
                                                   float maxfalsealarm,
                                                   int      symmetric,
                                                   float weightfraction,
                                                   int numsplits,
                                                   CvBoostType boosttype,
                                                   CvStumpError stumperror,
                                                   int maxsplits )
```

函数 icvCreateCARTStageClassifier 负责训练一个强分类器。

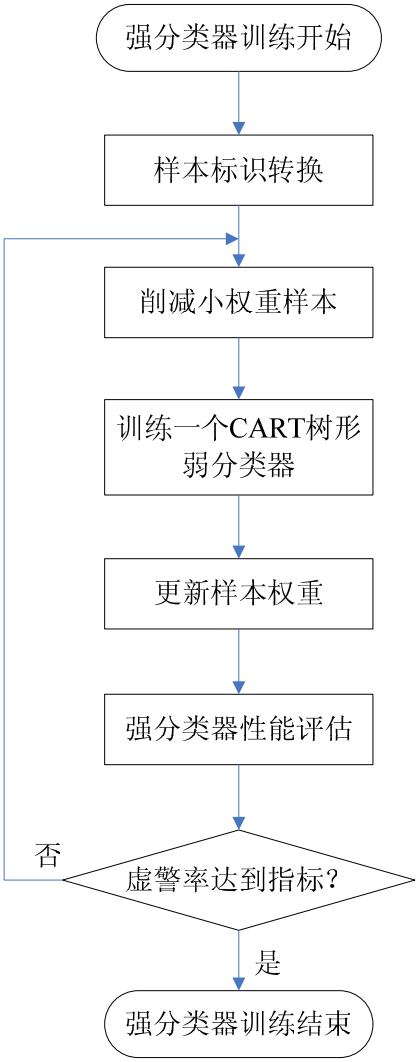


图 4.2 单个强分类器训练流程

4.7 保存强分类器信息到临时文件中

函数 icvSaveStageHaarClassifier 负责将新训练得到的强分类器信息保存到临时文件 AdaBoostCARTHaarClassifier.txt 中。icvSaveStageHaarClassifier 首先根据当前强分类器在级联强分类器中序号(cur_node->idx)在 dirname 目录下创建一个文件夹，然后在该文件夹下创建 AdaBoostCARTHaarClassifier.txt 文件，并将强分类器信息写到这个文件中。

4.8 将级联强分类器信息写到一个 XML 文件中

首先是从先前保存的临时文件中读取级联强分类器信息（cascade = cvLoadHaarClassifierCascade(dirname, cvSize(winwidth,winheight)) ）然后用 cvSave(xml_path, cascade)将级联强分类器信息保存到 xml 文件中。至此，整个训练部分完毕。

4.9 测试最终分类器性能

这部分工作通过简单地调用两个函数实现。调用 icvGetHaarTrainingDataFromVec 测试检出率；通过调用 icvGetHaarTrainingDataFromBG 测试虚警率。

5. 参考文献

- Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. International Journal of Computer Vision, Vol. 57, pp.137-154, May 2004.
- Richard O. Duda, Peter E. Hart, David G. Stork. Pattern Classification, 2nd Edition, November 2000.
- Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.
- Friedman, J. H., Hastie, T. and Tibshirani, R. Additive Logistic Regression: a Statistical View of Boosting. Technical Report, Dept. of Statistics, Stanford University, 1998.
- OpenCV Source Code, Intel, 2007.