# 一 haar训练

## 1.1 正样本标注

参考<https://github.com/takmin/ObjectMarker>的Object Marker实现.

执行标注exe后, 输入待处理的图像路径.

然后会显示出待处理的原图. 鼠标在原图上画矩形框.

画好后, 按回车键确认.

程序自动打开下一张图片.

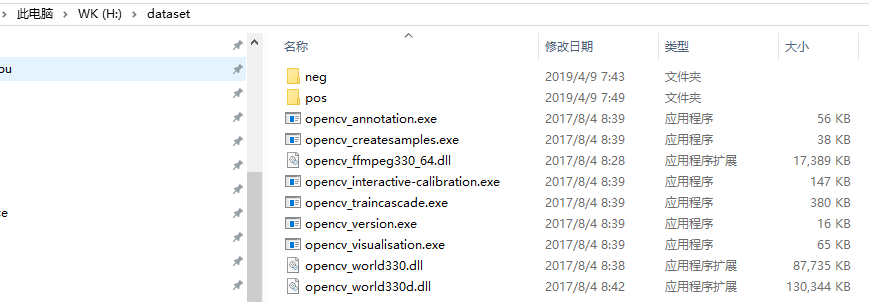
## 1.2 正样本生成对应的vec文件

首先利用generate\_haar\_pos.py生成正样本及其label

pos.txt格式如下:

|  |
| --- |
| Aaron\_Eckhart\_0001.jpg 1 84 161 92 169 |

pos.txt和图像放在一起. 同属于pos路径.



|  |
| --- |
| Cp G:\tools\openCV-3.3\opencv\build\x64\vc14\bin\\* H:\dataset |

Cmd打开H:dataset路径,并执行

|  |
| --- |
| **opencv\_createsamples.exe** -info pos/pos.dat -vec pos/sample\_pos.vec -num 17 -w 20 -h 20 –show YES  参数说明：-info，指样本说明文件  　　　　　　-vec，样本描述文件的名字及路径  　　　　　　-num，总共几个样本，要注意，这里的样本数是指标定后的20x20的样本数，而不是大图的数目，其实就是样本说明文件第2列的所有数字累加　　　　　　　　　和。  　　　　　　-w -h 指明想让样本缩放到什么尺寸。这里的奥妙在于你不必另外去处理第1步中被矩形框出的图片的尺寸，因为这个参数帮你统一缩放！  　　　　　　-show 是否显示每个样本。样本少可以设为YES，要是样本多的话最好设为NO，或者不要显式地设置，因为关窗口会关到你哭 |

设置400个正样本:

|  |
| --- |
| cd D:\work\stuff\modules\misc\sprd\_camera\alg\july\matlab\cvaa\cc\haar\_like\haar\_like\dataset\  opencv\_createsamples.exe -info ./pos/pos.dat -vec ./pos/pos.vec -num 400 -w 20 -h 20 |

## 1.3 understanding sample generation

总结并延伸以上内容：

1. 样本图片最好使用灰度图，且最好能根据实际情况做一定的预处理
2. 样本选择的原则是：数量越多越好，尽量高于1000；样本间差异性越大越好
3. 正负样本比例为1：3最佳，尺寸为20x20最佳

# 二 AdaBoost级联分类器

**为了解决什么问题.**

做决定时,会考虑多个专家的意见.反应到机器学习上,也是考虑多个分类器的意见. 这个思想(方法)就是元算法meta algorithm(集成方法ensemble method).

元算法是对其他算法的一种组合方式.

Adaboost是元算法中比较火的一个方法.

Adaboost是(属于)监督学习方法.

**算法集成**

1. 可以不同算法的集成(ensemble)
2. 也可以是同一个算法不同配置的集成.
3. 也可以是数据集的不同部分分给不同分类器(算法)的集成.

Adaboost优缺点

优点)

泛化能力强, 无参数调整.

缺点)

对离群点敏感.

适用于数值型和标称型数据.

## 2.1 bagging和boosting

**Bagging**

自举汇聚法(bootstrap aggeregating)也称为bagging.

是将原数据集选择S次后,得到S个新数据集的技术. 让后对S个数据集分类算法,得到S个分类器. 分类器投票结果最多的作为最终的结果.

**Boosting**

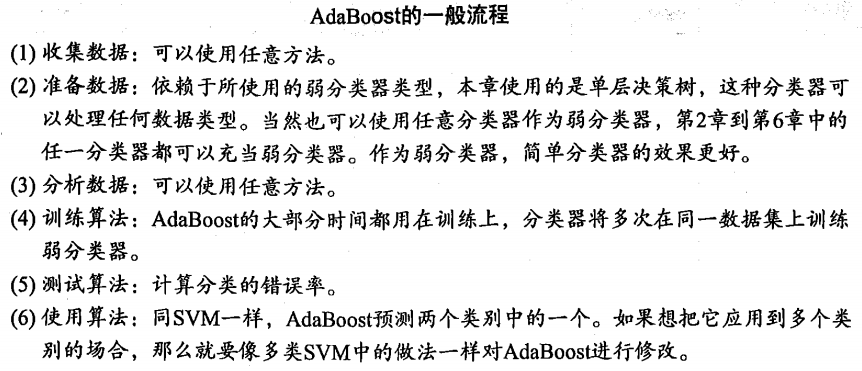
首先,boosting和bagging类似. 表现在使用的分类器类型一致.

然后,不同点)

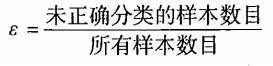
1. boosting的分类器是通过串行训练得到的. 新分类器依据已训练出的分类器的性能来进行训练.
2. Boosting结果是所有分类器的加权求和结果. Bagging的权重是相等的. Boosting的权重是不等的.每个权重代表上次迭代的成功度.

Adaboost是Boosting的一种.

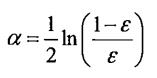
## 2.2 Adaboost



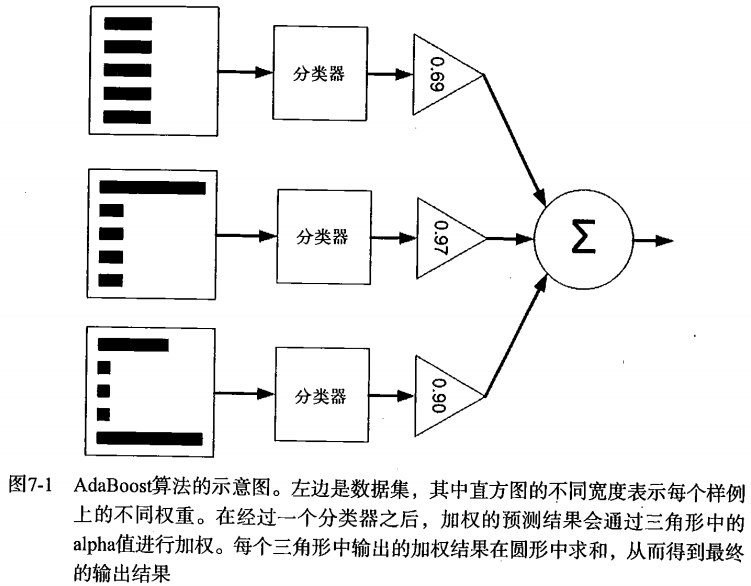
1. 弱分类器组合构成强分类器.
   1. 弱分类器是指”性能比随机猜测要好些但有限”
   2. 弱分类器在二分类中错误率会高于50%.
2. Adaboost是adaptive boost缩写.
   1. 对train数据集每个样本分配一个权重.
   2. 对这些样本训练出一个弱分类器, 并计算器错误率(参考accuracy,是accuracy的反面).
   3. 然后调整样本的权重, 上次弱分类器分对的样本的权重减小. 分错的样本的权重增大.
   4. 然后再训练一个分类器.
3. 错误率公式:



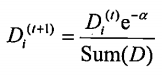
1. 分类器的权重alpha公式**(区别于样本的权重D)**.



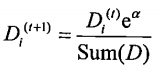
1. Adaboost如下图
   1. 0.69, 0.97, 0.90等是分类器的权重.



1. 样本权重
   1. 正确分类的样本

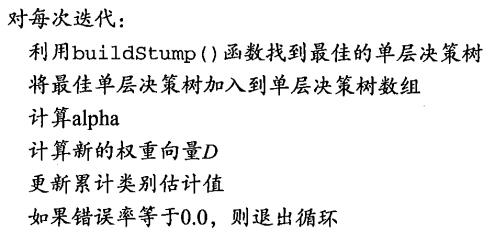


* 1. 错误分类的样本权重更新



# 三 Adaboost实例

伪代码:



## 3.1 训练权重更新

|  |
| --- |
| **def** adaBoostTrainDS**(**dataArr**,**classLabels**,**numIt**=**40**):**  weakClassArr **=** **[]**  m **=** shape**(**dataArr**)[**0**]**  D **=** mat**(**ones**((**m**,**1**))/**m**)** #init D to all equal  **## 开始的样本权重是一样的**  aggClassEst **=** mat**(**zeros**((**m**,**1**)))**  **for** i **in** range**(**numIt**): ## 迭代40次.**  bestStump**,**error**,**classEst **=** buildStump**(**dataArr**,**classLabels**,**D**)**#build Stump **构建(生成)分类器.**  #print "D:",D.T  alpha **=** float**(**0.5**\***log**((**1.0**-**error**)/**max**(**error**,**1e-16**)))**#calc alpha, throw in max(error,eps) to account for error=0  **## alpha公式. Error是epsilon.**    bestStump**[**'alpha'**]** **=** alpha  weakClassArr**.**append**(**bestStump**)** #store Stump Params in Array  #print "classEst: ",classEst.T  **## 多个弱分类器加权求和作为adaboost的推断**  expon **=** multiply**(-**1**\***alpha**\***mat**(**classLabels**).**T**,**classEst**)** #exponent for D calc, getting messy  ## 计算D更新的权重.   1. **从公式上看出,有部分是相同的.** 2. **不同的部分是exp的次方的正负号.**   D **=** multiply**(**D**,**exp**(**expon**))** #Calc New D for next iteration  D **=** D**/**D**.**sum**()**  #calc training error of all classifiers, if this is 0 quit for loop early (use break)  aggClassEst **+=** alpha**\***classEst  #print "aggClassEst: ",aggClassEst.T  aggErrors **=** multiply**(**sign**(**aggClassEst**)** **!=** mat**(**classLabels**).**T**,**ones**((**m**,**1**)))**  errorRate **=** aggErrors**.**sum**()/**m  **print** "total error: "**,**errorRate  **if** errorRate **==** 0.0**:** **break**  **return** weakClassArr |

## 3.2 推断

推断是多个弱分类器和其alpha权值的加权和,该和值作为adaboost的推断结果.

|  |
| --- |
| **def** adaClassify**(**datToClass**,**classifierArr**):**  dataMatrix **=** mat**(**datToClass**)**#do stuff similar to last aggClassEst in adaBoostTrainDS  m **=** shape**(**dataMatrix**)[**0**]**  aggClassEst **=** mat**(**zeros**((**m**,**1**)))**  **for** i **in** range**(**len**(**classifierArr**)):**  classEst **=** stumpClassify**(**dataMatrix**,** classifierArr**[**i**][**'dim'**],**\  classifierArr**[**i**][**'thresh'**],**\  classifierArr**[**i**][**'ineq'**])**#call stump classify  aggClassEst **+=** classifierArr**[**i**][**'alpha'**]\***classEst  **print** aggClassEst  **return** sign**(**aggClassEst**)** |

# 四 haar及Adaboost训练

工具有

### opencv\_haartraining.exe

负责(集成,包装了) haar特征提取及adaboost分类器. 实现adaboost分类器的训练.

在opencv3.0里已经找不到”opencv\_haartraining.exe”了,取而代之的是opencv\_traincascade工具

1. 支持haar 小波特征.
2. Lbp特征
3. Lbp特征有浮点精度(floating point precision), 比haar特征快.
4. Haar和Lbp的质量都依赖于训练集以及训练参数.

在sources\doc\tutorials\objdetect\traincascade.markdown中.

|  |
| --- |
| If you come accross any tutorial mentioning the ***old*** ***opencv\_haartraining*** tool <i>(which is deprecated and still using the OpenCV1.x interface)</i>, ***then please ignore that tutorial and stick to the opencv\_traincascade tool***. This tool is a newer version, written in C++ in accordance to the OpenCV 2.x and OpenCV 3.x API. The opencv\_traincascade supports both HAAR like wavelet features @cite Viola01 and LBP (Local Binary Patterns) @cite Liao2007 features. LBP features yield integer precision in contrast to HAAR features, yielding floating point precision, so both training and detection with LBP are several times faster then with HAAR features. Regarding the LBP and HAAR detection quality, it mainly depends on the training data used and the training parameters selected. It's possible to train a LBP-based classifier that will provide almost the same quality as HAAR-based one, within a percentage of the training time. |

在G:\tools\openCV-3.3\openCV--vs17--code\bin\Debug目录下有apps.

命令如下:

|  |
| --- |
| D:\work\stuff\modules\misc\sprd\_camera\alg\july\matlab\cvaa\cc\haar\_like\haar\_like\x64>G:\tools\openCV-3.3\openCV--vs17--code\bin\Debug\opencv\_traincascaded.exe -data cascade -vec D:\work\stuff\modules\misc\sprd\_camera\alg\july\matlab\cvaa\cc\haar\_like\haar\_like\dataset\pos\sample\_pos.vec -bg D:\work\stuff\modules\misc\sprd\_camera\alg\july\matlab\cvaa\cc\haar\_like\haar\_like\dataset\neg\neg\_sample.dat -numPos 20 -numNeg 60 -mode ALL -w 20 -h 20 |

### Haarconv.exe或者convert\_cascade.exe

# 五 何为haar特征