2016-3-30

HOMEWORK4

基于 Adaboost 与 HAAR 特征的人脸识别

汪思学 1300012784

Cecil Wang
MICROSOFT

目录

一、Adaboost

1.1 简介

Adaboost 是英文“Adaptive Boosting”的缩写，由 Yoav Freuud 和 Robert Schapire 在 1995 年提出。它本身是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器，即弱分类器，然后把这些弱分类器集合起来，构造一个更强的最终分类器。实际上这种方法是从 Boosting 算法上改进得到的，相比于 Boosting 算法，Adaboost 通过当前的错误分布来调整每个样本的权重与最终分类器的权重。

在 Deep Learning 火爆之前，Adaboost 与 SVM 在分类学习算法中算是佼佼者。虽然 SVM 在诸多方面往往强于 Adaboost，但是 Adaboost 也有其自身的优点，例如多标签分类比 SVM 简单的多，同时它不会过拟合。为了凸显 Adaboost 的优点，我在这次作业中选择了人脸识别这个主题，可以说 Adaboost 与 HAAR 的结合在人脸识别中是一套非常成熟的框架。

1.2 算法流程

简单地讲，Adaboost 需要若干弱分类器，和一组正负样本。初始，将样本的权重设为相等。对于第 t 次迭代，根据样本权重计算每个弱分类器的错误率，选取其中错误率最小的一个，然后根据此分类器的结果，将识别正确的样本权重降低，将识别错误的样本权重提高，迭代进行下一次。

下面我们进行半形式化规约，给定一个训练数据集

$$\text{Sample} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

其中实例$x \in \chi \subset R^N$， $y \in \{1, -1\}$

和弱分类器

$$G = G_t(x): \chi \to \{1, -1\}, t = 1,2, \dots, T$$

其中 T 为最大迭代次数，G_t为第 T 次迭代时的分类器集合

算法流程如下：

- 1 初始化训练样本数据的权值分布，每一个训练样本都被赋予相同的权重1/N

$$D_1 = (w_{11}, w_{12}, \dots, w_{1i}, \dots, w_{1n}), w_{11} = \frac{1}{N}, i = 1,2, \dots N$$

- 2 For t = 1,2,3,…,T
  - 使用具有权值分布D_t的训练集来学习

$$G_t(\mathrm{x}): \chi \to \{1, -1\}$$

  - 计算$G_t(\mathrm{x})$在训练集上的分类误差率

$$e_t = P(G_t(x_i) \neq y_i) = \sum_{i=1}^{N} w_{ti} I(G_t(x_i) \neq y_i)$$

  - 计算$G_t(\mathrm{x})$在最终分类器中的权重系数

$$\alpha_t = \frac{1}{2} log \frac{1 - e_t}{e_t}$$

    (注：由上述式子可知，e_t<= $\frac{1}{2}$时，α_t >= 0，且α_t随着e_t的减小而增大，意味着分类误差率越小的弱分类器在最终分类器中的作用越大。)

  - 更新训练集的权值分布

$$D_{t+1} = \left(w_{t+1,1}, w_{t+1,2}, \dots, w_{t+1,i}, \dots, w_{t+1,n}\right)$$

$$w_{t+1,i} = \frac{w_{t,1}}{Z_t} \exp\left(-\alpha_t y_i G_t(x_i)\right), \mathrm{i} = 1,2, \dots, \mathrm{N}$$

    (注：由上述式子可知，被弱分类器G_t误分类样本的权值增大，而被正确分类样本的权值减少，通过这样的方式，AdaBoost 方法能“聚焦于”那些较难分的样本上。)

    其中Z_t为规范因子

$$Z_t = \sum_{i=1}^{N} w_{ti} \exp\left(-\alpha_t y_i G_t(x_i)\right)$$

- 3 构建基本分类器的线性组合

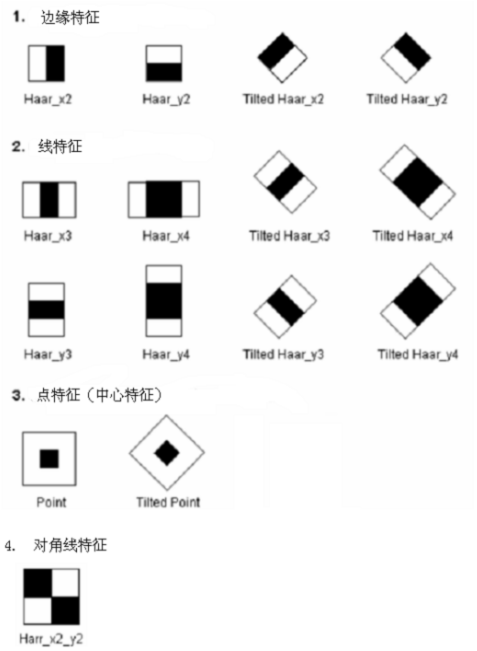
$$\mathrm{f(x)} = \sum_{t=1}^{T} \alpha_t G_t(\mathrm{x})$$

从而得到最终分类器，如下：

$$G(x) = sign\left(f(x)\right) = sign(\sum_{t=1}^{T} \alpha_t G_t(\mathrm{x}))$$

二、HAAR-like 特征

Haar-like 特征最早是由 Papageorgiou 等应用于人脸表示。Haar 特征分为四类：边缘特征、线性特征、中心特征和对角线特征。



特征模板内有白色和黑色两种矩形，并定义该模板的特征值为白色矩形像素和减去黑色矩形像素和。通过改变特征模板的大小和位置，可在图像子窗口中穷举出大量的特征。

HAAR-like 特征之所以常用与人脸识别是因为，HAAR 特征值反映了图像的灰度变化情况，而脸部的一些特征能由矩形特征简单的描述，如：眼睛要比脸颊颜色要深，鼻梁两侧比鼻梁颜色要深，嘴巴比周围颜色要深等。但矩形特征只对一些简单的图形结构，如边缘、线段较敏感，所以只能描述特定走向（水平、垂直、对角）的结构。

三、实现细节

3.1 ReadImgSet.m

读取文件夹下的一组图片

```
function [ImgSet, Succ] = ReadImgSet(Path, Total, type)
% ImgSet = ReadImg(Path, Num)
% Read all images from the Path
% ImgSet 为返回的图片数组
```

```
% succ  函数是否成功执行
% Path 图片所在文件夹
% Total  读取的数量
```

读取的图片会统一缩放到 30X30 大小。

为了保证函数健壮性，会首先判断文件夹是否存在

```
if ~exist(Path, 'dir')
   disp('No such folder') ;
   ImgSet=0;
   Succ=0;
   return;
end
```

之后通过 dir 函数来获得所有候选图片

```
imgList=dir(Path);
```

然后读取图片并将其转化为灰度图，最后将图片的积分图算出

```
ImgSet(:,:,num)=cumsum( cumsum( img, 1 ), 2 );
```

3.2 CreatHaarFeature.m

创建在 30X30 图片中所有的 HAAR 特征。

如之前所述，我们将特征模板在 30X30 的图像中通过改变位置大小得到所有的特征，为了方便我们选择了如下四个特征模板

1111	1100	0110	0000
1111	1100	0110	1111
0000	1100	0110	1111
0000	1100	0110	0000

在此只选择第一种特征模板的代码进行详述，其他模板构造方式类似

```
function f = CreatHaarFeature(h,w,frange)
% CreatHaarFeature = (h,w,frange)
% h,w 分别为图像高、宽，我的例子中为 30X30
% frange 表示 haar 特征的长宽范围
```

为了数据复用，每次结果会单独保存下来，如果以后需要用直接读取相应 mat 文件即可

```
dataName=['data/Feature',mat2str(h),'-',mat2str(w),'-
',mat2str(frange(1)),'-',...
            mat2str(frange(2)),'-',mat2str(frange(3)),'-
',mat2str(frange(4)),'.
                    mat'];
if exist(dataName, 'file')
    load(dataName);
    return;
```

```
end
```

之后通过枚举特征的高宽，和位置来生成特征。为了以后在积分图中方便使用，我们将所有特征存在一个 cell 元组中（f），具体的每个特征我们记录每个黑白块的矩形的四个角点以及黑白对应 1、-1。

```
f = {};
%% horizontal
% 枚举高
for fh = frange(1):frange(2)
    % 枚举宽
    for fw = frange(3):frange(4)
        %生成每个坐标的索引矩阵
        [x,y]   = meshgrid( 1:(w-fw), 1:(h-fh) );
        %通过之前的索引矩阵枚举左上角点的坐标，以此枚举每个位置的特征
        for dx = 1:(fh-1)
            for idx = 1:length(x(:))
                %上半部分四个角点的 x 坐标
                bx      = [x(idx) x(idx)+fw x(idx) x(idx)+fw];
                %上半部分四个角点的 Y 坐标
                by      = [y(idx) y(idx) y(idx)+dx y(idx)+dx];
                %上半部分的四个角点在 30X30 图像中的索引值
                top     = sub2ind([h w],by,bx);
                %存储下半部分的四个角点在 30X30 图像中的索引值
                bot     = sub2ind([h w],[by(1:2)+dx by(3:4)+(fh-
dx)],bx);
                %存储，翻转再存储
                f       = [f {[top -1; bot 1]} {[top 1; bot -1]}];
            end
        end
    end
end
```

3.3 adaBoost.m

Adaboost 算法实现代码

```
function classifier = adaBoost(sample,sampleLable,feature,T,NPos,NNeg)
% classifier = adaBoost(sample,sampleLable,feature,T)
% sample 训练样本
% sampleLable  样本标签
% feature  候选 haar 特征，即弱分类器
% T  需要迭代的次数
% NPos  正样本数
% NNeg  负样本数
```

计算每个 Haar 特征在训练样本上对应的特征值

```
%% calculate the feature result
nsample=length(sampleLable);
nfeature=length(feature);
result = zeros([nfeature nsample]);

    for i = 1:nfeature
        for j = 1:nsample
            img = sample(:,:,j);
            onefeature = feature{i};
            coordinate = onefeature(:,1:4);
            tmp = img(coordinate(:));
            tmp = tmp(1:4:end) - tmp(2:4:end) - tmp(3:4:end) +
tmp(4:4:end);
            tmp = tmp .* onefeature(:,5);
            result(i,j) = sum( tmp );
        end
    end

minresult = min(result,[],2);
maxresult = max(result,[],2);
deltaresult = (maxresult-minresult)/divideFactor;
```

计算每个 Haar 特征的分类误差

```
%% select the weak classifier
weights = zeros(nsample,1);
weights(:)=1.0/nsample;

for t = 1:T

    %% calculate the error and choose the minimum error
    weights = weights / sum(weights);

    %候选的 threshold 值
    tmp = min(minresult) : min(deltaresult) : max(maxresult);
    recorder = zeros(length(tmp),4);
    %枚举分类器为正分类器还是负分类器
    for p_i = -1 : 2 : 1
        %枚举每个 threshold
        for i = 1:length(tmp)
            theta_i = tmp(i);
            ht = ones(size(result));
            ht(find(p_i*result >= p_i*theta_i)) = -1;
            error = abs(ht - ones(length(feature),1)*sampleLable)/2;
            error = error*weights;
            [minerror, index] = min(error);
            %记录此情况下的误差
            recorder(i,:) = [minerror index p_i theta_i];
        end
    end
```

```
    %选择误差最小的分类器
    [minerror, index] = min(abs(recorder(:,1))) ;
    tmpval = minerror / (1.0 - minerror);
    if (tmpval == 0) tmpweight = 1.0; else tmpweight = 0.5 * log
(1.0/tmpval);
        end

    f_ = recorder(index,2);
    p_ = recorder(index,3);
    th_ = recorder(index,4);
```

更新权重

```
    %% update weights
    ht = ones(size(sampleLable));
    ht(find(p_ * result(f_,:) >= p_ * th_)) = -1;

    Z = sum(weights(find((sampleLable - ht) == 0))) * exp(-
tmpweight)...
        + sum(weights(find((sampleLable - ht) ~= 0))) * exp(tmpweight);

    for i = 1 : nsample
        if(sampleLable(i) ~= ht(i))
           weights(i) = weights(i)*exp(tmpweight)/Z;
        else
           weights(i) = weights(i)*exp(-tmpweight)/Z;
        end
    end
```

3.3 其他

train.m 训练脚本，其中数据集路径为 img/train/face 和 img/train/other

detect.m 识别脚本，其中数据集路径为 img/test/face 和 img/test/other