模式识别导论上机题6-AdaBoost

薛犇 1500012752

1. 程序实现说明

本次实验采用Matlab作为编程语言，使用的版本为2016b。

(1) 读取数据

由于样本过大，训练需要较长的时间，所以本部分把training和testing分成了两个程序。

由于两个数据集使用的方法类似，所以我选择新的较大的数据集来作为示例，说明我的程序实现。

training运行 AdaBoosting_1.m(对于新的数据集，运行AdaBoosting_2.m), testing运行 AdaBoostingTest.m

在实验的一开始，读取hw6_data_new.mat中的数据，保存在变量 raw_data中。之后读取样本。 实现如下：

```
% prepare data
raw_data = load('hw6_data_new.mat');
x_train_1 = raw_data.data(1:4000, 1:56);
x_train_2 = raw_data.data(5001:9000,1:56);
x_train = cat(1, x_train_1, x_train_2);
y_train_1 = ones(4000,1);
y_train_2 = ones(4000,1);
y_train_2 = -y_train_2;
y_train = cat(1, y_train_1, y_train_2);
```

(2) 设置adaboost基本的变量

之后就是一些变量的设置：

```
[n, dim] = size(x_train);

errs = []; % error of each Time step
a = []; % weight of classifiers, computed by errs
T = 0; % number of classifiers
d_list = []; % weight of samples
d = ones(n,1); % weight of samples, single step
d = d / n;
h_list = []; % weak classifiers
```

(3) 弱分类器规则

接着开始训练，弱分类器的设置如下：

只能用某一维作为分类标准，相当于分类器的分类线都垂直于超空间的坐标轴，分类标准为：
x(dim)>v分为正样本，x(dim)<v分为负样本, x(dim)代表样本在dim维的分量，v代表一个阈值

当然了，究竟是把"大于阈值"的样本作为正样本，还是"小于阈值"的样本作为负样本，是两种不同的分类策略，而且都是可行的，所以，以下的实验会基于这两个假设，进行不同的训练，最终会发现一些有趣的结论。

每次迭代时的分类器操作如下实现：

```
row_segs = zeros(dim,1);
errs = zeros(dim,1);
[xx_sort, II] = sort(x_train,'descend');
for i=1:dim
    x_sort = xx_sort(:,i);
    I = II(:,i);
    y_sort = y_train(I);
    tmp_d = d(I);
    tmp_pred = ones(n,1);
    tmp_pred = -tmp_pred;
    res_0 = (tmp_pred~=y_sort);
    tmp_err = sum(res_0.*tmp_d);
    col_error = 1;
    row_idx = 1;
    for j =1:n
        tmp_pred(j)=1;
        tmp_err = tmp_err - y_sort(j)*tmp_d(j);
        if tmp_err < col_error
            col_error = tmp_err;
            row_idx=j;
        end
    end
    errs(i) = col_error;
    row_segs(i) = x_sort(row_idx);
end

[error,idx] = min(errs);
```

这段代码的目的是对于每一维，都找一个阈值，使得用这个阈值分出来的效果错误率最低。以上的代码把样本按照降序排列，默认是把"大于阈值"的样本作为正样本。

实现的过程中运用了一些优化，比如：预先排序，求每个阈值的错误率的时候也不是分别求，而是利用了上一个阈值已经计算过的值，所以代码可能会有些不直观。但是最终的目的是为了减少训练时间，因为维数很高，所以需要的分类器也很多，而样本数目很大，所以需要减少单次迭代的训练时间，以得到更多的分类器。事实上，假如暴力得枚举每一种情况的话，在我的机器上训练出仅一个弱分类器就需要50~60s，这是非常高的代价，而进行优化之后，训练一个弱分类器仅仅只要10ms的量级时间。

(4) 保存弱分类器

```
h_list = [h_list;row_segs(idx),idx];
a(T) = 0.5 * log((1-error)/error);
seg = row_segs(idx);
```

将分类器保存在h_list这个数据体中，每个分类器包含两个项

- 分类阈值
- 维度

综合来看，就是在某个维度上设置了一个阈值，单一样本只在这个维度上和阈值比较，得出分类结论，这是在之前就讨论过的。

(5) 更新样本权重

```
pred = sign(x_train(:,idx)-seg+0.00001);
d_list = [d_list;d];
d_new = [];
for i=1:n
    d_new(i)=d(i)*exp(-a(T)*y_train(i)*pred(i));
end
d = d_new / sum(d_new);
d = d';
```

我们根据这个分类器，算出样本的预测值，然后根据adaboost的算法更新样本的权重。

(6) 判断是否结束训练

```
if error < 0.001 || error >= 0.5 || T== weak_classifier_max
break;
end
```

由于实际程序的精度问题，不可能将error降到0，所以设置了阈值0.001。

同样，有的分类器实在太弱，error并降不下来，所以也设置了最大分类器个数weak_classifier_max，之后的实验中取各种数目进行测试与对比。

2. 实验结果分析

(1) old data 100*3

- 策略1：大于阈值的作为正样本

结果不理想，train出了两个错误率为48%和49%的分类器，在test集上的运行结果正确率为50%（并没

有训练出东西）

- 策略2：小于阈值的作为正样本

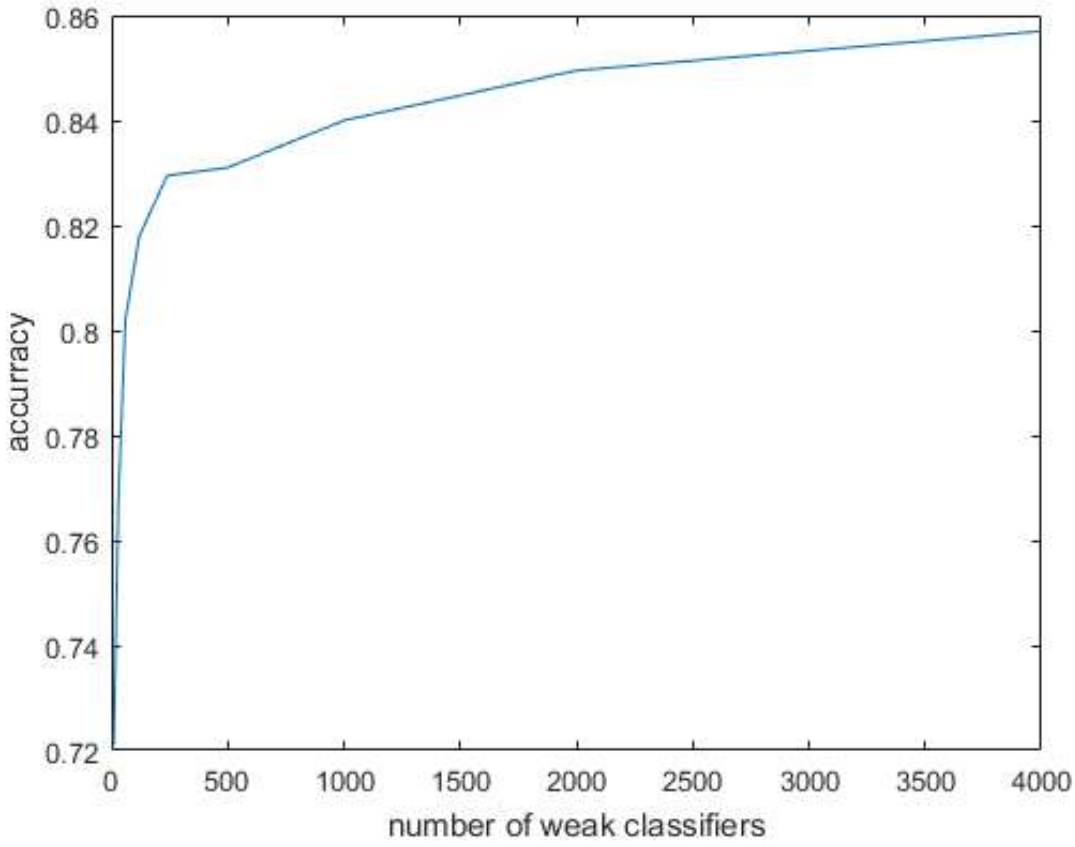
一次训练就得到了一个几乎100%分类正确的分类器，将这个分类器运用到测试集上，正确率为95%（由于程序实现的问题，得出的结果会是准确率5%，取反即可，同样也可以说明，一个每次都预测错误的分类器，其实是很强的毒奶君）。

由于训练样本太小，一两次训练就结束，所以在new data上的训练是重点。

(2) new data 10000*56

- 策略1：大于阈值的作为正样本

事实上，采用这个策略，每次得到的分类器的错误率都在45~48之间徘徊，说明每次都是得到的都是稍微有一点点能力的分类器，这和样本本身的分布也有关系。但是利用这种策略，可以将训练过程一直进行下去，我将分类器的个数设置成10, 30, 60, 120, 240, 500, 1000, 2000, 4000后进行测试，得到的测试结果如下：



发现随着训练器的增加，分类的正确性得到了稳定的提升。

- 策略2：小于阈值的作为正样本

采用这种策略，可以得到少数非常厉害的分类器，几次迭代的结果如下：

```
error =
```

```
    0.2972


error =

    0.1518


error =

    0.0310


error =

    0.0010


error =

   1.0510e-06
```

但是，利用这几个分类器在test集上测试的accuracy只有68%：

```
weak_num =

     6


accurracy =

    0.6895
```

3. 结论

从new data集的实验中可以发现，单个分类器即便很差，但是当数量积累到一定程度的时候，能够达到较好的精度。

与此同时，也似乎可以经验性地总结：对于adaboost来说，单个分类器太强不一定好，因为剩下来的样本就更加难分类了，剩下来的分类器并不能探索到更多的规律，因为这些规律可能超过了它们本身的分类能力。