Dirty Money: Feature selection using AdaBoost

J. van Turnhout (0312649) jturnhou@science.uva.nl,
N. Raiman (0336696) nraiman@science.uva.nl,
S. L. Pintea (6109969) s.l.pintea@student.uva.nl

January 30, 2010

Abstract

In the month January of 2010 a project on the classification of the fitness of money was proposed. During this month we have tested and implemented various techniques to handle the problem of money classification. The results from our experiment show promising development comparing to the current state of research.

In the below sections we have described the approaches that we have tried and the results obtained for each one of them.

1 Introduction

The goal of this project is to determine a reliable method that would be able to distinguish between dirty bills and clean bills. In order to achieve this goal we had to think of what are the representative features of dirt that can be found on money bills and what is the best method that can be used to model this.

Throughout this project we have tried a number of different approaches in order to gain good results and in the same time we have tried to get a better understanding of what methods are fit for describing the features of the dirty money and clean money.

The main techniques used in this project are: *PCA* in combination with *AdaBoost*, *Haar-like features* and *Adaboost*, *Convolution with predefined kernels*, *edge-detection* and *intensity* distributions.

In section 2, related work on *Haar-like features*, *convolution with kernels*, *PCA*, *SVM*, *AdaBoost*, *edge detection* and *intensity* are discussed. The implementation of the *AdaBoost* algorithm applied on the different techniques is explained in 3. We discuss our experiments and their results in section 4. Finally, we conclude in section 5 and propose some topics for future research.

2 Background

Finding the main features of dirty money and clean money was a true challenge in this project. In this section are indicated in more detail the ups and downs of the techniques used and the theoretical knowledge that represents the basis of these methods.

2.1 PCA and Adaboost

The *AdaBoost* algorithm is used in combination with different techniques throughout this project, such as: *PCA*, *Haar-like features* or *edge* and *intensity* distributions over different regions.

Adaptive Boosting (also known as AdaBoost) is a machine learning technique which can be used in conjunction with various other learning algorithms. The idea is to have a (convex) set of weak classifiers (classifiers that perform at least better than random) and then minimize the total error over the training-set by finding the best classifier at each stage of the algorithm.

The theoretical basis of this algorithm is that given a set of models (or features), M, the algorithm will determine the subset of T models that are the best for distinguishing between the two classes (fit and unfit bills). Thus, *AdaBoost* will learn the most representative features of the two classes. The algorithm for determining the best models is shown in section 3: *Algorithm 1.*

Another very important characteristic of this algorithm is the fact that it also specifies a method in which the models that were chosen as being the best, can be combined in order to give a strong classifier. The corresponding algorithm can be seen in section 3: *Algorithm 2*.

PCA !!!

2.2 Intensity and Edge distributions

Intensity & Edge!!!

2.3 Haar-like features and Convolution over kernels

The first idea that we have tried was to implement the *Viola & Jones* approach for object detection. The final cascade used in this paper was not implemented because the main idea of this cascade was not suitable for the purpose of this project. We have used the strong classifier computed in *AdaBoost* as the final output.

The first step of the algorithm is defining the

Figure 1: Patterns

patterns, that are mainly matrixes of different dimensions containing low and high values for intensity (-1 and 1) – Figure1 indicates a subset of the patterns used. The algorithm used in the *Viola & Jones* article was designed to loop over all predefined patterns on each position in the image and convolve the specific region of the image with the patterns using the formula: We have started by defining a large number of patterns of different sizes (≈100 patterns), and we have filtered out those that gave bad results when used in the *AdaBoost algorithm* for training the weak

$$Value = \sum_x \sum_y Image(y : y + h, x : x + w). * Pattern,$$

where: h – the heigh of the pattern
w – the width of the pattern

classifiers.

The resulted values for each pattern and location in the image would, then, be used in *AdaBoost* to train an *SVM* classifier. Taking into account the fact that the set of features generated by the algorithm for each pattern and each image, was extremely large, and the training took too much time. In order to solve this problem, we have decided to use just random locations at which to convolve the patterns with a region of the image (that would have the same size with the pattern). Figure 2 indicates what *AdaBoost* would choose as being the most representative 5 features for the front side and rear side of the bills.

The results obtained using *Haar features* were

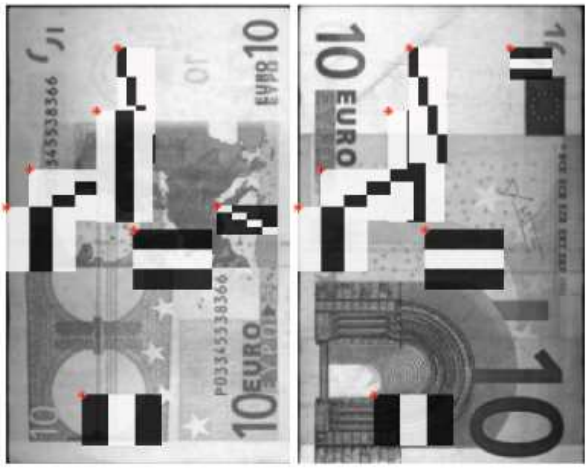

Figure 2: Haar Features for Front and Rear

not as good as we would have expected them to be. An explanation for this may be the fact that the patterns defined were not entirely able to model the dirt that can be found on the bills.

The second approach that we have tried was to define a set of patterns as before, and to segment each image into smaller regions that would be, then, convolved with the patterns.

We have tried using both no-overlapping segmentation of the images and 50% overlapping one. Due to the fact that many essential pixel

values (like those that would indicate the presence of dirt) may happen to be positioned on the line that separates two neighboring regions, is possible that some information will be lost. Taking into account the number of regions generated for all images we can notice that there is a large amount of pixel values that might not be considered in the case of no-overlapping segmentation. As expected, the second method of dividing the images into regions gave better results compared to the first one.

The results obtained using this technique represented the input set of features for the *AdaBoost algorithm*. Figure 3 will help getting a better understanding of how the convolved regions of image and patterns would look like. In this image it is plotted the result obtained when convolving a bill with a simple pattern such as: **[1 -1;-1 1]** with an entire bill.

Thus, the resulted set of models would represent the input features used in *AdaBoost*. For establishing the set of the best T features we have tried using both *SVM* and *Gaussian distribution*, and the results retrieved by the last one seemed slightly better than the ones obtained while using *SVM*. In the case in which *SVM* was used, a model was generated for each available feature. This model would give the best separation between the values corresponding to that feature for fit images and those for unfit images. In the case of the *Gaussian distribution*, the mean and covariance of features corresponding to fit images were computed, respectively the mean and covariance for the features corresponding to unfit images. In order to determine the predicted class of the input images we have tried two methods:

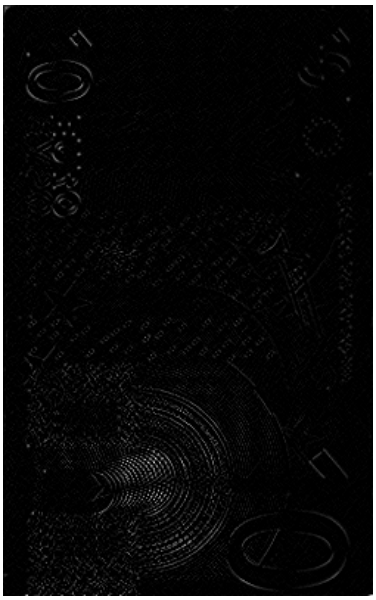

Figure 3: Convolution of an entire bill

$$argmax_{\theta_{fit}} P(\theta_{fit} \mid X) = \frac{P(\theta_{fit}) * P(X|\theta_{fit})}{P(\theta_{fit}) * P(X|\theta_{fit}) + P(\theta_{unfit}) * P(X|\theta_{unfit})}$$

$$argmax_{\theta_{unfit}} P(\theta_{fit} \mid X) = \frac{P(\theta_{unfit}) * P(X|\theta_{unfit})}{P(\theta_{fit}) * P(X|\theta_{fit}) + P(\theta_{unfit}) * P(X|\theta_{unfit})}$$

where:

- $P(\theta_{fit}), P(\theta_{unfit})$ – marginal probabilities of "fit" class, respectively "unfit" class
- $\theta_{fit}, \theta_{unfit}$ – the parameters of the classes (mean and covariance)
- $P(X \mid \theta_{fit}), P(X \mid \theta_{unfit})$ – the conditional probabilities of the two classes (representing normal distributions)

- In the first method we would simply compute the *mean* and *covariance* for the values corresponding to fit, respectively unfit values corresponding to a specific feature. During testing, for each input value, we would compare the two numbers returned by *Gaussian density function* for fit, respectively unfit and we would choose the predicted class to be the maximum of the two

- The second method was using *Naive Bayes* in order to make use of the prior knowledge available. We know that in a real-life situation there are always more fit bills than unfit one and we would like to use this information to improve our classifiers. The predicted class was defined by using the *MAP* (maximum aposterior probability) estimation. The formula for computing the maximum aposterior probability for fit, respectively unfit class is given by:

  It can be easily shown that the values of the parameters θ for a normal distribution

$\mu = \sum_i \frac{x_i}{|X|}$
$\sigma = \frac{(X-\mu)(X-\mu)^T}{|X|}$

where X – the training data set
μ – the mean
σ – the covariance

(mean and covariance) that would maximize the probability of the classes are exactly the corresponding formulas:

The second technique used for determining the predicted class of the data given the parameters seemed to provides slightly better results in practice due to the fact that it also incorporates some prior knowledge of the data.

For this approach we have used two classifiers: one for the rear side and another for the front side of the bills and the predictions given by the two classifiers were combined into a third one using Naive Bayes.

In order to create the final model, we have tried two different methods:

- the first one was essentially just choosing the best model (the one that gives the minimum error) throughout all the repetitions and all the rounds in the cross-validation;

- the second method was to use a system of voting – each time a feature was chosen by the *AdaBoost* algorithm, it would receive a vote. In the end the top T most voted features throughout all the cross-validation rounds and all the repetitions would be chosen from the best model found. The corresponding weights would be generate by taking the mean of the features' α-weights computed in the *AdaBoost* algorithm;

As we would have expected, the second method for defining the best model, proved to give better results in this case so it was chosen to be applied.

Algorithm 1 AdaBoost learning features

function AdaBoostLearn(T, M, S)
T = nr. of hypothesis
M = Models
S = training-set, $\{(x_1, y_1), ...(x_n, y_n)\}$ with $x_i \epsilon X$ and $y_i \epsilon \{-1, 1\}$
$D_{1(i)} \leftarrow \frac{1}{n}$, with $i = 1, ..., n$
for $t = 1$ to T **do**
  $error_t \leftarrow 0$
  for $m \epsilon M$ **do**
    $h_j(x_i) \leftarrow predict(x_i)$ *%svm or gaussian distribution*
    $error_j \leftarrow \sum_{i=1}^{n} D_t(i)[y_i \neq h_j(x_i)]$
    if $error_j < error_t$ **then**
      $error_t \leftarrow error_j$
      $h_t \leftarrow h_j$
  $\alpha_t \leftarrow 0.5 \cdot \log \frac{1-error_t}{error_t}$
  for $i = 1$ to n **do**
    $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp -\alpha_t \cdot y_i \cdot h_t(x_i)}{Z_t}$
return α, h

Algorithm 2 AdaBoost Prediction

function AdaBoostPredict(α, h, I)
α = weights
h = weak classifiers
I = image
p = prediction
for $t = 1$ to $length(\alpha)$ **do**
  $p \leftarrow p + \alpha_t h_t(I)$
return $sign(p)$

3 Implementation

4 Results

The available image set was spilt into a holdout set and the rest of the data was again split into a validation set and a part that was used for training the final model by *random sub-sampling cross-validation* technique. The whole process of defining the validation set and applying random sub-sampling cross-validation was repeated several times to ensure a correct estimation of the predictions. For each round of the cross-validation, a model was trained using the *AdaBoost algorithm* described above. The obtained model was, then, tested by building the strong classifier and computing the corresponding values for: *true-positive estimation*, *true-negative estimation*, *false-positive estimation* and *false-negative estimation*.

4.1 Convolution Results

Due to the fact that the number of features generated for only 21 patterns and a segmentation of 12 regions by 5 of the image gives 1,260 features**???!!!**, the training part is relatively slow for this method. In order to create reliable model we have used 5 repetitions, 10 hypothesis (features to be chosen by AdaBoost), 10 trials in the random sampling cross-validation method.

4.2 Convolution Results

RESULTS PCA

4.3 Convolution Results

RESULTS IE

5 Conclusion

References

[1] P. Viola & M. Jones:
Rapid Object Detection using a Boosted Cascade of Simple Features (CVPR 2001)

[2] AdaBoost:
http://en.wikipedia.org/wiki/AdaBoost