Evolutionary boosting strategy

Xavier Baró* and Jordi Vitrià+*

* *Centre de Visió per Computador*
+ *Dept. de Ciències de la Computació UAB*
Edifici O - Campus UAB, 08193 Bellaterra, Barcelona, Catalonia, Spain
{xbaro,jordi}@cvc.uab.es

Abstract This paper presents a brief introduction to the nowadays most used object detection scheme. From this scheme, we highlight the two critical points of this scheme in terms of training time, and present a variant of this scheme that solves one of these points. Our proposal is to replace the WeakLearner in the Adaboost algorithm by an evolutive method. In addition, this approach allows us to work with high dimensional feature spaces which can not be used in the traditional scheme.

Keywords: Boosting, genetic algorithms, object detection

1 Introduction

Object recognition is one of the most important, yet least understood, aspects of visual perception. For many biological vision systems, the recognition and classification of objects is a spontaneous, natural activity. Young children can recognize immediately and effortlessly a large variety of objects. In contrast, the recognition of common objects is still way beyond the capabilities of artificial systems, or any recognition model proposed so far [1].
Our work is focused in a special case of object recognition, the object detection problem. Object detection can be viewed as a specific kind of *classification*, where we only have two classes: the class *object* and the class *no object*. Given a set of images, the objective of object detection is to find regions in these images which contain instances of a certain kind of object.
Nowadays, the most used and accepted scheme to perform object detection is the one proposed by Viola and Jones in [2]. They use a cascade of weak classifiers based on simple rectangular features. Their scheme allows to do the object detection process in real-time. The main problem of this method is the training time, which increases exponentially with the number of features and samples, because in each iteration of the Adaboost, it must perform an exhaustive search over all the features in order to find the best one. Some strategies have been presented in order to speed-up this method, which affects the initial features set or the search strategy. For instance, in [3] they perform an initial feature selection in order to reduce the initial features set, and in [4] a heuristic search is performed instead of the exhaustive search over all features.
This paper is organized as follows: On section 2 we present a brief explanation of the Viola and Jones object detector, and remark the bottlenecks. Finally, in sections 3 and 4, we present the introduced modifications: an evolutionary strategy in the Adaboost method that not only speeds-up the learning process, but allows the use of Viola and Jones scheme with high-dimensional feature spaces and a redefinition of weak classifiers.

2 Object Detection

In this section we present a brief introduction to Viola's face detector [2], using the feature set extension of Lienhart and Maydt [5]. This method is the canonical example of rare object detection. The main points of their scheme is the choice of simple and fast calculable features, and a cascaded detector, which allows real-time object detection. In this section we will present briefly the features, the final detector structure and concentrate our attention to the learning process.

2.1 Haar-like features

Haar-like features are local differences between contiguous regions of the image (see fig. 1). This kind of features are inspired by the human visual system, and are very robust in front of illumination variance and noise. In addition, they can be calculated very fast using the integral image. In [5], Lienhart and Maydt extended the initial set used by Viola and Jones adding the rotated version of each feature. These new prototypes of features can be also calculated in a short time using the rotated integral image.

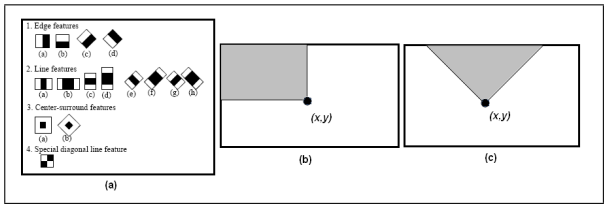

Figure 1: **a)**Extended Haar-like features. The sum of all the pixels under the black regions are subtracted from the sum of all the pixels under the white regions. **b)**Integral image. The value on the point (x, y) is the sum of all pixels under the gray region. **c)**Rotated integral image. The value on the point (x, y) is the sum of all pixels under the gray region.

2.2 Weak Classifiers

A Weak classifier $h : \Re^n \mapsto \{-1, 1\}$ is composed by a Haar-like feature f, a threshold thr and a polarity value p. It is the most low level classification in this detection scheme. Given an input image X, $h(X)$ can be calculated as:

$$h_{f,p,thr}(X) = \begin{cases} 1 & if \quad f(X) \times p \geq thr \times p \\ -1 & otherwise \end{cases} \tag{1}$$

where $f(X)$ is the value of the feature f on the image X.

2.3 Cascade

The use of a cascade of classifiers is one of the keys that allow the real-time detection. The main idea is to discard easy non-object windows in a short time, and concentrate the efforts on the hardest. Each stage of the cascade is a classifier (see fig. 2), composed by a set of weak classifiers, and their related threshold value.

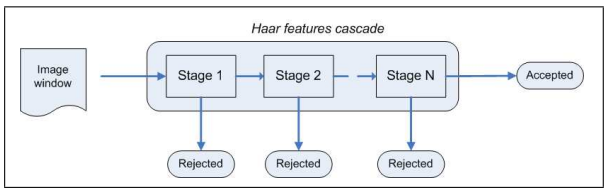

Figure 2: Cascade of classifiers.

2.4 Learning process

Once the basic components are introduced, now we will analyze the learning algorithm, remarking the critical points. Adaboost is an algorithm introduced by Freund and Schapire in [6] to build a strong classifier as a lineal combination of weak classifiers. In fig. 3 the Adaboost algorithm is presented. The step

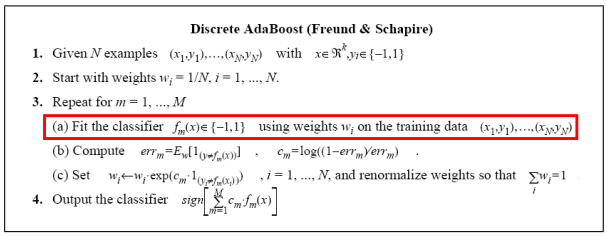

Figure 3: Adaboost algorithm.

3a of the algorithm is also named WeakLearner, because it is where the parameters of the weak classifier are learned. This is the bottleneck of the learning process, and is where we introduce our modifications.
The WeakLearner determines the best combination of feature, threshold and polarity value in order to minimize the weighted classification error. Basically, the steps followed to determine these parameters are:

1. For each feature f in the feature set:
   (a) Calculate the value of over each sample
   (b) Sort all the different values of f and store them into V
   (c) Calculate the error using the values in V as the threshold. Two errors are calculated, one for each polarity value.
   (d) Select the threshold and polarity value that minimizes the error
2. Select the feature, threshold and polarity values that minimize the error.

After analyzing these steps, one can see that we must repeat the step 1 as many times as features are in our feature set. Therefore, in large feature spaces this process will spend a lot of time. In order to reduce the number of iterations, we use a genetic algorithm, which will generate

in each generation a small feature set, reducing the number of iterations of the WeakLearner.
Inside the loop, for each feature we have to use a sorting algorithm, and recalculate the error over each possible threshold and polarity values. The time expended in this case is proportional either to the number of samples as to the number of features. To reduce the time inside the loop, in section 4 we propose the approximation by normal distributions instead of the use of threshold and polarity values.

3 Evolutive WeakLearner

Since Adaboost does not require the selection of the best weak classifier in each iteration, but a weak classifier with an error less than 0.5, an evolutive approach is what seems to be more logical to reduce the time complexity. As a first approach we propose the use of a classic genetic algorithm, which never will select the best feature, but will select a quite good feature.
A gaussian mutation function and a one-point crossover is used. In order to speed-up the convergence of the genetic algorithm, elitism is also used. To guarantee the diversity, we evolute 4 independent populations, and every a certain number of generations, the best individuals of these populations are added to the other populations.
Finally, it must be considered the fact that the convergence of genetic algorithms is not guaranteed for every initialization, or it can converge to a solution with an error higher than 0.5. When this occurs, we reinitialize the genetic algorithm using a new random population.

3.1 Evaluation function

This function provides a measure of performance with respect to a particular chromosome, and is the function that the evolutive algorithm tries to minimize. As we use genetics as a WeakLearner, the function to minimize is the weighted classification error. Therefore, the evaluation function will apply the feature represented by a certain chromosome to all the samples and calculate the weighted classification error.
Given a sample set $\Omega = \{(x_i, y_i) \mid i = 1..N\}$ where $y_i \in \{-1, 1\}$ is the label of x_i, $C \in \{0, 1\}^L$ a chromosome and $W \in \Re^N$ the weights distribution, the evaluation function $\xi : C \times \Omega \times W \mapsto \Re$ can be expressed as:

$$\xi(C, \Omega, W) = \sum_{i:y_i \neq C(x_i)} W_i \qquad (2)$$

where $C(x_i)$ is the value obtained when the feature represented by the chromosome C is applied to the sample x_i.

4 Normal Approximation

In order to speed-up the WeakLearner algorithm, we propose to approximate the values of each feature by two normal distributions, one for the positive samples and the other one for the negative samples (see fig. 4). This process can be addressed in polynomial time, and avoid the calculus of the threshold and polarity values, because once we estimate the two normals, we calculate the value of each distribution for the value of the feature over the sample and pick the class with higher value to classify a sample. This approach not only speeds-up the learning process, but also allows the use of online methods as the ones presented by Oza in [8], in the sense that these methods require incremental Weak Classifiers, and now we have them.

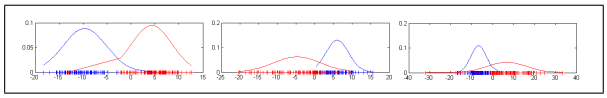

Figure 4: Approximation of the values of three features by pairs of normal distributions.

5 Results

In order to demonstrate usefulness of our approach, we train two cascades using the Adaboost with the genetic WeakLearner. Using the face images from the Caltech database as the positive samples, and patches from the Corel database as negative samples, we build two sample sets, each one with 214 *face* images and 642 *no-face* images. The first one is used as a training set, and new *no-face* images are added at each step of the cascade in order to replace negative samples discarded by the previous stages. The second set is used as a test set, and remains invariant during all the process. The parameters of the training algorithm are set to 0.995 for the hit ratio and 0.5 for the false alarm ratio. The results are showed in table 1.

From the results obtained on our experiment, we see that the resulting method allow us to learn a cascade of detectors with a high discriminative power.
The reduced number of features in the final detector guarantees the real-time capability of the detection process, which can be done by only 8 accesses to the integral image per feature.

Figure 5: Some of the lost faces on the gray-scale and color features cascade.

Evolutive strategies use to be non deterministic, therefor, we prepare another experiment to verify that the learning process always converge in a good way. Using the same

Cascade		Train			Test	
Stages	Num Features	HR	FA	Num Samples	HR	FA
0	0	100.0%	100.0%	642	100.0%	100.0%
1	3	99.5%	39.8%	1,613	99.5%	41.7%
2	3	99.5%	20.5%	3,133	99.5%	21.3%
3	4	99.1%	6.7%	9,575	97.2%	6.1%
4	5	98.6%	2.6%	24,237	97.2%	3.1%
5	5	98.6%	0.5%	128,686	94.9%	0.6%
6	6	98.1%	0.2%	374,828	93.9%	0.2%
7	10	98.1%	0.1%	747,670	93.0%	0.0%

Table 1: Hit ratio and false alarm evolution during the training process. The number of needed patches in order to complete the negative samples set is showed on the *Train* column. This table also shows the number of features used in each step.

training and test sets, we repeat the training process 30 times, and compare the mean error over all the runs to the error obtained with the classical approach. The results are shown in fig. 6, when we see that the mean convergence using the evolutive strategy and the classical method are equal.

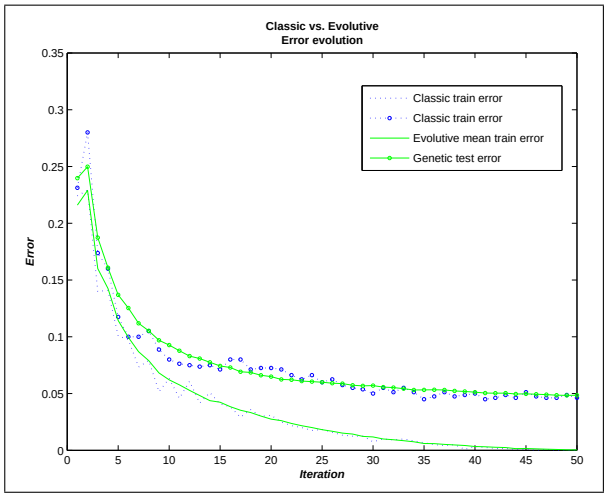

Figure 6: Error evolution using the classical Adaboost algorithm and the evolutive one.

6 Conclusions and future work

We have presented a variation on the Adaboost algorithm using an evolutionary WeakLearner, which remove the computational bottlenecks on the nowadays most used object detection scheme. Since the exhaustive search over the features space is removed, this proposal also allows to use high dimensional feature spaces.
As a future work, we want to change the genetic algorithm by other evolutive strategies which allows a more intelligent evolution, and the addition of extra problem-dependent knowledge.

Acknowledgements

This work has been partially supported by MCYT grant TIC2003-00654, Spain. This has been developed in a project in collaboration with the "Institut Cartogràfic de Catalunya" under the supervision of Maria Pla.

References

[1] S. Ullman, *High-level Vision. Object Recognition and Visual Cognition*, ser. A Bradford Book. New York: The MIT Press, 1996.

[2] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision - to appear*, 2002.

[3] X. Baró and J. Vitrià, "Feature selection with non-parametric mutual information for adaboost learning," *Frontiers in Artificial Intelligence and Applications / Artificial intelligence Research and Development, IOS Press, Amsterdam*, October 2005.

[4] B. McCane and K. Novins, "On training cascade face detectors," *Image and Vision Computing*, pp. 239–244, 2003.

[5] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," *Proc. of the IEEE Conf. On Image Processing*, pp. 155–162, 2002.

[6] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, 1996, pp. 148–156.

[7] Balas and Sinha, "STICKS: Image-representation via non-local comparisons," *Journal of Vision*, vol. 3, no. 9, pp. 12–12, 10 2003.

[8] N. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics 2001*. Morgan Kaufmann, 2001, pp. 105–112.