Adaboost Classifier by Artificial Immune System Model

Hind Taud[1], Juan Carlos Herrera-Lozada[2], and Jesús Álvarez-Cedillo[1]

[1] Centro de Innovación y Desarrollo Tecnológico en Cómputo
[2] Centro de Investigación en Computación
Instituto Politécnico Nacional
Av. Juan de dios Bátiz, Gustavo A. Madero, C.P. 07700, México, D.F.
{htaud,jlozada,jaalvarez}@ipn.mx

Abstract. An algorithm combining Artificial Immune System and AdaBoost called Imaboost is proposed to improve the feature selection and classification performance. Adaboost is a machine learning technique, which generates a strong classifier as a combination of simple classifiers. In Adaboost, through learning, the search for the best simple classifiers is replaced by the clonal selection algorithm. Haar features extracted from face database are chosen as a case study. A comparison between Adaboost and Imaboost is provided.

Keywords: Artificial immune system; Feature selection; Adaboost; Clonal selection algorithm; Haar Features.

1 Introduction

Machine learning is an active research topic in computer vision and pattern recognition research, which is applied in various fields such as the identity authentication; man-machine interface; virtual reality; content-based retrieval and many other aspects. Recently, Viola and Jones [1, 2] developed a method based on Adaboost classifier that performs a high detection rate. On the one hand, this method is considered to be one of the fastest systems and can be used to detect any object. On the other, Adaboost is a machine-learning algorithm that performs two tasks simultaneously: feature selection and forming a classifier using combination of these features. In Adaboost, to select the best features, a search is made of all the features. In the case of a large feature space such as Haar features, a reduction in this search step allows an increase in the performance of the classification.

In this article, we propose a new algorithm called Imaboost which is based on combining the Artificial immune system AIS model with Adaboost . The search step is replaced by the clonal selection algorithm. CLONALG is integrated with Adaboost to improve the feature selection for image classification. The objective is to present the Imaboost system but not to compare it in detail with the many other algorithms. Nonetheless, the results are compared with the original Adaboost algorithm. Haar features extracted from face database are chosen as a case study. Related works, AIS, Adaboost and Haar features are briefly described before the main lines of the algorithm are presented. Experiments and results are provided.

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2010, LNCS 6256, pp. 171–179, 2010

2 Related Works

Viola and Jones Method contains three essential points: Haar features from the integral image, the Adaboost classifier and the cascade structure. The cascade allows non-object to be rejected quickly and subsequently quickens the detector. Features can be calculated extremely fast from the integral image. Adaboost is a machine-learning algorithm that performs two tasks simultaneously: feature selection and forming a classifier using a combination of these features. It is considered to be the key to a high detection rate. Feature selection is a step that follows feature extraction in a pattern recognition process. The aim of this step is to obtain the optimal feature subset from the input space that can achieve the highest accuracy results. Most of feature selection algorithms involve a combinatorial search through the whole space. Usually, heuristic methods, such as hill climbing, have to be adopted, because of the large number of features of input space [3].

Due to the high detection rate and real-time execution of the Viola and Jones approach, different investigations try to enhance the idea of boosting simple weak classifiers or to improve the response of the Adaboost classifier. Lienhart and Maydt [4] showed that extending the basic feature set yields detectors with lower error rates. Li and Zhang [5] described a variant of Adaboost called Floatboost for learning better classifiers. Zhang y al. [6] presented Z-Adaboost and Chang and Lee [7] proposed the Segment-Boost.

The use of Evolutionary Algorithms has received growing interest in the field of automatic learning. Genetic Algorithms are used as optimization procedures inspired by the mechanisms of natural selection. Within Adaboost, which is considered to be an optimization problem, genetic algorithms are used to find better classifiers as proposed by Treptow and Zell [8]. Zin et al. [9] extended the work of these authors by implementing GA inside the Adaboost to select features. Jang and Kim [10] introduced the employment of Evolutionary Pruning that reduces the number of weak classifiers. Chouaib et al. [11] presented a fast method combining genetic algorithm and Adaboost classifiers for feature selection. Li et al. [12] proposed dynamic Adaboost learning with feature selection based on a parallel genetic algorithm.

Just as the GA, the Artificial immune system (AIS) has been also applied successfully to a variety of optimization problems [13]. AIS is a computational intelligence paradigm inspired by the biological immune system, which has found an application in pattern recognition [14] and machine-learning [15].

3 Artificial Immune System

Artificial immune systems (AIS) are computational systems inspired by the principles and processes of the immune system. Formal definition is given by De Castro and Timmis [16]. The algorithms typically exploit different theories and processes, which allow the acquired immunity system to solve a specific problem. Common techniques are the clonal selection algorithm, negative selection algorithm, immune network algorithms and dendritic cell algorithm. The first is chosen for this investigation.

3.1 Clonal Selection Algorithm

Proposed by Castro and Von Zuben [17], CLONALG (CLONal selection ALGorithm) is an algorithm inspired by the clonal selection theory of acquired immunity. As described by these authors, the algorithm starts with an initial set of random solutions called population and iterates over a number of rounds (G) or generations until a specific stopping condition is reached (Fig. 1).

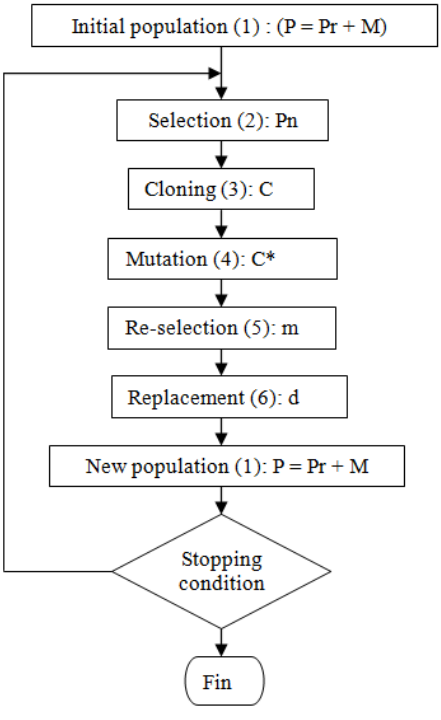

Fig. 1. One Clonal algorithm selection

The following provides the six steps composing CLONALG:

(1) Generate a set (P) of candidate solutions or antibodies, composed of the memory cells (M) and the remaining (Pr) population (P = Pr + M);
(2) Select the n best antibodies (Pn), based on an affinity measure;
(3) Clone these n best antibodies in proportion to their affinity; giving rise to a temporary set of clones (*C*);
(4) Apply a hypermutation to the temporary clones; the degree of mutation is inversely proportional to the affinity. A maturated antibodies is generated (*C**);
(5) Re-select the best elements from *C** to compose the memory set M. Some members of P can be replaced by other improved members of *C**;
(6) Replace d antibodies by novel ones to introduce the diversity concept. The probability to be replaced is inversely proportional to the affinity of the previous remaining (Pr) population.

4 Adaboost and Haar Features

4.1 Adaboost

The Adaboost was introduced by Freund and Schapire [18]. It is a machine learning technique, which combines weak classifiers in an iterative way to generate a final strong classifier through the learning process. A final classifier *H(x)* is a linear combination of the weak or simple classifiers $h_t: X \rightarrow \{0,1\}$

$$H(x) = \begin{cases} 1 & if\ g(x) \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\text{where } g(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \quad (2)$$

Each weak classifier h_t describes a single feature f_t:

$$h_t(x) = \begin{cases} 1 & if\ p_t f_t(x) < p_t \theta_t \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where θ_t is a threshold and p_t is a parity to indicate the direction of the inequality.

Input : N training set (x_i, y_j), i = 1,2, …,N
with negative $(y_j = 0)$ and positive $(y_j = 1)$ examples.

• Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2n}$ where *m* and *n* are the number of negatives and positives examples respectively

• For $t = 1, \ldots, T$:

1) Normalize the weights, $W_{t,i} \leftarrow \frac{Wtj}{\sum_{j=1}^{n} wt,j}$

2) For each feature *j* train classifier $h_{j,t}$ with error $\epsilon_j = \sum_i w_i |h(x_i) - y_i|$

3) Choose the weak classifier h_t with the lowest error ϵ_t

4) Update the weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$ with

$$e_i = \begin{cases} 0 & x_i \text{ is classified correctly} \\ 1 & \text{otherwise} \end{cases}$$

and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

• The final strong classifier is : $C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$

where $\alpha_t = log \frac{1}{\beta_t}$

Fig. 2. Adaboost algorithm

Adaboost (Fig.2) iterates over a number of T rounds. In each round, the features space is scanned in order to train the weak classifiers and to find the threshold θ_t, which discriminates between positive and negative examples. This threshold is calculated as the mean values of features that results on the positive and negative examples [8]. For each feature, the error value $\epsilon_t$is estimated. The best feature with the lowest

error is selected as the weak classifier for this iteration. All training examples are reweighted and normalized. The following iteration is executed and another weak classifier is selected. After T iterations, the resulting strong classifier is formed as a combination of all T weak classifiers.

4.2 Haar Features Extraction

Three kinds of feature are considered: two-rectangle, three-rectangle, and four-rectangle feature. The rectangular regions have the same size and shape and are horizontally or vertically adjacent. The value of each feature (Fig. 3) is the difference between the sum of the pixels within the white and black rectangular regions.

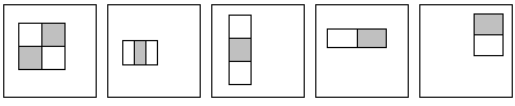

Fig. 3. Haar features

The possible positions and scales of the three basic feature types within a sub window size, for example of 24x24 pixels produce about 160,000 possible alternative features. In order to compute these features very quickly, the integral image representation or Summed-area table [19], is introduced. At the location (*x*, *y*) (Fig. 4(a)) the integral image *II*(*x,y*) contains the sum of the pixels above and to the left of *x, y*:

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x', y') \tag{4}$$

(a) (b)

Fig. 4. Feature Estimation: (a) Integral image at point (*x, y*); (b) Four references to obtain the gray rectangular sum

The integral image can be computed from an image using a few operations per pixel. Therefore, Harr-like features can be calculated at any scale or location in constant time. Any rectangular sum can be estimated by four references (Fig. 4(b)). Two adjacent rectangular sums can be obtained from six array references, eight or nine references in the case of the three-rectangle, and four-rectangle features respectively.

5 Artificial Immune Adaboost: Imaboost

The aim in this article is to apply the artificial immune system in Adaboost to improve the feature selection step. The search over all features in step 2 (Fig. 2) is replaced by

CLONALG (Fig. 1). This algorithm is relatively low in complexity and requires a small number of user parameters such as the number of generations (G), population size (P), memory antibody size (M), selection antibody size (N), remainder replacement size (d), clonal factor (β), Re-select size (m) from the maturate antibodies.

The antibody representation has to be chosen to represent the features and the affinity to resolve the minimization problem. The antibody representation and the affinity are chosen as those given by Treptow and zell [8] and Zin et al. [9]. Every feature is encoded by a string of 5 integer variables (t, x, y, x´, y´) where t represents a type of feature (Fig. 3), (x,y) and (x´,y´) are the coordinate of the upper left and the lower right corner of the feature in the sub-window. The affinity is described as follows:

$$Aff = 1 - \epsilon \tag{5}$$

ϵ is the error function as estimated in Adaboost.

The different operations of the algorithm are directly or inversely proportional to the affinity. The rank based measure is achieved by sorting the set of selected antibodies in ascending or descending order by their affinity. The number of clones created for each antibody is calculated as follows:

$$c_i = \left[\frac{\beta . P}{i} + 0.5\right] \tag{6}$$

where β is a clonal scaling factor, P is the population size, and i is the antibody current rank where $i \in [0, N]$ and N is the number of selected antibodies. The total number of clones is then calculated as a summation of all c_i. The mutation is performed by creating a new type t and changing the corner positions of the feature by adding a random constant in the integer set {-3,…,3}. A probability rate $Pm \in [0,1]$ is used to defined the m and d size.

6 Experiments and Result

The Imaboost is compared with the standard Adaboost to test the response and performance of Imaboost in relation to features selection and classification. The face is chosen as a case study. The training and testing set are obtained from various sources. They consist of 3000 positive and 5000 negative images for the training set (Fig. 5). The testing set, which is different to the training set, consists of 2000 positive and 3000 negatives images. Gray images of size 24×24 are employed.

Fig. 5. Images face / non face set

The clonal algorithm parameters are chosen by testing different values: 50 for population size, 35 for memory antibody size, β=1 for the clonal coefficient. The mutation process decreases by 0.2% with each generation .The algorithm converges and stops when no better solution is found within the next 50 generations. If there is no convergence within the maximum number of generations, set at 300 generations, the algorithm is stopped as well.

With both algorithms, the training step is stopped when all the examples are labeled correctly. The experiments are carried out on an Intel Core 2 Quad Q9550 processor. Imaboost is run 27 times and the average results are taken. The result of applying Adaboost and Imaboost is summarized in Table 1, 2 and 3. Imaboost is able to find classifiers with a lower number of features with less training time compared to Adaboost. The average number of features is 160 for Imaboost compared to 209 for Adaboost. It represents 77% of the number of features selected by Adaboost. The mean time for the search for a weak classifier on the face set is 14.7 seconds for Imaboost compared to 44.8 seconds for Adaboost. The selection of a single feature is then 3 times faster in Imaboost than Adaboost. Reducing the features number and time per iteration implies a reduction of total training time for Imaboost.

Table 1. Features selection

	worst	Best	Average feature
Adaboost	209	209	209
Imaboost	172	148	160

Table 2. Training time

	Average time per iteration(s)	Average Total time (s)
Adaboost	44.8s	9363s
Imaboost	14.7s	2352s

Table 3. Rates on test set

	Classification rate %	False positive rates%
Adaboost	95.9	0.040
Imaboost	96.6	0.031

The learned classifiers are evaluated on the test set to compare classification and false positive rates as shown in Table 3. The learned classifiers use 150 features with Imaboost and 209 with Adaboost. Although a lower features number is used, Imaboost provides similar detection and false positive rates. Imaboost classifies 96.6% of the set correctly whereas Adaboost gives a classification rate of 95.9%.

7 Conclusion and Future Works

In this paper, an approach for feature selection and classification is presented, using a model of artificial immune system. A new combination of Adaboost and clonal algorithm is investigated in order to overcome the problem of feature selection in the huge search space. A comparison between Adaboost and Imaboost applied to a set of face is presented. Preliminary results show that Imaboost improves the performance of the classifier. The number of features and training time is reduced preserving the same classification rate. Moreover, a more thorough study of the performance of Imaboost, requires the performance of more experiments in different image sets.

In order to increase the detection speed, Viola and Jones use a cascade of various Adaboost. Improving each Adaboost implies improving the entire cascade. On the one hand future work can be directed to test the cascade of classifiers produced by Imaboost. On the other hand, in order to enhance the performance of Imaboost, a micro immune system with a reduced population size should be studied. A comparison between different evolutionary algorithms used with Adaboost should also be made.

Acknowledgments

We thank Anna Reid for improving the English grammar and style of our manuscript.

References

1. Viola, P., Jones, M.: Rapid object detection using boosted cascade of simple features. In: Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Hawaii, vol. 1, pp. 511–518 (2001)
2. Viola, P., Jones, M.: Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), 137–154 (2004)
3. Zheng, L., He, X.: Classification Techniques in Pattern Recognition. In: Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Bory, pp. 77–79 (2005)
4. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: IEEE ICIP2, New York, vol. 1, pp. 900–903 (2002)
5. Li, S.Z., Zhang, Z.: FloatBoost Learning and Statistical Face Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1112–1123 (2004)
6. Zhang, W., Tong, R., Dong, J.: Z-AdaBoost: Boosting 2-Thresholded Weak Classifiers for Object Detection. In: IITA08 Second International Symposium on Intelligent Information Technology Application, Shanghai, vol. 2, pp. 839–844 (2008)
7. Chang, W.S., Lee, J.S.: Segment-Boost Learning for Facial Feature Selection. In: Proceedings of the Third International Conference on Convergence and Hybrid Information Technology, vol. 1, pp. 358–363 (2008)
8. Treptow, A., Zell, A.: Combining Adaboost Learning and Evolutionary Search to select Features for Real-Time Object Detection. In: CEC 2004 Congress on Evolutionary Computation, vol. 2, pp. 2107–2113 (2004)
9. Zin, Z.M., Khalid, M., Yusof, R.: Enhanced Feature Selections OF Adaboost training for face detection using genetic algorithm (gaboost). In: Proceedings of the Third IASTED International Association of Science and Technology For Development, Alberta, pp. 34–39 (2007)

10. Jang, J.S., Kim, J.H.: Evolutionary Prunning for Fast and Robust Face Detection. In: CEC 2006 IEEE Congress on Evolutionary Computation, Vancouver, pp. 1293–1299 (2006)
11. Chouaib, H., Ramos Terrades, O., Tabbone, S., Cloppet, F., Vincent, N.: Feature selection combining genetic algorithm and Adaboost classifiers. In: 19th International Conference on Pattern Recognition (ICPR), Tampa, pp. 1–4 (2008)
12. Li, R., Lu, J., Zhang, Y., Zhao, T.: Dynamic Adaboost learning with feature selection based on parallel genetic algorithm for image annotation. Knowledge-Based Systems 23(3), 195–201 (2010)
13. Tan, K.C., Goh, C.K., Mamun, A.A., Ei, E.Z.: An evolutionary artificial immune system for multi-objective optimization. European Journal of Operational Research 187(2), 371–392 (2008)
14. Carter, J.H.: The immune system as a model for pattern recognition and classification. Journal of the American Medical Informatics Association 7(1), 28–41 (2000)
15. Hunt, J.E., Cook, D.E.: Learning using an artificial immune system. Journal of Network and Computer Applications 19, 189–212 (1996)
16. De Castro, L.N., Leandro, N.: Timmis, Jonathan Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)
17. De Castro, L.N., Von Zuben, F.J.: The clonal selection algorithm with engineering applications. In: Workshop Proceedings of GECCO'00, Workshop on Artificial Immune Systems and their Applications, Las Vegas, pp. 36–37 (2000)
18. Freund, Y., Schapire, R.E.: A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence 14(5), 771–780 (1999)
19. Crow, F.C.: Summed-area tables for texture mapping. In: SIGGRAPH '84 Proceedings of the 11th annual conference on Computer graphics and interactive techniques, pp. 207–212. ACM Press, New York (1984)