

# Fast Training Algorithm by Particle Swarm Optimization for Rectangular Feature Based Boosted Detector

Akinori Hidaka<sup>†</sup> and Takio Kurita<sup>‡</sup>

<sup>†</sup>:University of Tsukuba, [hidaka.akinori@aist.go.jp](mailto:hidaka.akinori@aist.go.jp)

<sup>‡</sup>:National Institute of Advanced Industrial Science and Technology

**Abstract** Adaboost is an ensemble learning algorithm that combines many other learning algorithms to improve their performance. Starting with Viola and Jones' researches [14][15], Adaboost has often been used to local-feature selection for object detection. Adaboost by Viola-Jones consists of following two optimization schemes: (1) parameter fitting of local features, and (2) selection of the best local feature. Because the number of local features becomes usually more than tens of thousands, the learning algorithm is very time consuming if ones completely do the two optimizations. In this paper, we propose fast boosting algorithms by using particle swarm optimization (PSO). Proposed learning algorithm is 25 times faster than the usual Adaboost while keeping comparable classification accuracy.

## 1 Introduction

Adaboost is one of the most powerful algorithm among existing ensemble learning methods. Originally, Adaboost was proposed as the algorithm that combines several weak hypotheses (= classifiers) to construct more powerful classifier [3]. Adaboost has a theoretical guarantee that a training error of the ensemble classifier converges on 0 if enough number of hypotheses which have a slightly better estimation performance than random guess are obtained.

Feature selection is a important issue of pattern recognition. Especially for object detection, local-feature selection is effective to improve both accuracy and speed of detectors. Since the aspect of computational difficulty, feature selection is usually performed by forward stepwise selection (FSS) or backward stepwise selection (BSS) [9][10]. Recently, a variant of Adaboost was proposed to create local-feature based face detector, by Viola and Jones [14][15]. Adaboost by Viola-Jones (we call it *Adaboost.VJ* for convenience) performs feature selection from many candidate local-features, called rectangular features (RFs). Due to an easy algorithm and high classification performance, their method of feature selection became popular and is still used by many reseachers for object detection [1][5][13].

However, feature selection by Adaboost.VJ is sometimes time consuming. In recent object detectors, tens of thousands of training images are often used to obtain a sufficient classification per-

formance. In such cases, it is not unusual that the number of candidates amounts to hundreds of thousands. Because Adaboost.VJ is a variant of FSS, all candidate features are reevaluated whenever one feature is selected. Therefore, sometimes the training time exceeds a week. This is inconvenient for users.

It is known that the performance of ensemble classifiers are closely related to the diversity of components in the ensemble [11]. Since the candidate RFs were extracted from all (overlapped) rectangular regions in images, the set of candidates will contain not only sufficient diversity but also quite a few redundancy for feature selection. Although diverse set of candidate features will lead good ensemble performance, too much redundancy gives bad influence in the learning times. For efficient learning, it is needed to study how to reduce unnecessary redundancy in a features set while keeping classification performance of created ensemble.

In this paper, we propose a fast rectangular feature selection algorithm for Adaboost, by using Particle Swarm Optimization (PSO). PSO [8] performs a randomized search, like a genetic algorithm [6] or particle filters [2][4][7]. Considering the coordinates of RF's region to be parameters, PSO does effective search over the parameter space. As a result, the RF with high classification power is found efficiently.

In our experiments, the best proposed algorithm runs about 25 times faster than the Adaboost.VJ while maintaining comparable classification accu-

- 
- Input labeled samples  $\{I_i, y_i\}_{i=1}^N$ . ( $I_i \in R^d$ : sample,  $y_i \in \{0, 1\}$ : class label.)
  - Initialize samples weights: if  $y_i = 1$  then  $w_i = \frac{1}{2p}$ , otherwise  $w_i = \frac{1}{2q}$ . ( $p$ : # of face,  $q$ : # of non-face)
  - for  $t = 1, \dots, T$ 
    - Normalize samples weights:  $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{i=1}^N w_{t,i}}$ .
    - Optimize base-classifiers  $\{b_c\}_{c=1}^C$ :  
 $\{b_t, err_t\} = OptFunc(\{b_c\}_{c=1}^C, \{I_i, w_i, y_i\}_{i=1}^N)$
    - Compute  $\alpha_t = \log((1 - err_t)/err_t)$
    - Update samples weights:  $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot \delta(y_i - b_t(I_i))]$ . ( $\delta(x) = 1$  (if  $x = 0$ ),  $0$  (otherwise).)
  - Final classification function is:  

$$H(I) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \alpha_t b_t(I) \geq \Theta \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise.} \end{cases} \quad (\Theta: \text{Threshold})$$

Fig. 2: Common Adaboost algorithm. *OptFunc* for the described methods are shown in Figs. 3 to 5.

---

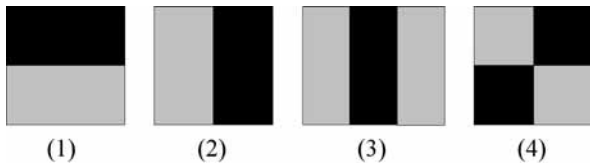


Fig. 1: Configuration of small rectangles.

racy.

## 2 Rectangular Features Based Boosted Detector

### 2.1 Rectangular Features

For object detection problem, Viola and Jones proposed rectangular features (RFs), which indicates difference of brightness between local rectangular regions neighbouring each other, and they treated such a feature as a simple base-classifier [14][15]. RFs consists of two to four small rectangles which are same size and neighbouring each other, so RFs have five degrees of freedom; position  $(x_s, y_s)$ ,  $(x_e, y_e)$  of small rectangles, and configuration  $z \in \{1, 2, 3, 4\}$  of small rectangles (Fig. 1).

Eq. (1) shows the classification function of rectangular features.

$$b(I) = \begin{cases} 1 & \text{if } pf(I) > p\theta \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where  $I$  shows an input image,  $f(I)$  implies a feature value of a rectangular feature at  $I$ , and  $p \in \{1, -1\}$  and  $\theta \in R$  are the parameters determined by training (see [14][15]).

### 2.2 Adaboost for Feature Selection

Adaboost [3] is the ensemble learning method that trains multiple base-classifiers and assembles these to create a more powerful classifier. In each iteration of Adaboost, a classifier that assists the weakness of assembled classifiers is chosen and added into the ensemble. Therefore, the ensemble will effectively obtain a perfect classification power for given training samples.

The algorithms of original Adaboost (Adaboost.M1) is shown in Figs. 2 and 3. In the training of Adaboost, each training sample  $I_i$  is assigned weight  $w_i$  that implies the ‘‘difficulty’’ of sample  $I_i$ . The cost function of Adaboost is designed as weighted classification error rate for training samples. Weight  $w_i$  is made heavy if the base-classifier selected newly misclassified sample  $I_i$ . Therefore, at the next iteration, the base-classifier that can correctly classify the samples which the ensemble fails will be chosen.

In this paper, we use rectangular features as our base-classifiers for the face detection. There are many RFs in training images, and the variant of Adaboost proposed by Viola and Jones performs feature selection from those candidate RFs. For convenience, we call it Adaboost.VJ.

The optimization function for Adaboost.VJ is shown in Fig. 4. Adaboost.VJ only added the feature selection phase to Adaboost.M1. Thus, there are two optimization phases in Adaboost.VJ:

- 1) parameters fitting for candidate classifiers, and
- 2) selection of the best classifiers.

If ones consider that five degrees of freedom in RF are kinds of base-classifier’s parameters, Adaboost.VJ is equivalent to Adaboost.M1.

- 
- Input arguments  $\{b, \{I_i, w_i, y_i\}_{i=1}^N\}$ .
  - Fit a classifier  $b$  to the training samples using weights  $w_i$ .
  - Compute  $err = \frac{\sum_{i=1}^N w_i \cdot \delta(y_i - b(I_i))}{\sum_{i=1}^N w_i}$ .
  - Return  $\{b, err\}$
- 

Fig. 3: Optimization function for Adaboost.M1.

- 
- Input arguments  $\{\{b_c\}_{c=1}^C, \{I_i, w_i, y_i\}_{i=1}^N\}$ .
  - For  $c = 1, \dots, C$ ,
    - Fit a classifier  $b_c$  to the training samples using weights  $w_i$ .
    - Compute  $err_c = \frac{\sum_{i=1}^N w_i \cdot \delta(y_i - b_c(I_i))}{\sum_{i=1}^N w_i}$ .
  - Choose the classifier  $b^*$  with the lowest error  $err^*$ .
  - Return  $\{b^*, err^*\}$
- 

Fig. 4: Optimization function for Adaboost.VJ.

### 3 Particle Swarm Optimization

Particle swarm optimization (PSO) [8] is a search or an optimization algorithm that performs a randomized search in a multi-particle system, like a genetic algorithm [6] or particle filter [2][4][7]. PSO modeled behaviors of swarms of creatures, for example ants looking for food. In PSO, hundreds or thousands of particles search the optimum while communicating with other particles.

Each particle  $p$  has two state vectors: position  $x_\tau^p$  and velocity  $v_\tau^p$ . These state vectors are updated as follows:

$$\begin{aligned} x_{\tau+1}^p &= x_\tau^p + v_\tau^p, \\ v_{\tau+1}^p &= wv_\tau^p + c_s r_s (h_s^p - x_\tau^p) + c_g r_g (h_g - x_\tau^p), \end{aligned}$$

where  $w$  is the inertia term,  $c_s$  and  $c_g$  are parameters given manually,  $r_s$  and  $r_g$  are random values between 0 to 1,  $h_s^p$  and  $h_g$  show the best position in the history of  $p$ -th particle and all particles, respectively. PSO is easy to implement compared with genetic algorithm or particle filter. Each particle communicates with other particles and obtains the current best position  $h_g$ .

As understood from the above rule, PSO has the following action policies:

- 1) keep the same direction as  $v^p$ ,
- 2) go to the direction to  $h_s^p$ ,
- 3) go to the direction to  $h_g$ .

Actually, each particle will go to weighted and randomized average of the three directions.

### 4 Fast Feature Selection by PSO

After the work of Viola and Jones [14][15], many researchers are using Adaboost for local-feature selection [1][5][13]. However, original Adaboost is not designed for the purpose of feature selection. Adaboost.VJ optimizes tens of thousands or more candidates in every step of boosting, while Adaboost.M1 performs only one optimization. It is considered that such multiple optimizations are excessive learning for the purpose of improvement of ensemble's accuracy. The full set of candidate RFs has rich redundancy, but more redundancy to saturate the improvement of ensemble performance is actually not necessary for the training and is the cause of waste of computational cost. Therefore, we have to remove such unnecessary redundancy for the efficient feature selection.

The classification power of ensemble classifier is closely related to the diversity of ensemble [11]. As described in Section 5.2, a simple approach that ones thin out candidate features regularly will not provide sufficient performance. In this paper, we propose Adaboost.PSO algorithm that can reduce the number of optimizations for candidates while maintaining high classification accuracy.

As described in Section 2.1, RFs have five parameters; position  $(x_s, y_s)$ ,  $(x_e, y_e)$  of small rectangles, and configuration  $z \in \{1, 2, 3, 4\}$  of small rectangles (Fig. 1). Considering the state vector as  $x_\tau^p = (x_s, y_s, x_e, y_e, z)_\tau^p$ , the best RF is searched over the state space by the manner of PSO. As a result, the RF with high accuracy is selected efficiently. The optimization algorithm is shown in Figs. 2 and 5.

The computational cost of Adaboost.PSO depends on the two hyperparameters: the number of particles  $P$  and the number of iterations  $T_{it}$ . The total  $P \times T_{it}$  classifiers will be optimized in each step of the boosting.

### 5 Experiments

In this paper, we used MIT CBCL face database [12] for our experiments. The database consists of 2,901 facial and 28,121 background  $19 \times 19$  pixel images. Approximately 53,000 RFs are extracted from each image. We divided all images into training and test set; 2,000 face and 4,000 background

- 
- Input arguments  $\{\{b_c\}_{c=1}^C, \{I_i, w_i, y_i\}_{i=1}^N\}$ .
  - Set  $c_s = c_g = 2, w_{min} = 0.2, w = w_{max} = 1.2$ .
  - Set random parameters:  $r_s, r_g \in [0, 1]$ .
  - Set state vector:  $x_\tau^p \in R^d$  and  $v_\tau^p \in R^d$  at random.
  - For  $\tau = 1, \dots, T_{it}$ ,
    - For  $p = 1, \dots, P$ ,
      - \* Fit a classifier  $b(x_\tau^p; I)$  to the training samples using weights  $w_i$ .
      - \* Compute  $err_\tau^p = \frac{\sum_{i=1}^N w_i \cdot \delta(y_i - b(x_\tau^p; I_i))}{\sum_{i=1}^N w_i}$ .
      - \* Update states of particles:
 
$$x_{\tau+1}^p = x_\tau^p + v_\tau^p,$$

$$v_{\tau+1}^p = w v_\tau^p + c_s r_s (h_s^p - x_\tau^p) + c_g r_g (h_g - x_\tau^p).$$
    - Update momentum:
 
$$w \leftarrow w_{max} - \frac{\tau}{T_{it}} (w_{max} - w_{min})$$
  - Return  $\{h_g, err_{h_g}\}$
- 

Fig. 5: Optimization function for Adaboost.PSO.  $h_s^p$  shows the best position in history of  $p$ -th particle, and  $h_g$  shows the best position in history of all particles.  $err_{h_g}$  implies computed error of  $h_g$ . State vector  $x_\tau^p = (x_s, y_s, x_e, y_e, z)$  implies type of a rectangular feature.  $(x_s, y_s)$  and  $(x_e, y_e)$  are the diagonal apex of small rectangles, and  $z \in \{1, 2, 3, 4\}$  shows the configuration of small rectangles (Fig. 1).

---

images are used for training, and remained images are used for test. Three pairs of training and test sets were generated randomly. All of the accuracies presented in this section are the averaged values calculated from the three pairs.

### 5.1 Adaboost.PSO

First, we compared Adaboost.PSO with various parameters: 10 particles and 10 iterations (written as 10-10 for convenience), 30-20, 50-20, 100-20, 200-10, 200-20 and 200-30. The results is shown in Fig. 6.

Especially in the early stages of boosting, the parameters with small computational cost showed inferior results. When the number of optimized classifiers exceeded 2,000, the improvement of the accuracy became small. Therefore, we adopted PSO-100-20 for comparison of Adaboost.PSO and Adaboost.VJ (Section 5.3).

### 5.2 Adaboost.VJ with Regularly Reduced Features

Next, we compared our approach and Adaboost.VJ in same computational cost. A simple way to reduce the candidate features is to thin out RFs regularly. We use PSO-50-20 which evaluates 1,000 rectangular features in each iteration of boosting.

To arrange the computational cost of Adaboost.VJ, the step length of  $x_s, y_s, x_e, y_e \in [1, 19]$  of RFs is changed to 3 from 1, so the number of RFs for Adaboost.VJ were reduced to 1,068 from 53,130.

Figure 7 shows the comparing result. Adaboost.VJ with regularly reduced candidates marked the bad result. It is considered that the regular reduction of RFs will decrease the diversity of features, and as a result the classification power of ensemble turned worse.

### 5.3 Comparison Results

Finally we compared our Adaboost.PSO and Adaboost.VJ with full set of RFs. As shown in Tab. 1 and Fig. 7, the accuracy of PSO-100-20 is almost equal to Adaboost.VJ. This result shows that PSO has reliable search ability compared with exhaustive search.

The computational cost of our method is 3.8% of Adaboost.VJ using full set of candidates.

## 6 Conclusions

In this paper, we showed that feature selection by a variant of Adaboost (written as Adaboost.VJ) has much redundancy, and that we can save the computational cost of Adaboost.VJ by using Parti-

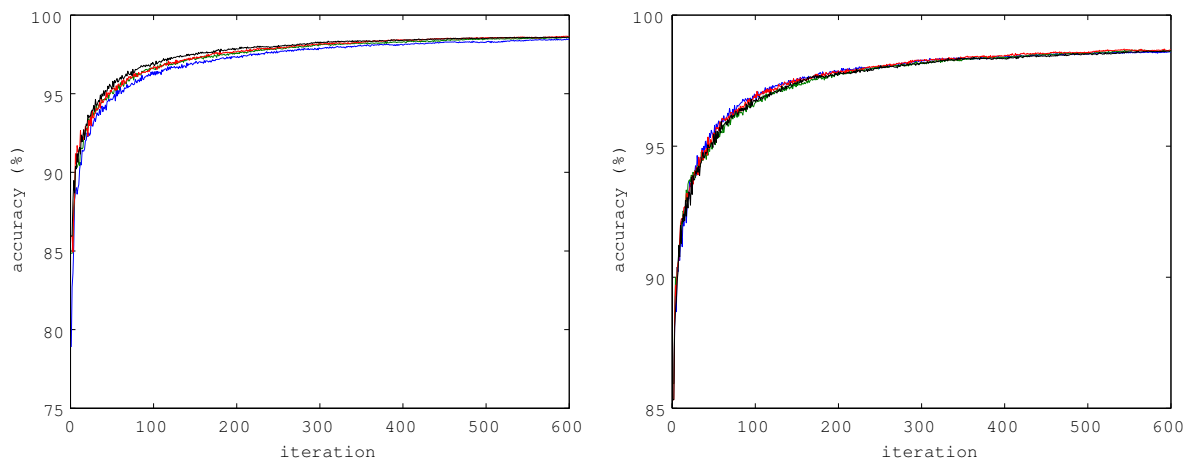


Fig. 6: Classification accuracy of PSO for test sets. Left: black/red/green/blue lines show the experimental results by using 100-20/50-20/30-20/10-10 particles-iterations, respectively. PSO-100-20 (means PSO with 100 particles and 20 iterations) marked the best result. Right: the four lines show the results by using 100-20, 200-10, 200-20, 200-30 particles-iterations.

	VJ	PSO-100-20
Accuracy ( $T_{it} = 600$ )	98.75%	98.58%
# of Optimized Classifiers	53,130	2,000
Computational Cost	100.0%	3.8%

Table 1: Accuracy and computational cost for the test set.

cle Swarm Optimization (PSO) method. Our Adaboost.PSO can make ensembles which has comparable accuracy with Adaboost.VJ, by using only 3.8% of all candidate features in each iteration of the training.

In future works, we would like to investigate about combinations of our algorithm and bagging or random subspace method.

## References

- [1] S. Avidan, “Ensemble tracking,” In IEEE CS Conf. Computer Vision and Pattern Recognition, Jun. 2005.
- [2] A. Doucet, C. Andrieu and S. Godsill, “On Sequential Monte Carlo Sampling Methods for Bayesian Filtering”, Statistics and Computing, vol. 10, no. 3, pp. 197-208, 2000.
- [3] Y. Freund, R. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting”, Journal of Computer and System Sciences, vol. 55, 1, pp. 119–139, 1997
- [4] N. Gordon, D. Salmond and A. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”, In IEE Proc.-F, vil. 140, no. 2, pp.107-113, 1993.
- [5] A. Hidaka, K. Nishida and T. Kurita, “Face Tracking by Maximizing Classification Score of Face Detector Based on Rectangle Features,” Proc. of IEEE International Conf. on Computer Vision Systems (ICVS2006): 48, 2006.01.
- [6] J. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975
- [7] M. Isard and A. Blake, “CONDENSATION – conditional density propagation for visual tracking,” Int. J. Computer Vision, Vol.29, No.1, pp.5-28, 1998.
- [8] J. Kennedy, and R. Eberhart, “Particle Swarm Optimization,” In Proc. IEEE int’l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.

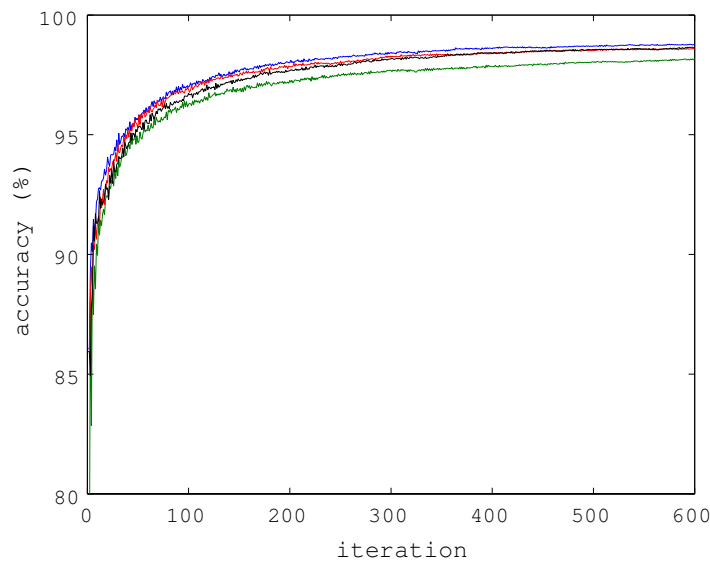


Fig. 7: Classification accuracies for the test set. Blue/red/black/green lines show the experimental results of Adaboost.VJ (with 53,130 RFs)/PSO-100-20/PSO-50-20/Adaboost.VJ (with 1,068 RFs), respectively.

- [9] T. Kurita, K. Hotta and T. Mishima, "Feature Ordering by Cross Validation for Face Detection," Proc. of IAPR Workshop on Machine Vision Applications, pp. 211-214 (2000).
- [10] T. Kobayashi, A. Hidaka and T. Kurita, "Selection of Histograms of Oriented Gradients Features for Pedestrian Detection," Proc. of 14th International Conference on Neural Information Processing (ICONIP 2007), WAB-2, *accepted, now publishing*.
- [11] L. Kuncheva, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy", Machine Learning, vol. 51, 2, pp. 181-207, 2003
- [12] MIT Center For Biological and Computation Learning, "CBCL Face Database #1", <http://www.ai.mit.edu/projects/cbcl>.
- [13] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection", In Proc. of IEEE Int. Con. on Image Processing, 2002.
- [14] P. Viola and M. Jones, "Robust real time object detection," In IEEE ICCV Workshop on Statistical and Computational Theories of Vision, July 2001.
- [15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features,"

In Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, Dec. 2001.

**Hidaka, Akinori:** received the B.Eng degree from Ibaraki University and the M.Eng. degree from the University of Tsukuba, in 2004 and 2006, respectively. He is currently a student of Dr. Kurita in University of Tsukuba. His current research interests include object recognition and tracking based on statistical pattern recognition. He is a member of IEICE.

**Kurita, Takio:** received the B.Eng. degree from Nagoya Institute of Technology and the Dr.Eng. of University of Tsukuba, in 1981 and 1993, respectively. He joined the Electrotechnical Laboratory, AIST, MITI, Japan in 1981. From 1990 to 1991 he was a visiting research scientist at Institute for Information Technology, NRC, Ottawa, Canada. He is currently Deputy Director of Neuroscience Research Institute, National Institute of Advanced Industrial Science and Technology (AIST). His current research interests include statistical pattern recognition and neural networks. He is a member of IEEE Computer Society, IEICE, IPSJ, JNNS, the Behaviormetric Society of Japan, and Japanese Academy of Facial Studies.