Fast Training Algorithm by Particle Swarm Optimization and Random Candidate Selection for Rectangular Feature Based Boosted Detector

Akinori Hidaka and Takio Kurita

***Abstract*— Adaboost is an ensemble learning algorithm that combines many base-classifiers to improve their performance. Starting with Viola and Jones' researches, Adaboost has often been used to local feature selection for object detection. Adaboost by Viola-Jones consists of following two optimization schemes: (1) training of the local features to make base-classifiers, 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 time consuming if the two optimizations are completely performed. To omit the unnecessary redundancy of the learning, we propose fast boosting algorithms by using Particle Swarm Optimization (PSO) and random candidate selection (RCS). Proposed learning algorithm is 50 times faster than the usual Adaboost while keeping comparable classification accuracy.**

I. 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) [10][11]. Recently, a variant of Adaboost was proposed to create local feature based face detector, by Viola and Jones [15][16]. 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][14].

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 performance. 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 [12]. 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) and Random Candidate Selection (RCS). PSO [9] performs a randomized search, like a genetic algorithm [7] or particle filters [2][4][8]. 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 [6]. In RCS, small candidate subsets are extracted at random from all candidates in every iteration of boosting, and normal Adaboost is performed by using these subsets. RCS might seem to give bad influence to accuracy of feature selection. As the results of experiments, we found that RCS can achieve comparable accuracy with Adaboost.VJ and Adaboost.PSO except the early stage of boosting. Furthermore, the combination algorithm of PSO and RCS yielded the best result.

In our experiments, we tested feature selection of Adaboost by using PSO, RCS or these combination. The best result of proposed learning algorithm runs about 50 times faster than the Adaboost.VJ while maintaining comparable classification accuracy.

II. Rectangular Features Based Boosted Detector

A. 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 [15][16]. RFs consists of two to four small rectangles which are same size and neighbouring each other,

Akinori Hidaka is with the Systems and Information Engineering, University of Tsukuba, Japan, and Takio Kurita is with the Neuroscience Research Institute, National Institute of Advanced Industrial Science and Technology (email: hidaka.akinori@aist.go.jp).

978-1-4244-1821-3/08/$25.00©2008 IEEE

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

Fig. 2. Common Adaboost algorithm. *OptFunc* for the described methods are shown in Figs. 3 to 6.

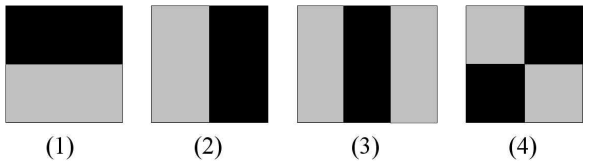


Fig. 1. Configuration of small rectangles.

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.

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

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 [15][16]).

B. 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 the iterations of Adaboost, a classifier that assists the weakness of assembled ensemble is chosen and added into the ensemble. Therefore, the assembled 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.

```
• 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 $c = 1, \cdots, C$,
    – 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.

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.

III. Particle Swarm Optimization

Particle swarm optimization (PSO) [9] is a search or an optimization algorithm that performs a randomized search in a multi-particle system, like a genetic algorithm [7] or particle filter [2][4][8]. 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_τ^p and velocity v_τ^p. These state vectors

```
• 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 $\tau = 1, \cdots, T_{it}$,
    – For $p = 1, \cdots, P$,
        * 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:
            $x_{\tau+1}^p = x_\tau^p + v_\tau^p$,
            $v_{\tau+1}^p = wv_\tau^p + c_s r_s(h_s^p - x_\tau^p) + c_g r_g(h_g - x_\tau^p)$.
    – 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 histry 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).

are simply 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 histry 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.

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

IV. Fast Feature Selection by PSO and RCS

After the work of Viola and Jones [15][16], many researchers are using Adaboost for local feature selection [1][5][14]. 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 [12]. As described in Section V-C, a simple approach that ones thin out candidate features regularly will not provide sufficient performance. In this paper, we propose Adaboost.PSO and Adaboost.RCS algorithms that can reduce the number of optimizations for candidates while maintaining high classification accuracy. These algorithms are described in following sections.

A. Adaboost.PSO

As described in Section II-A, 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 throughout the training.

B. Adaboost.RCS

In Adaboost.VJ, all RFs are used as candidates for optimization and the RF with highest performance is chosen from the candidates. On the other hand, Adaboost.RCS use only the candidates in the small subset of RFs. The subset is selected at random from all candidates, in each iteration of boosting. Therefore, the frequency of optimization can be saved keeping the diversity of the features. The optimization algorithm is shown in Figs. 2 and 6.

RCS directly brings saved computation time because the number of candidates in the subset just implies the computational cost.

V. Experiments

In this paper, we used MIT CBCL face database [13] for our experiments. The database consists of 2,901 facial and 28,121 background 19×19 pixel images. Approximately

```
• Input arguments $\{\{b_c\}_{c=1}^{C}, \{I_i, w_i, y_i\}_{i=1}^{N}\}$.
• Select $K$ classifiers $\{b_k\}_{k=1}^{K}$ at random ($K < C$).
• For $k = 1, \cdots, K$,
    – Fit a classifier $b_k$ to the training samples using weights $w_i$.
    – Compute $err_k = \frac{\sum_{i=1}^{N} w_i \cdot \delta(y_i - b_k(I_i))}{\sum_{i=1}^{N} w_i}$.
• Choose the classifier $b^*$ with the lowest error $err^*$.
• Return $\{b^*, err^*\}$
```

Fig. 6. Optimization function for Adaboost.RCS.

53,000 RFs are extracted from each image. We divided all images into training and test set; 2,000 face and 4,000 background 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.

We describe four classification experiments as follows.

A. Adaboost.RCS

First, we examined Adaboost.RCS with 1, 10, 100, 500, $1,000$, $1,500$, and $2,000$ random candidates. Figure 7 shows the relationship between candidate numbers of Adaboost.RCS and the ensembles' accuracy.

In our experiments, it seems the classification performances were saturated when the number of candidates reached about 1,000. In later stage of boosting, Adaboost.RCS has comparable accuracy with Adaboost.VJ (Tab. I, Fig. 10).

However, the accuracy of Adaboost.RCS in early stage is slightly worse than other methods. It is considered that the first several classifiers selected by Adaboost will have an important role in the ensemble. Therefore, we have to choose surely good RFs in the early stage of boosting. The improvement of this point is argued in Section V-D.

B. Adaboost.PSO

Second, we examined Adaboost.PSO with 10 particles and 10 iterations (written as 10-10 for convenience), 30-20, 50-20, 100-20, 200-10, 200-20 and 200-30 (Fig. 8).

Comparing with Adaboost.RCS, the differences of accuracies between PSO with various computational cost were relatively small. As shown in Tab. I and Fig. 10, the accuracy of Adaboost.PSO in early stage is almost equal to Adaboost.VJ. This result shows that PSO has reliable search ability compared with exhaustive search.

On the other hand, Adaboost.PSO has slightly low performance in later stage of boosting. This phenomenon may be caused by history-based searching of Adaboost.PSO. In the PSO, each particle reffers the search history of oneself and others. And, many particles consentrate near the best position which has found so far. Therefore, the practical diversity of candidate features will be decreasing. To solve this problem, we propose the hybrid algorithm of Adaboost.PSO and Adaboost.RCS in Section V-D.

C. Regularly Reduced Features

Third, we compared our random approach and Adaboost.VJ in same computational cost. A simple way to reduce the candidate features is to thin out RFs regularly. We use RCS-1000 and 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.

Figure 9 shows the comparing result. Adaboost.VJ with regularly reduced candidates marked bad results. 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.

D. The Hybrid Algorithm

From these experiments, it is found that Adaboost.PSO and Adaboost.RCS have good performances as well as Adaboost.VJ even if only use 1,000 candidates. And Adaboost.RCS and Adaboost.PSO are slightly weak in early or later stage of boosting, respectively.

However, these algorithms can supplement their weak points each other. As described Section V-A, it is important to select superior features as first several weak classifiers. So we tested the hybrid algorithm which performs PSO-300-30 from 1st to 10th iteration, and RCS-1000 from 11th to 600th iteration.

The results are shown in Tab. I, Fig. 10. Our hybrid algorithm has comparable accuracy with Adaboost.VJ while reducing computational cost to 1/50.

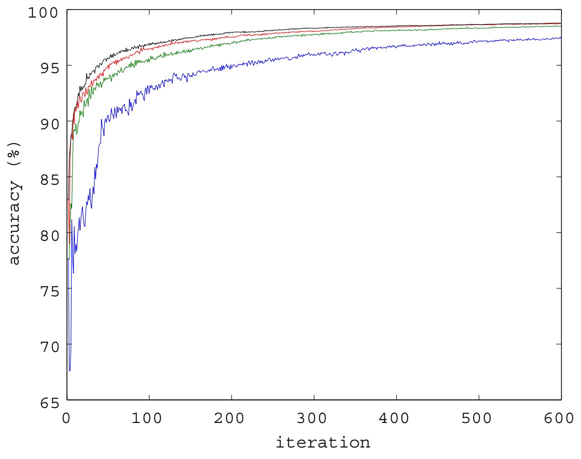
VI. Conclusions

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 Particle Swarm Optimization (PSO) and Random Candidate Selection (RCS). Adaboost.RCS and Adaboost.PSO can make emsembles which has comparable accuracy with Adaboost.VJ, by using only 1.9% to 3.8% of all candidate features in each iteration of the training. The best proposal algorithm, PSO-RCS hybrid algorithm, runs about 50 times faster than the Adaboost.VJ while maintaining comparable classification accuracy.

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 online learning and an application to boosting", Journal of Computer and System Sciences, vol. 55, 1, pp. 119–139, 1997



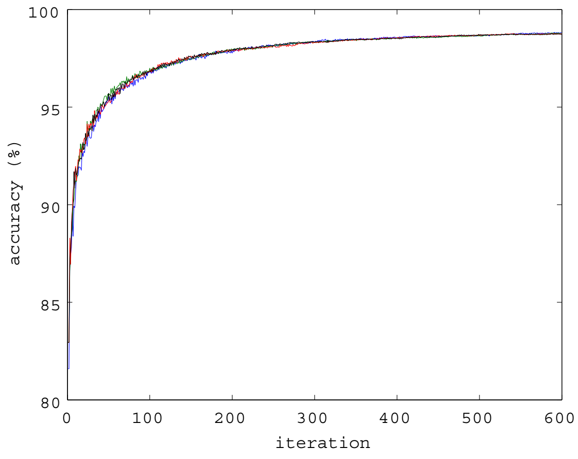
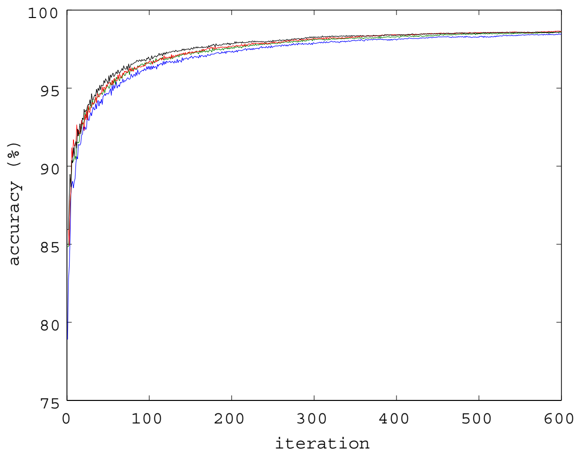


Fig. 7. Classification accuracy of RCS for test sets. Left: black/red/green/blue lines show the experimental results by using 1,000/100/10/1 candidates, respectively. RCS-1000 marked the best result. Right: the four lines show the results by using 500, 1,000, 1,500, 2,000 candidates. In spite of the difference of computational cost, classification results was not so different.



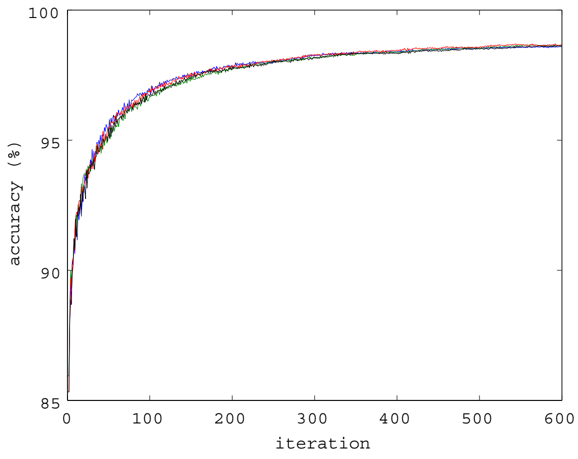


Fig. 8. Classification accuracy of PSO for test sets. Left: black/red/green/blue lines show the experimental results by using 100-20/50-20/30-20/10-10 particles-iterations, respectively. PSO-100-20 (means PSO with 100 particles and 20 iterations) marked the best result. Right: the four lines show the results by using 100-20, 200-10, 200-20, 200-30 particles-iterations. There was no remarkable difference in these four results.

	VJ	RCS-1000	PSO-100-20	PSO-300-30+RCS-1000
Accuracy ($T_{it} = 5$)	90.12%	88.48%	88.68%	89.88%
Accuracy ($T_{it} = 600$)	98.75%	98.78%	98.58%	98.77%
# of Optimized RFs	$53,130$	$1,000$	$2,000$	$1,133.33$ (averaged)
Computational Cost	100.0%	1.9%	3.8%	2.1%

TABLE I

ACCURACY AND COMPUTATIONAL COST OF EACH METHOD.

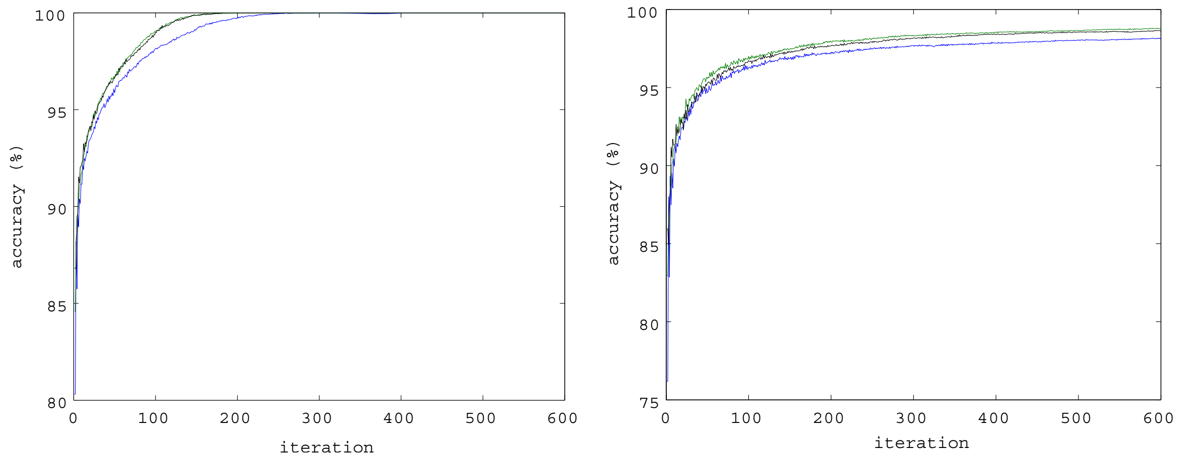


Fig. 9. Comparison of Adaboost.VJ, RCS and PSO in same computational condition. Black and green lines show accuracy of PSO-50-20 and RCS-1000, respectively. Both methods evaluate 1,000 candidate classifiers in each iteration. Blue line shows accuracy of Adaboost.VJ by using regularly reduced candidate fetures. To arrange the computational cost in the same condition, candidate features for Adaboost.VJ were regularly thinned out and reduced to 1,068. The left and right graphs show training and test accuracy, respectively. The result of Adaboost.VJ is worse than other two methods.

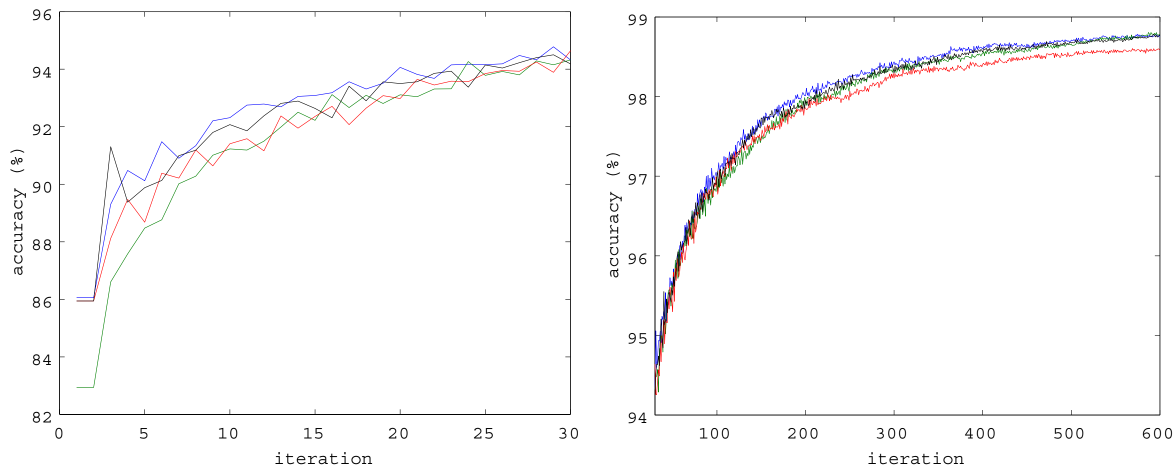


Fig. 10. Comparison of Adaboost.VJ, RCS, PSO and PSO-RCS hybrid algorithms. Top row shows training results and bottom one shows test results. Left column shows to 30-th iteration, and right one shows to 600-th iteration. Blue/green/red/black lines imply the results of Adaboost.VJ/RCS/PSO/PSO-RCS. Proposal hybrid algorithm has comparable performance with Adaboost.VJ in the test set.

[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] A. Hidaka and T. Kurita, "Fast Training Algorithm by Particle Swarm Optimization for Rectangular Feature Based Boosted Detector," 14th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV 2008), *Submitted*.

[7] J. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975

[8] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking," Int. J. Computer Vision, Vol.29, No.1, pp.5-28, 1998.

[9] 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.

[10] T. Kurita, K. Hotta and T. Mishima, "Feature Ordering by Cross Validataion for Face Detection," Proc. of IAPR Workhop on Machine Vision Applications, pp. 211-214 (2000).

[11] 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*.

[12] L. Kuncheva, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy", Machine Learning, vol. 51, 2, pp. 181–207, 2003

[13] MIT Center For Biological and Computation Learning, "CBCL Face Database #1", http://www.ai.mit.edu/projects/cbcl.

[14] 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.
[15] P. Viola and M. Jones, “Robust real time object detection,” In IEEE ICCV Workshop on Statistical and Computational Theories of Vision, July 2001.
[16] 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.