Boosting Soft-Margin SVM with Feature Selection for Pedestrian Detection

Kenji Nishida and Takio Kurita

Neuroscience Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568 Japan
kenji.nishida@aist.go.jp

Abstract. We present an example-based algorithm for detecting objects in images by integrating component-based classifiers, which automaticaly select the best feature for each classifier and are combined according to the *AdaBoost* algorithm. The system employs a soft-margin SVM for the base learner, which is trained for all features and the optimal feature is selected at each stage of boosting. We employed two features such as a histogram-equalization and an edge feature in our experiment. The proposed method was applied to the MIT CBCL pedestrian image database, and 100 sub-regions were extracted from each image as local-features. The experimental results showed fairly good classification ratio with selecting sub-regions, while some improvement attained by combining the two features, histogram-equalization and edge. However, the combination of features could to select *good* local-features for base learners.

1 Introduction

In this paper, we present an example-based algorithm for detecting objects in images by integrating component-based classifiers, which **automatically** select the best local-feature for each classifier and are combined according to the *AdaBoost*[1] algorithm. Our method can be applied to any object composed of distinct identifiable parts that are arranged in a well-defined configuration, such as cars and faces. However, we focused on the pedestrian detection in images, which could be used in driver assistance systems and video surveillance systems. Pedestrian detection is more challenging than detecting other objects such as cars and faces, since people take a variety of shapes and it is nontrivial to define a single model that captures all of these possibilities.

Gavrila[3] employed hierarchical template matching to find pedestrian candidates from incoming images. His method provide multiple templates in advance that were outline edge images of typical pedestrians, and dissimilarities (or similarities) between the edge feature of incoming images were measured by the chamfer distance. The variety of shapes of pedestrians was accommodated with the variety of templates, which bound system performance.

Viola et al.[4] presented a pedestrian detection system that integrated image intensity information with motion information. Their detection algorithm scaned a detector over two consecutive frames of a video sequence, and the detector was trained using *AdaBoost* to take advantage of both motion and appearance information. They achieved a high detection speed (about four frames/second) and a very low false positive rate, while combining two different modalities of information in one detector.

Although they showed the advantage of integrating motion information, it is still difficult to apply their algorithm to an on-board pedestrian detection system, since canceling out the movement of the camera only from visual information is difficult. Therefore, we focused on pedestrian detection from static images to achieve our example-based object detection method.

Mohan et al.[2] applied an Adaptive Combination of Classifiers (ACC) to pedestrian detection. Their system consisted of two stage hierarchical classifiers. The first stage was structured with four distinct example-based classifiers, which were separately trained to detect different component of pedestrians, such as the head, legs, right arm, and left arm. The second stage had an example-based classifier which combined the results for the component detectors in the first stage to classify the pattern as either a "person" or a "non-person". A Support Vector Machine (SVM)[5][6][7] is employed for each classifier. Their results indicated that combination of component-based detectors performed better than a full-body person detector. The components in their system were determined in advance and they were not exactly optimal to classify the examples.

We employed a feature-selection algorithm in the training phase of each component-based classifier, so that the classifier could automatically select the optimal component to classify the examples. Mohan et al.[2] pre-defined the number of the component-based classifiers as four; however, our proposed method combines a larger number of component-based classifiers with the *AdaBoost* algorithm.

Our experimental results show that the proposed method achieves a fairly good classification ratio by selecting a sub-region of the input image, while a slightly greater improvement is achieved by selecting the optimal feature from histogram-equalization images and edge images of inputs.

We describe our object detection method in the next section, and the experimental results are presented in the final section.

2 System Configuration

Our key-idea is introducing feature selection and the soft-margin SVM into *AdaBoost* to enhance the generalization ability of a strong learner by automatically selecting the best feature for base learners at each boosting step. We describe our boosting algorithm with feature selection and outline the soft-margin SVM in this section. We also describe the sample images and experimental conditions we used for our experiments.

2.1 Algorithm

Figure 1 presents the overall algorithm for our pedestrian detection system, which introduces a feature selection algorithm into each step of *AdaBoost*. We define a local-feature as the combination of a feature (a characteric extracted from an input image such as an edge feature) and a sub-region of an input image. Our feature selection method selects the best local-feature (with the lowest error ratio) at each boosting step.

```
1. Let $N$ be the number of samples, $M$ be the number of boosting steps, $L$ be the number of sub-regions, and $K$ be the number of features. Thus, $K \times L$ is the total number of local-features in the local-feature pool.
2. Generate a local-feature pool for all local-features from input samples $\boldsymbol{x}$, such as $\boldsymbol{x} \rightarrow \boldsymbol{x}^{11}, \ldots, \boldsymbol{x}^{1L}, \boldsymbol{x}^{21}, \ldots, \boldsymbol{x}^{kl}, \ldots, \boldsymbol{x}^{KL}$.
3. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
4. For $m = 1$, to $M$:
   (a) For $k = 1$ to $K$, for $l = 1$ to $L$
       i. Fit classifier $G_m^{kl}(x^{kl})$ to the training samples of local-feature $x^{kl}$ which are randomly selected depending on weights $w_i$ from all the training samples
       ii. Compute $err_m^{kl} = \frac{\sum_{i-1}^{N} w_i I(y_i \neq G_m(x_i^{kl}))}{\sum_{i=1}^{N} W_i}$ .
   (b) Set $err_m$ with the smallest $err_m^{kl}$, $l = 1, 2, \ldots, L, k = 1, 2, \ldots, K$.
   (c) Set $G_m(x) \leftarrow G_m^{kl}(x^{kl})$ with $k$ and $l$ in the above step.
   (d) Compute $\alpha_m = \log((1 - err_m)/err_m)$.
   (e) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
5. Output $G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$.
```

Fig. 1. *AdaBoost* with Feature Selection

Initially, features (in our experiment, histogram-equalizaton and edge features are employed) are extracted from input images and a pre-defined number (in our experiment: 100) of sub-region images for each feature are generated as a local-feature pool (Fig. 2) with equal sample weight. In the ith boosting step, the ith base learner is trained under the sample weights determined by the $i-1$th base learner for all local-features in the local-feature pool, and it selects the best local-feature for ith base learner. The sample weights for the next boosting step and significance of the base learner are computed according to the error ratio. Variations in features and sub-regions are determined by a trade-off in the feasible CPU time and desired precision.

2.2 Boosting Soft-Margin SVM

We employed a soft-margin SVM for the base learner. We will first describe a SVM briefly followed by a description of the soft-margin SVM.

The classification function is given as

$$y = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x} - h), \tag{1}$$

where $\boldsymbol{x}$ stands for the input vector, $\boldsymbol{w}$ stands for the weight vector of input, and h stands for the threshold. Function sign(u) is a sign function, which outputs 1 when $u > 0$ and outputs -1 when $u \leq 0$. The SVM determines the separating hyperplane with a maximal margin (distance), which is the distance between the separating hyperplane and the nearest sample. If the hyperplane is determined, there exists a parameter to satisfy

$$t_i(\boldsymbol{w}^T \boldsymbol{x_i}) \geq 1, \;\; i = 1, \ldots, N, \tag{2}$$

where t_i stands for the correct class label for input vector x_i. This means that the samples are separated by two hyperplanes of H1: $\boldsymbol{w}^T \boldsymbol{x}_i - h = 1$ and H2: $\boldsymbol{w}^T \boldsymbol{x}_i - h = -1$, and no samples exist between them. The distance between the separating hyperplane and H1 (or H2) is defined as $1/\|\boldsymbol{w}\|$. Determining parameters $\boldsymbol{w}$ and h that give a maximal margin is defined as an optimization problem for the following evaluation function

$$L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3}$$

under constraint

$$t_i(\boldsymbol{w}^T \boldsymbol{x}_i - h) \geq 1, \;\; i = 1, \ldots, N. \tag{4}$$

A soft-margin SVM allows some training samples to violate hyperplanes H1 and H2. When the distance from H1 (or H2) is defined as $\xi_i/\|\boldsymbol{w}\|$ for the violating samples, the sum

$$\sum_{i=1}^{N} \frac{\xi_i}{\|\boldsymbol{w}\|} \tag{5}$$

should be minimized. Therefore, a soft-margin SVM is defined as an optimization problem for the following evaluation function

$$L(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \xi_i \tag{6}$$

under constraint

$$\xi_i \geq 0, \;\; t_i(\boldsymbol{w}^T \boldsymbol{x}_i - h) \geq 1 - \xi_i, \;\; i = 1, \ldots, N \tag{7}$$

where C stands for the **cost** parameter for violating hyperplane H1 (or H2). Solving this problem with optimal solution $\boldsymbol{\alpha}^*$, the classification function can be redefined as

$$\begin{aligned} y &= \mathrm{sign}(\boldsymbol{w}^{*T} \boldsymbol{x} - h^*) \\ &= \mathrm{sign}(\sum_{i \in S} \alpha_i^* t_i \boldsymbol{x}_i^T \boldsymbol{x} - h^*). \end{aligned} \tag{8}$$

The samples are grouped with α_i^*; sample x_i is classified correctly when $\alpha_i^* = 0$, when $0 < \alpha_i^* < C$, sample x_i is also classfied correctly and is located on the hyperplane H1 (or H2) as a support-vector. If $\alpha_i^* = C$, sample x_i becomes a support-vector but is located between H1 and H2 with $\xi \neq 0$.

The kernel-trick, which drastically improved the performance of the SVM, can also be applied to the soft-margin SVM. In Kernel-Trick, the input vectors are transformed by non-linear projection $\phi(\boldsymbol{x})$ and linearly classified in the projected space. Since SVM depends on the product of two input vectors, the product of the input vectors in projected space can be used instead of computing the non-linear projection of the each input vector, such as

$$\phi(\boldsymbol{x}_1)^T\phi(\boldsymbol{x}_2) = K(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{9}$$

K is called a *Kernel Function* and is usually selected as a simple function, a Gaussian function

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(\frac{-||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2}{2\sigma^2}\right) \tag{10}$$

for instance. The classification function can be redefined by replacing input vectors with kernel functions, as follows

$$\begin{aligned} y &= \mathrm{sign}(\boldsymbol{w}^{*T}\phi(\boldsymbol{x}) - h^*) \\ &= \mathrm{sign}(\sum_{i\in S}\alpha_i^* t_i \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) - h^*) \\ &= \mathrm{sign}(\sum_{i\in S}\alpha_i^* t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h^*). \end{aligned} \tag{11}$$

Introducing cost parameter C, we could have two choices to achieve sample weighing with *AdaBoost*, the first is building the SVM by defining a cost parameter as the weight of each sample, the second is re-sampling according to the sample weights. Schwenk et al.[8] showed that defining a pseudo-loss function and re-sampling had similar effects with *AdaBoost*. We therefore selected re-sampling so that we could use LIBSVM[9] for our evaluation. One thousand images were re-sampled from 1,400 in the training samples.

2.3 Sample Images and Local Features

We employed the MIT CBCL database for the sample data, which contained 926 pedestrian images with 128×64 pixels, and we collected 2,000 random non-pedestrian images. We reduced the resolution of all the samples to 64×32 before we applied them to our system. We extracted histogram-equalization and edge features from the input images, and local-features were extracted as adequate sub-regions of the featured images. We selected 100 sub-regions for our evaluation; extracting small sub-regions from input images with a variety of region sizes such as 4×8 pixels ranging to the whole image.

Figure 2 shows the original image, extracted features, and local features. We selected 700 pedestrian and 700 non-pedestrian images for training, and 200 pedestrian and 200 non-pedestrian images to test the generalization error.

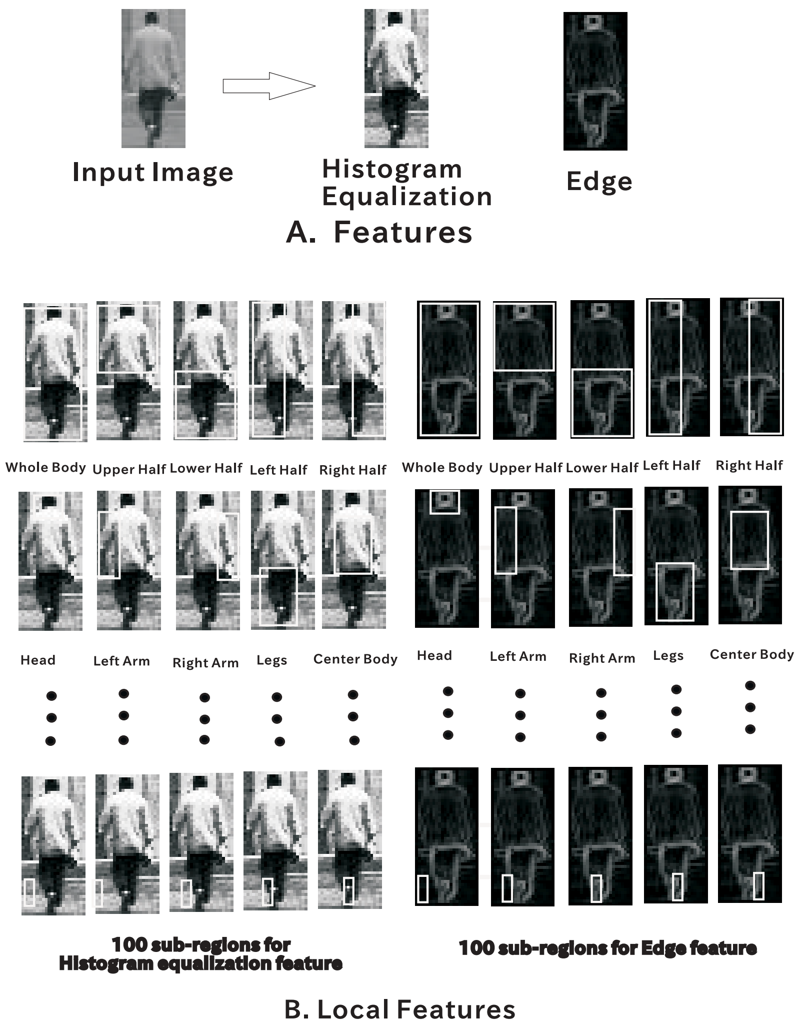

Fig. 2. Sample Images and Local Features

3 Results

We first examined the effect of SVM cost parameter C by using a raw input image with ten sub-regions, such as the whole body, upper half, lower half, right half, left half, right arm, left arm, head, legs, and center body. Figure 3 plots error ratio against the number of boosting steps with cost parameter C for 0.1, 0.7, and 100. Ten local features are almost evenly selected at each boosting step. After 100 boosting steps, test error reaches 4% with C=0.7, 4.5% with C=100, and 5.25% with C=0.1. Figure 4 plots test error against cost parameter C after 50 boosting steps. Test error records a minimal value of 4% at c=0.7. This indicates

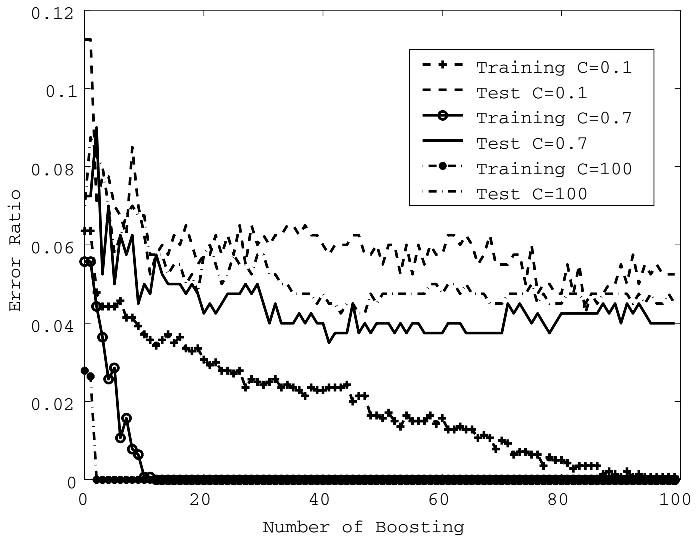

Fig. 3. Error Ratio for Ten Local-Features

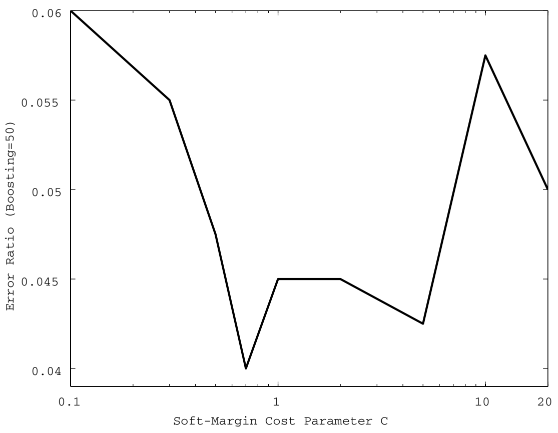

Fig. 4. Error Ratio against Cost Parameter C

that the soft-margin SVM advantageous to usual hard-margin (or firm-margin) SVM for boosting; however, there exists an optimal value for cost parameter C.

Figure 5 plots error ratio against the number of boosting steps with cost parameter C for 0.7, accorcding to the previous results. The experimantal results were averaged over three trials.

The training error converges to zero at boosting step 5 for histogram-equalization, and at boosting step 10 for Edge features. This indicates the histogram-equalization tends to have lower training error than edge features.

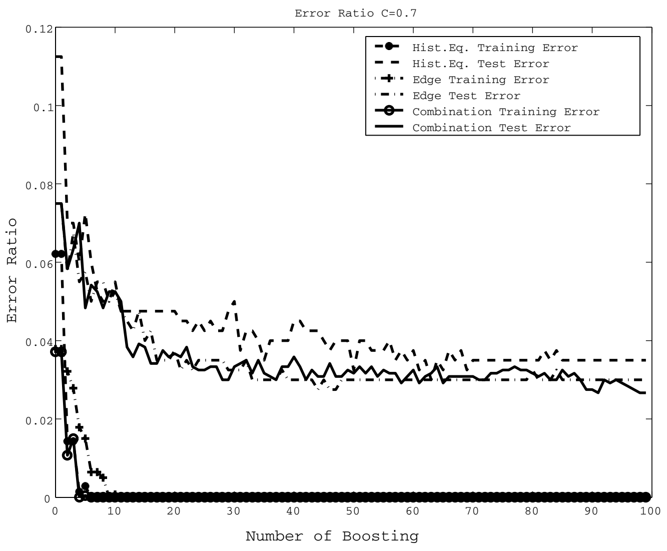

Fig. 5. Error Ratio for 100 (200 for Combination) Local-Features

Athough histogram-equalization has the lower training error, edge features has the lower test error after 100 boosting steps, 3% for edge features and 3.5% for histogram-equalization.

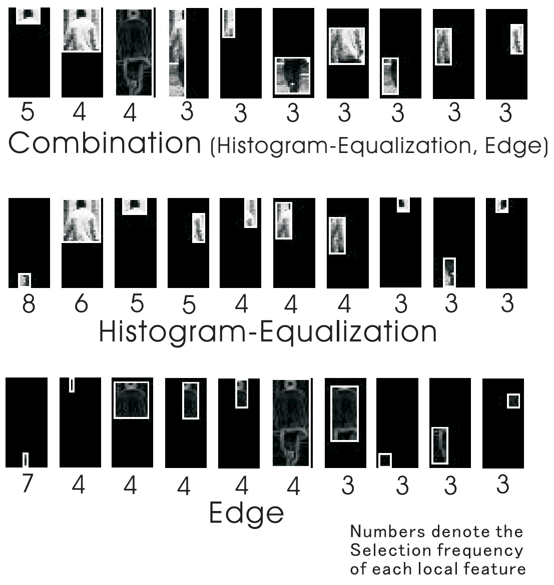
5 4 4 3 3 3 3 3 3 3
Combination (Histogram-Equalization, Edge)
8 6 5 5 4 4 4 3 3 3
Histogram-Equalization
7 4 4 4 4 4 3 3 3 3
Edge
Numbers denote the Selection frequency of each local feature

Fig. 6. Example of Selected Local Features

Selecting the best local feature from the combination of histogram-equalization and edge features, test error reached 2.7% after 100 boosting steps. Eighty-four boosting steps selected histogram-equalization for their base-learners, and 16 steps selected edge features. This indicates that the test error is improved by only 16 classifiers selecting edge features, since test error for histogram-equalization tends to be higher than that of edge feature.

Although we provided 100 sub-regions, only about 50 sub-regions were selected. Figure 6 shows examples of selected local features. This indicates that pedestrian components are automatically selected by our local-feature selection. Small sub-regions, which are not meaningful as pedestrian components, tend to be selected with one feature (such as histogram-equalization or edge), while large sub-regions, which can be meaningful as pedestrian components, are selected with a combination of the two features. Therefore, the combination of features selects *good* local-features.

Table 1. The Error Ratio Comparison

	Error Ratio
Gavrila	10-40%*
Mohan	1-2%
Viola	10%
Our Result	2.7-3%

* For first stage

Table 1 compares error ratio against previous research. Our results achieved a better error ratio than Gavrila and Viola et al., while it was a little worse than Mohan's. Considering the difference in non-pedestrian data, we concluded that our result had almost the same performance as that achieved by Mohan et al.

4 Conclusion

We presented an object detection method that was achieved by boosting the soft-margin SVM with feature selection. In this paper, we focused on pedestrian detection using a combination of two features, histogram-equalization and edge features. The experimental results showed a fairly good generalization error ratio of 2.7%. The good components were automatically selected by local-feature selection by the combining the two features.

We had to limit the number of sub-regions to 100 in this paper, because we had limited computational time to train the classifiers. We are planning to evaluate with a larger number of sub-regions to prove our method can automatically extract good pedestrian components.

References

1. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, Springer-Verlag, USA, 2001.
2. A. Mohan, C. P. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 4, pp. 349-361, 2001.
3. D. M. Gavlira, "Pedestrian Detection from a Moving Vehicle", *Proc. of European Conference on Computer Vision*, pp. 37-49, 2000.
4. P. Viola, M. J. Jones and D. Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance", *Proc. of Int'l Conf. on Computer Vision*, pp. 734-741, 2003.
5. V.N.Vapnik, *Statistical Learning Theory*, John Wiley & Sons, USA, (1998).
6. B. Scholkopf, C. J. C. Burges, and A.J.Smola, *Advances in Kernel Methods - Support Vector Learning*, The MIT Press, USA, 1999.
7. N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, 2000.
8. H. Schwenk and Y. Benjio, "Boosting Neural Networks", *Neural Computation*, Vol. 12, pp. 1869-1887, 2000.
9. C. C. Chung and C. J. Lin, "LIBSVM: a libraly for support vector machines", *Software available at* `http://www.csie.ntu.edu.tw/`, 2001.
10. R. E. Schapire, "The Boosting Approach to Machine Learning: An OverView", *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.